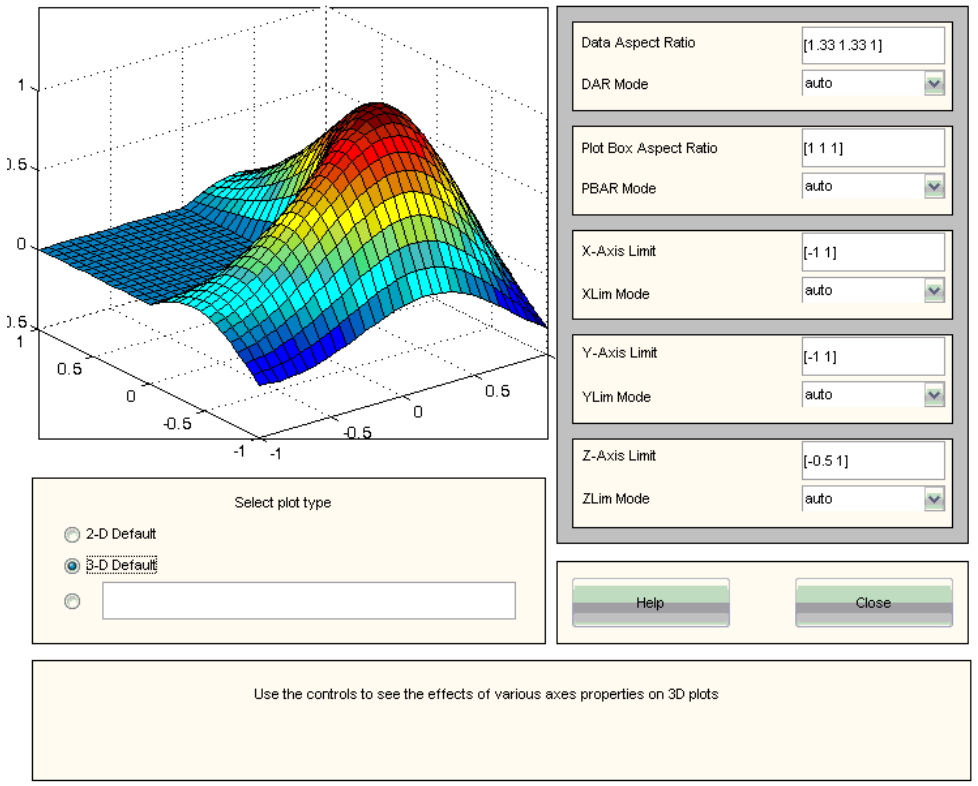
قسمت اول

- ✓ **فصل اول** — **MATLAB و ماتریس‌ها**
- ✓ **فصل دوم** — **ترسیم نمودارهای دوبعدی و سه‌بعدی**
- ✓ **فصل سوم** — **چند جمله‌ای‌ها و محاسبات سیمبولیک در MATLAB**
- ✓ **فصل چهارم** — **درون‌یابی و برازش منحنی‌ها**
- ✓ **فصل پنجم** — **کنترل جریان محاسبات**
- ✓ **فصل ششم** — **سیمولینک**
- ✓ **فصل هفتم** — **مدل‌سازی در SimMechanics**
- ✓ **فصل هشتم** — **مدل‌سازی و شبیه‌سازی سیستم‌های هیدرولیکی**
- ✓ **فصل نهم** — **مدل‌سازی سیستم‌های الکتریکی در SimPowerSystems**
- ✓ **فصل دهم** — **GUI (Graphical User Interface)**
- ✓ **فصل یازدهم** — **حل معادلات دیفرانسیل**



در این قسمت، به بررسی و توضیح جامع موضوعات کاربردی در نرم افزار MATLAB پرداخته شده است. این موضوعات عبارتند از: عملیات ماتریسی، داده‌های ورودی و خروجی، رسم نمودارهای دوبعدی و سه‌بعدی، درون‌یابی و میان‌یابی داده‌ها، پردازش سیمبولیک، استفاده از دستورات شرطی و حلقه‌های تکرار، شبیه‌سازی سیستم‌های مکانیکی، هیدرولیکی، و الکترونیکی در سیمولینک، رابط گرافیکی (GUI) و حل معادلات دیفرانسیل با مشتقات جزئی. با آموزش کامل این موضوعات، آمادگی کافی برای مواجه شدن با پروژه‌های سنگین و کاربردی را خواهید داشت.

فصل اول

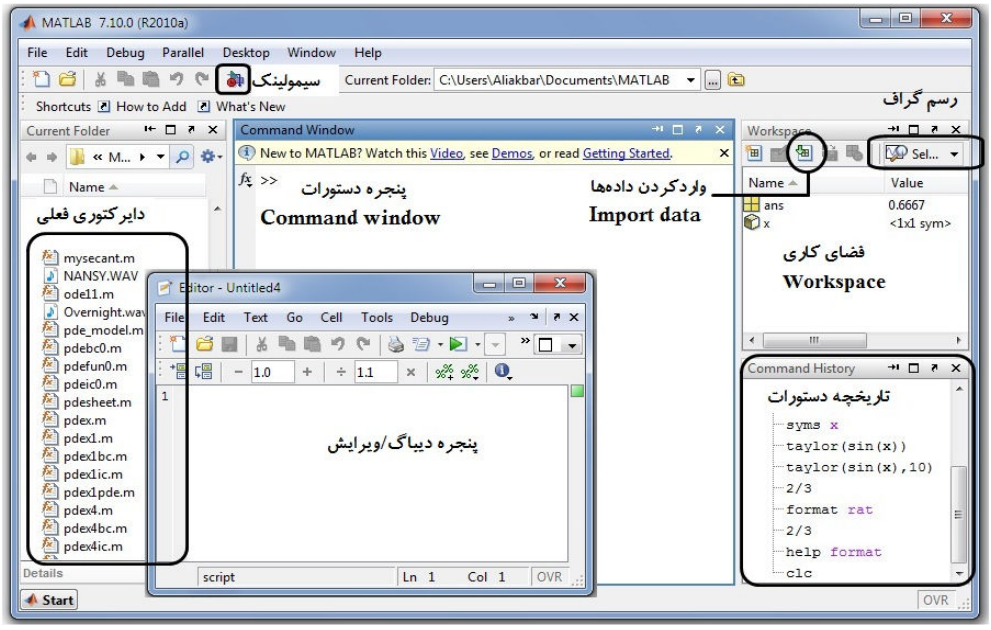
MATLAB و ماتریس‌ها

مقدمه

MATLAB، یک برنامه نرم افزاری قوی برای کسانی است که با محاسبات عددی و به‌ویژه جبرخطی سروکار دارند. نام این نرم افزار از عبارت انگلیسی MATrix LABoratory گرفته شده و هدف اولیه آن عبارت است از قادر ساختن مهندسان و دانشمندان به حل مسائل شامل عملیات ماتریسی، بدون نیاز به نوشتن برنامه در زبان های برنامه نویسی متداول همچون C. ماتریس‌ها، معادلات دیفرانسیل، رشته‌های عددی، ترسیمات و گراف‌ها از لوازم اصلی به‌کار رفته در ریاضیات و نیز در MATLAB هستند.

۱–۱ آشنایی با پنجره‌های MATLAB

هنگامی که برنامه MALAB را اجرا می‌کنید پنجره ای مطابق شکل (۱-۱) ظاهر می‌شود:



شکل (۱-۱) محیط‌کار MATLAB

پنجره دستورات، اصلی‌ترین پنجره MATLAB بوده و تمام دستورات و توابع MATLAB در این قسمت، در جلوی علامت >> درج می‌شوند.

هر بار که برنامه MATLAB را باز می‌کنید، زمان و تاریخ ورود، در پنجره Command History یا تاریخچه دستورات به رنگ سبز نشان داده می‌شود. اگر دستوراتی را در پنجره دستورات تایپ کنید، این دستورات پشت سرهم در این پنجره ذخیره می‌شوند. این دستورات حتی بعد از خارج شدن از برنامه نیز همچنان سرجای خود باقی مانده و حذف نمی‌شوند. با دوبار کلیک کردن روی این دستورات دوباره می‌توانید آن‌ها را اجرا کنید. از مسیر Edit>>Clear Command History می‌توانید محتویات پنجره تاریخچه دستورات را پاک کنید.

یکی دیگر از پنجره‌های مهم MATLAB، پنجره فضای کاری (Workspace) است. در این پنجره می‌توان تمام متغیرهای تعریف شده در MATLAB را به‌صورت فهرست وار مشاهده کرد. اگر در پنجره Workspace روی هر کدام از متغیرها دوبار کلیک کنید پنجره Array Editor مربوط به آن متغیر باز می‌شود. در این پنجره می توان هرگونه تغییری در مقادیر متغیرها ایجاد کرد. به کمک Workspace می‌توان ویرایش و نمودار متغیرها را به راحتی انجام داد. لازم به ذکر است که محتوای این پنجره را نیز می‌توان از منوی Edit پاک کرد.

اگر یکی از پنجره‌ها را به اشتباه ببندید، می‌توان از مسیر Desktop>>Desktop Layout>>Default، یک‌بار دیگر به صفحه با شکل و پنجره‌های اولیه بازگشت.

در قسمت Current Directory می‌توان M-fileهای موجود در مسیر جاری را بررسی کرد، به‌گونه‌ای که هرگونه خطایی در هر M-file یا پیشنهادی برای بهبود M-file در این پنجره ظاهر خواهد شد.

برای خروج از نرم‌افزار می‌توانید دستور quit یا exit را روبروی علامت << در پنجره دستورات تایپ کنید.

۱–۲ قوانین نام گذاری متغیرها در MATLAB

در این نرم افزار نمی‌توان از هر اسمی برای متغیرها و نام فایل‌ها استفاده کرد. قوانین زیر، در نام گذاری متغیرها و نام فایل‌ها حاکم است:

- متغیر باید یک کلمه بوده و نباید بین حروف آن فاصله‌ای وجود داشته باشد، یا نباید از نقطه بین حروف استفاده شود.
- این نرم‌افزار بین حروف بزرگ و کوچک تفاوت قائل می‌شود. به‌عنوان مثال دو متغیر A و a با هم تفاوت دارند.
- نام متغیرها باید با یک حرف آغاز شود: 'a-z'و'A-Z'.
- در نام‌گذاری‌ها از حروف 'a-z'، 'A-Z'، '0-9' و '_' می‌توان استفاده کرد. به‌عنوان مثال اسامی pay_day,r2d معتبر، اما اسامی a 2، Ali$ و pay-day غیر معتبر هستند.
- نام متغیر باید کمتر از ۳۱ کاراکتر باشد.
- کلمات کلیدی تعریف شده در نرم افزار همانند if ، for ، break ، return ، end ، for و catch و... را نمی‌توان به‌عنوان نام در نظر گرفت.

در ابتدا برای مشاهده عملکرد این نرم افزار، محاسبات ساده ریاضی زیر را انجام دهید:

```
>>2+3
ans =
     5
```

```
>>2^3
ans =
     8
```

```
>>1/0
ans =
   Inf
```

Inf نشان دهنده infinity یا بینهایت است.

در سطر بعد مقدار ۱۰ را به متغیر a و مقدار ۲ را به متغیر b نسبت دهید و در انتهای سطر دوم از سیمیکالن(;) استفاده کنید. سیمیکالن(;) باعث می‌شود مقدار b نمایش داده نشود. برای دیدن مقدار b ، متغیر b را در سطر بعدی تایپ کنید تا خروجی را مشاهده نمائید.

```
>>a=10
a =
    10
```

```
>>b=2;
```

```
>>b
b =
     2
```

برای قرار دادن حاصل جمع دو متغیر a و b در متغیر c دستور زیر را تایپ کنید:

```
>>c=a+b
c =
    12
```

در صورتیکه متغیری را قبلاً تعریف نکرده باشید، MATLAB یک پیغام خطا می‌دهد که اعلام می‌کند متغیر، تعریف نشده و ناشناس است. به‌طور مثال دستور زیر را تایپ کنید:

```
>>y=c+d
??? Undefined function or variable 'd'.
```

در این نرم افزار تمامی توابع ریاضی معمول همانند توابع مثلثاتی sin ، cos و توابع لگاریتمی مانند log ، log10 و غیره تعریف شده اند.

```
>>x=pi/6;
>>sin(x)
ans =
    0.5000
```

توجه کنید که متغیر x بر حسب رادیان است.

```
>>sin(90*pi/180)
ans =
     1
```

۱-۳ متغیرهای خاص

متغیرهای خاصی در MATLAB وجود دارند که برای موارد خاصی مورد استفاده قرار می‌گیرند به عنوان مثال:

ans : نام متغیر پیش فرض می‌باشد.

pi : نسبت محیط به قطر دایره $\pi=3.14$ است.

i,j: برابر $\sqrt{-1}$ می‌باشند.

eps : کوچکترین عددی که از اختلاف دو عدد حاصل شده و برابر 2.22e-16 است.

inf,Inf : نشان دهنده بی نهایت(infinity) است، به‌عنوان مثال 1/0

Nan,nan : عددی وجود ندارد به‌عنوان مثال 0/0

date : تاریخ فعلی کامپیوتر را به صورت رشته نمایش می‌دهد.

realmax : بزرگترین عدد موجود که برابر 1.79e+308 است. عدد بزرگتر از این عدد inf می‌باشد.

realmin : کوچکترین عدد موجود که برابر 2.22e-308 است. عدد کوچکتر از این عدد 0 مطلق می‌باشد.

۱-۴ آرایه‌ها و عملیات ریاضی روی آرایه‌ها

در MATLAB چهار نوع آرایه می‌توان تعریف کرد که عبارتند از:

۱- اعداد اسکالر که تک عضوی هستند.

۲- بردارها که شامل یک سطر یا یک ستون می‌باشند؛ به‌عبارتی یک بعدی هستند.

۳- ماتریس‌ها که از اعضای چیده شده در یک آرایش مربعی تشکیل می‌شوند.

۴- آرایه‌ها با ابعاد بیش از دو بعد.

اعضای یک آرایه می‌توانند عدد و یا حرف باشند. آرایه‌ها را به سادگی می‌توان ایجاد کرد. ساده‌ترین و ابتدائی ترین روش، تایپ تمام مقادیر بین دو براکت [ ] است. در این روش، اعداد با یک فاصله یا کاما (,) از یکدیگر جدا می‌شوند.

```
>> x = [2.5]
>> y = [0 1 2 3 4 5]
>> z = [0,1,2,3,4,5]
z =
     0     1     2     3     4     5
```

برای تولید آرایه‌های ستونی از سیمیکالن(;) استفاده می‌شود.

```
>> x = [0;1;2;3;4;5];
```

برای تبدیل یک آرایه سطری به ستونی یا بالعکس، از عملگر ترانهاده یا ترانزپوز (') استفاده می‌شود:

```
>> x = [0;1;2;3;4;5]'
x =
     0     1     2     3     4     5
>> transpose(x)
```

توجه : اگر در انتهای هر سطر دستور از سیمیکالن استفاده شود، سطر مفروض همانند قبل اجرا شده اما خروجی مشاهده نمی‌شود.

برای تولید آرایه‌های بزرگتر استفاده از روش‌های بالا وقت‌گیر است . بنابراین MATLAB چند دستور دیگر را معرفی می‌کند. این دستورات عبارتند از:

- Linspace(a,b,c): این دستور، c نقطه با فاصله‌های مساوی در بازه [a,b] تولید می‌کند. اگر c گفته نشده باشد، به صورت پیش فرض مقدار آن برابر ۱۰۰ خواهد بود.

```
>> x=linspace(0,pi,6)
x =
     0    0.6283    1.2566    1.8850    2.5133    3.1416
```

- Logspace(a,b,c): این دستور، c نقطه با فواصل لگاریتمی در بازه [a,b] تولید می‌کند. اگر c گفته نشده باشد، به صورت پیش فرض مقدار آن برابر ۵۰ خواهد بود.

```
>> x=logspace(0,4,5)
x =
     1      10         100      1000        10000
```

- دستور a:b:c، به این معناست که " از a شروع کن و به اندازه b جلو برو تا به c برسی". توجه کنید که b گام حرکت بوده و می‌تواند مثبت یا منفی باشد. اگر سخنی از b مطرح نشده باشد، به صورت پیش فرض مقدار آن برابر ۱ خواهد بود. علامت (:) کولن نام دارد.

```
>> x1= 1:2:13
x1 =
     1     3     5     7     9    11    13
>> x2 = 9:-1:1
x2 =
     9     8     7     6     5     4     3     2     1
>> x3=0.5:5
x3 =
    0.5000    1.5000    2.5000    3.5000    4.5000
```

برای دسترسی به درایه‌های یک آرایه از اندیس آن استفاده می‌کنیم. بطور مثال برای دسترسی به مولفه سوم بردارسطری x3 داریم:

```
>> x3(3)
ans =
    2.5000
>> x3(2)=[]    مولفه دوم را پاک می‌کند
```

برای دسترسی به چند مولفه از یک بردار از نوشتن اندیس‌ها بین دو براکت بهره می‌بریم. منظور از end، آخرین اندیس است.

```
>> x=[0.1 0.2 0.4 0.5 0.55 0.7 0.9];
>> y=sin(x)
y =
    0.0998    0.1987    0.3894    0.4794    0.5227    0.6442    0.7833
>> x([1 5 2])  برای دسترسی به مولفه اول،پنجم، و دوم
ans =
    0.1000    0.5500    0.2000
>> y([2:5])     برای دسترسی به مولفه دوم تا پنجم
ans =
    0.1987    0.3894    0.4794    0.5227
```

```
>> y(1:2:end)     برای دسترسی به مولفه اول تا آخر با گام دو
ans =
          0.9980.3894     0.5227     0.7833
```

۱–۵ روند حل مسائل مهندسی و علمی

اغلب مهندسین برای حل مسائل از یک روش و رویکرد ساختار یافته و مناسب استفاده می‌کنند. در این رویکرد از یک الگوریتم که به صورت گام به گام تعریف شده است، بهره می‌بریم تا به جواب مساله برسیم. برای حل مسائل مهندسی باید روند زیر را دنبال کرد:

۱- در ابتدا مساله را تعریف کرده و مسأله را به دقت مورد بررسی قرار دهید تا ببینید چه سوالی را باید پاسخ بدهید و چه خروجی‌ای باید حاصل شود. از طرفی چه دانش عملی و تئوری باید برای حل مسأله استفاده شود. اطلاعاتی از قبیل ورودی و خروجی را از مسأله استخراج کنید.

۲- در مرحله دوم باید مدل ریاضی حل مسأله ایجاد شود. سعی کنید برای فهم دقیق مسأله آن را به صورت بلوک نمودار رسم کنید. متغیرهای اساسی را تعریف کرده و نمادهای لازم را تعیین کنید. سعی کنید مسأله را به قدری ساده کنید که تنها کافی باشد که یکسری اطلاعات ورودی از کاربر گرفته شده و نتیجه حاصل به‌دست آید.

۳- در مرحله سوم، باید روش محاسباتی برای حل مسأله براساس مدل ریاضی، توسعه پیدا کند. یعنی باید مجموعه معادلاتی را که برای انجام محاسبات با متغیرهای تعریف شده مطلوب است استخراج کنید. سپس الگوریتم حل مسأله را در جمله‌های ریاضی تعریف کرده و سپس آن را برای نوشتن برنامه کامپیوتری آماده نمائید.

۴- در مرحله چهارم باید روش محاسباتی را اجرا کرده و برنامه را مطابق الگوریتم بنویسید.

۵- در مرحله آخر حل عددی مسأله باید به دقت تست شود تا از صحت برنامه نوشته شده مطمئن شوید.

چند دستور مفید دیگر

- با استفاده از دستور who می‌توان لیستی از تمامی متغیرهای موجود در فضای کاری را در پنجره دستورات نمایش داد.
- با استفاده از دستور whos می‌توان لیستی از نام، سایز، و بقیه مشخصات تمامی متغیرهای موجود در فضای‌کاری را در پنجره دستورات نمایش داد.
- با استفاده از دستور clc می‌توان صفحه پنجره دستورات را پاک کرد. دستور home نیز همین کار را انجام می‌دهد.
- با استفاده از دستور clear all می‌توان تمامی متغیرهای موجود در فضای کاری را پاک کرد.

- با استفاده از دستور clear x y میتوان فقط متغیرهای x,y را پاک کرد.
- دستور demo دموهای MATLAB را باز می کند .
- با دستور save می توانید مقادیر متغیرها را ذخیره کنید.
- با دستور load می توانید مقادیر متغیرهای ذخیره شده را لود کنید.
- با استفاده از دستور clock می توان زمان و تاریخ جاری کامپیوتر را در یک آرایه ذخیره کرد.

```
>>t=clock
t =
  1.0e+003 *
    2.0100   0.0040   0.0080   0.0220   0.0020   0.033
```

داده های فوق به صورت زیر می باشند.

t=[year month day minute seconds]

- دستور date تاریخ جاری را می‌دهد.

```
>>date
ans =
08-Apr-2010
```

- با استفاده دستور calendar می‌توان تقویم هر ماه را مشاهده کرد.

```
>> calendar
                  Apr 2010
     S     M     Tu     W     Th     F     S
     0     0      0     0      1     2     3
     4     5      6     7      8     9    10
    11    12     13    14     15    16    17
    18    19     20    21     22    23    24
    25    26     27    28     29    30     0
     0     0      0     0      0     0     0
>> calendar(2008,5)
```

- دستور pwd مسیر دایرکتوری جاری را به عنوان خروجی می‌دهد.
- دستورات dir و ls محتویات دایرکتوری جاری را نشان می‌دهند.
- با دستور copyfile می توان فایل ها را کپی کرد.

```
copyfile('source','destination')
```

- با دستور helpbrowser پنجره help نرم افزار MATLAB باز می‌شود.
- برای حذف خطوط اضافی موجود در پنجره دستورات از دستور format compact، و برای برگشتن به حالت قبل یعنی فاصله خالی بین سطرها، از دستور format loose استفاده می‌گردد.
- برای نمایش اعداد تا چهار رقم اعشار، از دستور format short، چهارده رقم اعشار، از دستور format long، چهار رقم اعشار اما با نماد علمی، از دستور format short e، و چهارده رقم اعشار به‌صورت نماد علمی، از دستور format long e استفاده می‌شود.

- دستور format short g اعداد را تا ۵ رقم بامعنا و format long g اعداد را تا ۱۵ رقم بامعنا نمایش می‌دهد. دستور format bank نیز اعداد را تا رقم اعشار نشان می‌دهد.
- دستور format rat برای نمایش کسری اعداد استفاده می‌شود.
- برای نمایش خروجی‌ها می‌توان از دستور disp استفاده کرد که به دو فرم زیر می‌باشد:

disp(variable) → a=2; disp(a)
disp(string)→ disp('what's your name: ')

۱-۶ توابع کاربردی در عملیات ماتریسی

تعدادی از توابعی‌که روی آرایه‌ها عمل می‌کنند در جدول (۱-۱) آمده است.

جدول (۱-۱) توابع کاربردی در عملیات ماتریسی

دستور	توضیحات	مثال
`[a,b]=sort(x)`	بردار x را به صورت صعودی منظم می‌کند. در آن، a بردار مرتب شده و b بردار موقعیت اعداد است.	x=[-1 6 -10 5 12 1]; [a,b]=sort(x)
`length(x)`	طول بردار x را بر می‌گرداند.	length(x)→ ans=6
`[d,n]=max(x)`	بزرگترین عدد را در یک آرایه پیدا می‌کند.n موقعیت عدد را مشخص می کند.	max(x) →ans=12
`[d,n]=min(x)`	کوچکترین عدد را در یک آرایه پیدا می‌کند. .n موقعیت عدد را مشخص می کند.	min(x)→ans= -10
`mean(x)`	میانگین بین اعداد یک آرایه را محاسبه می‌کند.	mean(x)→ ans =2.1667
`std(x)`	انحراف معیار بین اعداد یک آرایه را محاسبه می‌کند.	std(x)→ ans =7.4677
`sum(x)`	حاصل جمع مولفه‌های x را محاسبه می‌کند.	sum(x)→ ans=13
`prod(x)`	حاصل‌ضرب مولفه‌های x را محاسبه می‌کند.	prod(x)→ans= 3600
`cumsum(x)`	حاصل جمع مولفه‌های x را از اول تا هرمولفه محاسبه می‌کند.	cumsum(x)
`cumprod(x)`	حاصل‌ضرب مولفه‌های x را از اول تا هرمولفه محاسبه می‌کند.	cumprod(x)
`[m,n]=size(x)`	سایز بردار x را برمی‌گرداند.	[m,n]=size(x)→ans= 1 6

یادآوری: معادل ریاضی دستورات prod ، cumprod، sum، و cumsum در زیر آمده است.

$$\text{Sum y:}\quad y=\sum_{n=1}^{N}x(n)=x(1)+x(2)+\ldots+x(N)$$

$$\text{Product y:}\quad y=\prod_{n=1}^{N}x(n)=x(1)\times x(2)\ldots\times x(N)$$

Cumulative Sum y: $y(k)=\sum_{n=1}^{k} x(n)=x(1)+x(2)+...+x(k)$

Cumulative product y: $y(k)=\prod_{n=1}^{k} x(n)=x(1)\times x(2)...\times x(k)$

با ترکیب آرایه‌های سطری و ستونی می‌توان آرایه‌هایی با سطرها و ستون‌های بیشتر تولید کرد. البته، در ترکیب سطری [a,b] باید تعداد سطرها، و در ترکیب ستونی [a;b] باید تعداد ستون‌ها برابر باشند.

```
>> a=[1 2 3];
>> b=[4 5 6];
>>c=[7 8 9];
>>d=[a b],f=[a;b;c]
d =
     1     2     3     4     5     6
f =
     1     2     3
     4     5     6
     7     8     9
```

راه دیگر برای نوشتن ماتریس‌ها استفاده از کولن و سیمیکالن(;) می‌باشد. هنگام نوشتن توجه داشته باشید که باید در هر سطر تعداد ستون‌ها برابر باشد در غیر این صورت پیغام خطا ظاهر می‌شود. از طرفی MATLAB فاصله‌های اضافی را متوجه شده و حذف می‌کند.

```
>> a=[1,2,3;4,5,  6;7,8,9]
a =
     1     2     3
     4     5     6
     7     8     9
```

اگر تعداد سطرها و ستون‌های آرایه‌ها با هم یکسان باشد می‌توان روی آرایه‌ها عملیات ریاضی جمع ، تفریق، ضرب و تقسیم را به صورت مولفه به مولفه انجام داد.

در هنگام اجرای محاسبات ریاضی اولویت با پرانتز می‌باشد. در اولویت‌های بعدی به ترتیب توان(^)، ضرب(*) و تقسیم(/) (اولویت از چپ به راست در عبارت)، جمع(+) و تفریق(-) (اولویت از چپ به راست در عبارت) قرار دارند.

جدول(۱-۲) فرم جبری و MATLABی عملیات ساده ریاضی

فرم جبری	MATLAB
$a+b$	a+b
$a-b$	a-b
$a\times b$	a.*b
$a\div b$	a./b
a^b	a.^b

```
>> a = [2 4 8];
```

```
>> b = [3 2 2];
>> [a(1)*b(1) a(2)*b(2) a(3)*b(3)]
ans =
     6     8    16
>> a+ b
ans =
     5     6    10
>> a .* b
ans =
     6     8    16
>> a ./ b
ans =
    0.6667    2.0000    4.0000
>> a = [2 4 8;-2 -2 8];
>> b = [3 2 2;-3  5 7];
>> C= a./b
C =
    0.6667    2.0000    4.0000
    0.6667   -0.4000    1.1429
>> C=a.*b  در این حالت مولفه‌های نظیر به نظیر در هم ضرب می‌شوند و با ضرب ماتریسی متفاوت است
C =
     6     8    16
     6   -10    56
>> 2*C
ans =
    12    16    32
    12   -20   112
>> 2.*C
ans =
    12    16    32
    12   -20   112
>> C.^2     در این حالت تک تک مولفه‌ها به توان۲ می‌رسند
ans =
          36          64         256
          36         100        3136
>> 2.^a  در این حالت عدد ۲ به توان تک تک مولفه می‌رسد
ans =
    4.0000   16.0000  256.0000
    0.2500    0.2500  256.0000
>> tan(a)
ans =
   -2.1850    1.1578   -6.7997
    2.1850    2.1850   -6.7997
>> [2 3 4] .^[4 3 1]
ans =
    16    27     4
```

برای دستیابی به مولفه‌های یک ماتریس، به صورت زیر عمل می‌کنیم:

```
>> a = [2 4 8;-2 -2 8;4 6 -7];
>> a(2,3)
ans =
     8
>> a(2,:)  سطر دوم ماتریس
ans =
    -2    -2     8
>> a(:,2) ستون دوم ماتریس
ans =
     4
    -2
     6
>> a(:)'      مولفه‌ها را به صورت ستونی پشت سرهم قرار داده، سپس ترانهاده آن را حساب می‌کند
ans =
     2    -2     4     4    -2     6     8     8    -7
>> a(1:2,2:3)
ans =
     4     8
    -2     8
>> a(2,:)=[]    این دستور سطر دوم را پاک می‌کند
```

ضرب و توان ماتریس‌ها نیز به راحتی قابل تعریف است. در ضرب ماتریس‌ها، تعداد ستون‌های ماتریس اول باید برابر تعداد سطرهای ماتریس دوم باشد. در توان رسانی ماتریس‌ها نیز باید ماتریس‌ها مربعی باشند.

```
>> a=[1 2 -3;4 5 7];
>> b=[2 1 3;4 -3 -5;0 3 2];
>> a*b
ans =
    10   -14   -13
    28    10     1
>> a^2
??? Error using ==> mpower
Matrix must be square.

>> b^2
ans =
     8     8     7
    -4    -2    17
    12    -3   -11
```

برای محاسبه تعداد سطرها و ستونهای یک ماتریس از دستور size استفاده می‌شود.

```
>> [r,c]=size(a)
r =
     2
c =
     3
```

```
>> r = size(a,1)
r =
     2
>> c = size(a,2)
c =
     3
```

توابع ریاضی ساده: در جدول (۱- ۳)، فهرستی از توابع ساده ریاضی آمده است.

جدول (۱- ۳) توابع ساده ریاضی

دستور	توضیحات	مثال
abs(x)	قدر مطلق آرایه x را محاسبه می‌کند.	abs(-2.5)
sign(x)	تابع علامت	sign(-3.5)
exp(x)	تابع نمایی	exp(x.^2/2)
expm(A)	تابع نمایی ماتریس	expm([1 0;0 1])
log(x)	لگاریتم طبیعی آرایه x را محاسبه می‌کند.	log(6)
log10(x)	لگاریتم در مبنای ۱۰ آرایه x را محاسبه می‌کند.	log10(x)
log2(x)	لگاریتم در مبنای ۲ آرایه x را محاسبه می‌کند.	log2(x)
sqrt(x)	جذر آرایه x را محاسبه می‌کند.	sqrt(2)/2
ceil(x)	عدد را به سمت بالا، به نزدیکترین عدد صحیح گرد می‌کند.	ceil(-3.5)
floor(x)	عدد را به سمت پائین، به نزدیکترین عدد صحیح گرد می‌کند.	floor(-3.5)
pow2(x)	توان x عدد ۲ را محاسبه می‌کند.	pow2(5)
fix(x)	قسمت صحیح عدد x را برمی‌گرداند.	fix(sqrt(2))
rem(x,y)	باقیمانده عدد x بر y را به‌عنوان خروجی نمایش می‌دهد.	rem(10,4)

نکته: در انجام محاسبات ریاضی اولویتِ انجام محاسبات با پرانتز است؛ یعنی ابتدا محاسبات داخل پرانتز انجام می‌شود. بعد از پرانتز، به ترتیب اولویت با توان، ضرب وتقسیم (از چپ به راست)، و جمع و تفریق (از چپ به راست) است.

مجموعه‌ای از آرایه‌های استاندارد در MATLAB تعریف شده اند که گروهی از آنها در جدول (۱-۴) آورده شده است.

جدول(۱- ۴) آرایه‌های استاندارد در MATLAB

دستور	توضیحات	مثال
ones(n,m)	ماتریس n در m با درایه‌های یک تولید می‌کند.	ones(2,4)
zeros(n,m)	ماتریس n در m با درایه‌های صفر تولید می‌کند.	zeros(3,4)

ones(3)	ماتریس n در n با درایه‌های یک تولید می‌کند.	ones(n)
zeros(3)	ماتریس n در n با درایه‌های صفر تولید می‌کند.	zeros(n)
eye(2)	ماتریس n در n با درایه‌های قطری یک تولید می‌کند.	eye(n)
rand(4,5)	ماتریس n در m با درایه‌های رندوم بین ۰-۱ تولید می‌کند.	rand(n,m)
rand(3)	ماتریس n در n با درایه‌های رندوم بین ۰-۱ تولید می‌کند.	rand(n)
randn(30);	ماتریس n در n با درایه‌های رندوم بین ۰-۱ با توزیع گوسین تولید می‌کند.	randn(n)
randn(1,50);	ماتریس n در m با درایه‌های رندوم بین ۰-۱ با توزیع گوسین تولید می‌کند.	randn(n,m)
randperm(10)	اعدا ۱ تا n را به صورت اتفاقی در یک بردار سطری قرار می‌دهد.	randperm(n)
magic(3)	ماتریسی است که درایه‌های هر سطر، ستون، و قطر با هم برابرند.	magic(n)

۱-۷ ضرب داخلی و خارجی

برای ضرب داخلی و خارجی بردارها به ترتیب از دستورات dot و cross استفاده می‌شود.

```
>> x = [4 -1 3]
>> y = [-2 5 2]
>> inner= dot(x,y)
>> Outer= cross(x,y)
```

برای محاسبه اندازه یک بردار از دستور norm استفاده می‌شود.

$$\|x\| = \sqrt{x_1^2 + x_2^2 + ... + x_n^2}$$

```
>> nx = norm(x)
```

برای محاسبه بردار یکه ux، دستورات زیر را وارد کنید.

```
>> ux = x/norm(x)
>> norm(ux)
ans = 1
```

برای محاسبه زاویه بین دو بردار به صورت زیر عمل کنید.

```
>> theta = acos(dot(x,y)/(norm(x)*norm(y)))
theta =1.8121
>> thetad = (180/pi)*theta
thetad = 103.8261
```

برای محاسبه تصویر یک بردار بر بردار دیگر دستور زیر را وارد کنید.

```
>> z = (dot(x,y)/norm(y)^2)*y
```

در جدول (۱-۵) گروهی از توابع مورد استفاده در ماتریس‌ها آورده شده است.

جدول(۱- ۵) توابع کاربردی مورد استفاده در ماتریس‌ها

دستور	توضیحات	مثال
diag(a)	اگر a ، یک ماتریس باشد، عناصر روی قطر اصلی را بر می‌گرداند و اگر یک بردار باشد، ماتریسی تولید می‌کند که عناصر روی قطر اصلی آن بردار و بقیه عناصر صفر هستند.	diag([-2 5;7 8]) diag([-2 5 7 8])
trace(a)	حاصل جمع عناصر روی قطر اصلی را محاسبه می‌کند.	trace([-2 5;7 8])
minfo(a)	اطلاعاتی درباره ماتریس مورد نظر برمی‌گرداند.	minfo([-2 5;7 8])
flipud(a)	جای سطرها را نسبت به سطر وسط عوض می‌کند.	flipud([1 2 3;4,5,6;7 8 9])
fliplr(a)	جای ستونها را نسبت به ستون وسط عوض می‌کند.	fliplr([1 2 3;4,5,6;7 8 9])
rot90(a)	ماتریس را به اندازهٔ ۹۰ درجه در جهت پادساعتگرد می‌چرخاند.	rot90([1 2 3;4,5,6;7 8 9])
tril(a)	ماتریس پائین مثلثی تولید می‌کند.	tril([1 2 3;4,5,6;7 8 9])
triu(a)	ماتریس بالا مثلثی تولید می‌کند.	triu([1 2 3;4,5,6;7 8 9])
repmat(a,m,n)	ماتریسa را به تعداد m در n تکرار می‌کند.	repmat([2 3],2,5)
cat(n,a,b)	دو ماتریس a,b را ملحق می‌کند. n نشان دهنده جهت الحاق است.	cat(1,[1 2;3 4],[3 5;8 9])
inv(a)	ماتریس وارون را محاسبه می‌کند.	inv([2 -3;1 -4])
det(a)	دترمینان ماتریس مربعی را محاسبه می‌کند.	det([2 -3;1 -4])
rank(a)	مرتبه یک ماتریس را محاسبه می‌کند.	rank([2 -3;1 -4])
eig(a)	مقادیر و بردارهای ویژه ماتریس مربعی را محاسبه می‌کند.	[E,V]=eig([2 -3;1 -4])
poly(a)	چند جمله‌ای مشخصه ماتریس a را تولید می‌کند.	Poly([2 -3;1 -4])

مثال: دستگاه معادلات زیر را حل کرده، سپس برای اطمینان به درستی محاسبات، پاسخ آن را بررسی کنید.

$$\begin{cases} 3x_1 + 2x_2 - x_3 = 10 \\ -x_1 + 3x_2 + 2x_3 = 5 \\ x_1 - x_2 - x_3 = -1 \end{cases}$$

برای حل دستگاه، ابتدا ماتریس ضرایب و پاسخ‌ها را جدا کرده، هر یک را در یک متغیر قرار می‌دهیم:

$$\begin{pmatrix} 3 & 2 & -1 \\ -1 & 3 & 2 \\ 1 & -1 & -1 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}=\begin{pmatrix} 10 \\ 5 \\ -1 \end{pmatrix}$$

$$A=\begin{pmatrix} 3 & 2 & -1 \\ -1 & 3 & 2 \\ 1 & -1 & -1 \end{pmatrix} \quad X=\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \quad b=\begin{pmatrix} 10 \\ 5 \\ -1 \end{pmatrix}$$

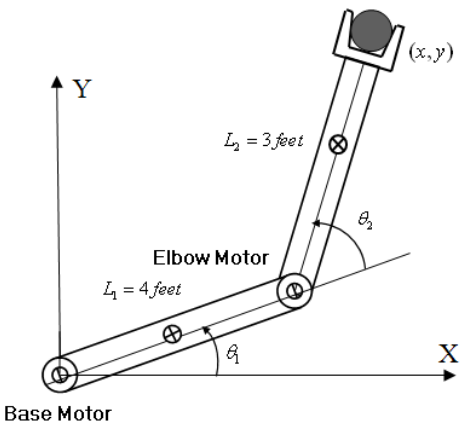
```
>> A = [3 2 -1; -1 3 2; 1 -1 -1]
>> b= [10; 5; -1]
```

سپس با معادله $X = A^{-1}b$ ، پاسخ‌های X را به‌دست می‌آوریم:

```
>> x = inv(A)*b
x =
-2.0000
 5.0000
-6.000
>> check=A*x
ans =
  10.0000
  5.0000
 -1.0000
```

مثال: روبات دولینکی زیر را در نظر بگیرید. در هر یک از مفاصل، یک موتور قرار گرفته و طول و زاویه‌های هر یک از لینک‌ها به ترتیب برابر $(\theta_1, L_1), (\theta_2, L_2)$ است. می‌دانیم مختصات (x,y) مربوط به پنجه ربات به صورت زیر می‌باشد:

$$x = L_1.Cos(\theta_1) + L_2 Cos(\theta_1 + \theta_2)$$
$$y = L_1.Sin(\theta_1) + L_2 Sin(\theta_1 + \theta_2)$$



شکل (۱-۲) روبات دولینکی با مفاصل صلب

اکنون مسأله این است که چگونه می‌توان با استفاده از موتورها ربات را به‌گونه‌ای کنترل کرد که ربات در یک مسیر مشخص از یک نقطه مفروض در صفحه به نقطه‌ای دیگر حرکت کند. در این مسیر حرکتی، ربات با سرعت

و شتاب صفر شروع به حرکت می‌کند و با همین سرعت و شتاب متوقف می‌شود($t_f = 2\sec$). مقادیر اولیه و پایانی زوایه‌های ربات و طول لینک‌ها و زمان پایانی حرکت مطابق جدول (۱-۶) می‌باشد.

جدول (۱- ۶) زوایای ربات و طول لینک‌ها

$L_1 = 4\,feet$	$L_2 = 3\,feet$
$\theta_1(0) = -19^0$	$\theta_2(0) = 44^0$
$\theta_1(t_f) = 43^0$	$\theta_2(t_f) = 151^0$

برای کنترل حرکت ربات چند جمله‌ای مرتبه پنج زیر را برای هریک از مفاصل در نظر می‌گیریم:

$$\theta_1(t) = a_1t^5 + a_2t^4 + a_3t^3 + a_4t^2 + a_5t + \theta_1(0)$$

$$\theta_2(t) = b_1t^5 + b_2t^4 + b_3t^3 + b_4t^2 + b_5t + \theta_2(0)$$

از آنجایی که می‌دانیم که سرعت وشتاب زاویه‌ای در ابتدا و انتهای مسیر برای هر یک از لینک‌ها برابر صفر است، بنابراین از هر یک از روابط بالا مشتق می‌گیریم تا سرعت زاویه‌ای هر یک از لینک‌ها به‌دست آید:

$$\theta_1'(0) = 5a_1t^4 + 4a_2t^3 + 3a_3t^2 + 2a_4t + a_5 = 0 \quad ,\theta_1'(0) = a_5 = 0$$

$$\theta_2'(0) = 5b_1t^4 + 4b_2t^3 + 3b_3t^2 + 2b_4t + b_5 = 0 \quad ,\theta_2'(0) = b_5 = 0$$

مشتق سرعت زاویه‌ای برابر شتاب زاویه‌ای است، بنابراین:

$$\theta_1''(0) = 20a_1t^3 + 12a_2t^2 + 6a_3t + 2a_4 = 0 \ ,\theta_1''(0) = a_4 = 0$$

$$\theta_2''(0) = 20b_1t^3 + 12b_2t^2 + 6b_3t + 2b_4 = 0 \ ,\theta_2''(0) = b_4 = 0$$

برای زاویهٔ $\theta_1(t)$ معادله‌های زیر را خواهیم داشت:

$$\begin{cases} \theta_1(t) = a_1t^5 + a_2t^4 + a_3t^3 + \theta_1(0) \\ \theta_1'(t) = 5a_1t^4 + 4a_2t^3 + 3a_3t^2 \\ \theta_1''(t) = 20a_1t^3 + 12a_2t^2 + 6a_3t \end{cases}$$

این رابطه را می‌توان به صورت ماتریسی نوشت:

$$\begin{pmatrix} t^5 & t^4 & t^3 \\ 5t^4 & 4t^3 & 3t^2 \\ 20t^3 & 12t^2 & 6t \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} = \begin{pmatrix} \theta_1(t) - \theta_1(0) \\ 0 \\ 0 \end{pmatrix}$$

مراحل بالا را می‌توان برای زاویه $\theta_2(t)$ نیز تکرار کرد و به روابط مشابهی دست یافت. برای لحظه پایانی t_f می توان روابط زیر را نوشت:

$$\begin{pmatrix} t_f^5 & t_f^4 & t_f^3 \\ 5t_f^4 & 4t_f^3 & 3t_f^2 \\ 20t_f^3 & 12t_f^2 & 6t_f \end{pmatrix} \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix} = \begin{pmatrix} \theta_2(t_f) - \theta_2(0) \\ 0 \\ 0 \end{pmatrix}$$

$$\begin{pmatrix} t_f^5 & t_f^4 & t_f^3 \\ 5t_f^4 & 4t_f^3 & 3t_f^2 \\ 20t_f^3 & 12t_f^2 & 6t_f \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} = \begin{pmatrix} \theta_1(t_f) - \theta_1(0) \\ 0 \\ 0 \end{pmatrix}$$

و

با جایگذاری شرایط اولیه ارائه شده در جدول (۱- ۶)، به‌راحتی می‌توان با رابطه $X = A^{-1}b$ پارامترهای ثابت را به‌دست آورد.

برنامه حرکت ربات در مسیر مطلوب از نقطه $(6.5,0)$ تا نقطه $(0,2)$ عبارت است از:

```
% Initial values, angles in degrees 2 radian
tf = 2;
theta10 = -19*pi/180;
theta1tf = 43*pi/180;
theta20 = 44*pi/180;
theta2tf = 151*pi/180;
% Equations for a coefficients
T = [ tf^5          tf^4         tf^3
      5*tf^4       4*tf^3        3*tf^2
      20*tf^3      12*tf^2       6*tf ];
c = [ theta1tf-theta10; 0; 0 ];
disp('Coefficients for theta1 motion:')
```

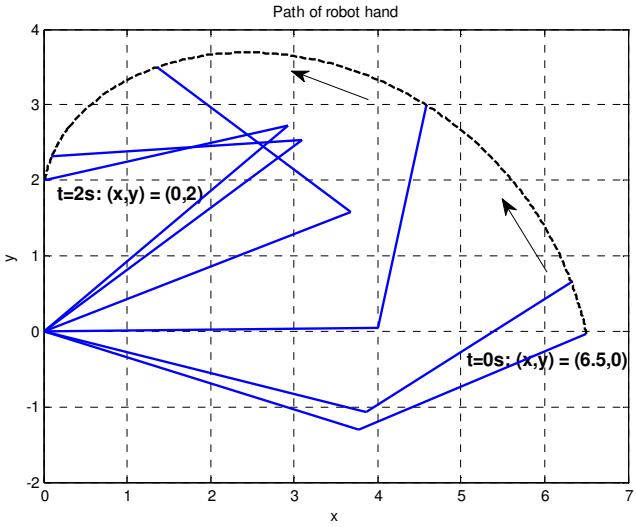
برای نمایش خروجی‌ها ازاین دستور استفاده می‌شود

```
a = inv(T)*c
% Equations for bcoefficients
d = [ theta2tf-theta20; 0; 0 ];
disp('Coefficients for theta2 motion:')
b= inv(T)*d
% Equations of motion
L1 = 4;
L2 = 3;
t = linspace(0,2,401);
tq = [ t.^5; t.^4; t.^3 ];
theta1 = theta10 + a'*tq;
theta2 = theta20 + b'*tq;
x = L1*cos(theta1) + L2*cos(theta1 + theta2);
y = L1*sin(theta1) + L2*sin(theta1 + theta2);
% Plot path of hand
plot(x,y)
xlabel('x')
ylabel('y')
title('Path of robot hand')
text(4.3,0,'t=0s: (x,y) = (6.5,0)')
text(0.2,2,'t=2s: (x,y) = (0,2)')
grid on
x1 = L1*cos(theta1);
```

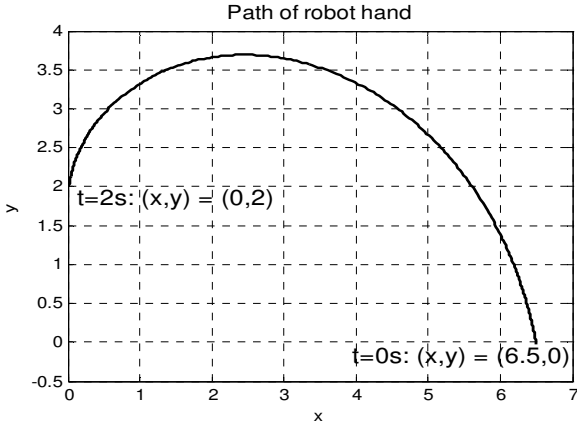
```
y1 = L1*sin(theta1);
hold on
for i=1:80:length(t)
    plot([0 x1(i)],[0 y1(i)])
    plot([x1(i),x(i)],[y1(i) y(i)])
end
```

توجه: برای نوشتن توضیحات روبروی دستورات و خوانا شدن بیشتر برنامه، قبل از این توضیحات از علامت درصد (%) استفاده می‌شود. این توضیحات هنگام اجرای برنامه در نظر گرفته نمی‌شوند.

مسیر حرکت ربات و شش موقعیت اختیاری لینک‌ها در شکل‌های زیر نشان داده شده است. (دستور for در بخش‌های بعدی توضیح داده می‌شود.)



شکل (۱-۳) مسیر حرکت ربات به‌همراه موقعیت بازوها



شکل(۱-۴) مسیر حرکت ربات

۱-۸ توابع مثلثاتی

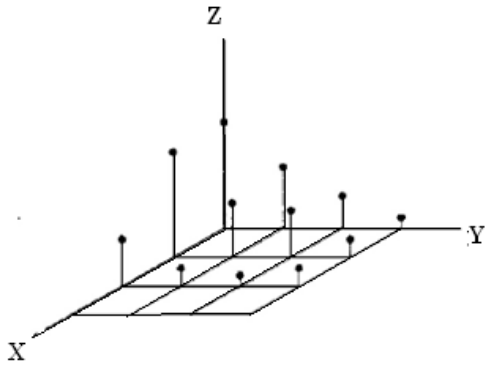
توابع مثلثاتی موجود در MATLAB عبارتند از:

جدول (۱-۷) توابع مثلثاتی موجود در MATLAB

دستور	توضیحات	مثال
sin(alpha)	سینوس x ، x بر حسب رادیان می‌باشد.	alpha=pi/3; sin(alpha)
cos(alpha)	کسینوس x، x بر حسب رادیان می‌باشد.	cos(alpha)
tan(alpha)	تانژانت x، x بر حسب رادیان می‌باشد.	tan(alpha)
asin(x)	آرک سینوس x، که x بین [1 1-] و خروجی بین $[\frac{-\pi}{2} \quad \frac{\pi}{2}]$ است.	a=0.5; asin(a)
acos(x)	آرک کسینوس x، که x بین [1 1-] و خروجی بین $[0 \quad \pi]$ است.	acos(a) *180/pi
atan(x)	آرک تانژانت x ،که x بین $[\infty \quad -\infty]$ و خروجی بین $[\frac{-\pi}{2} \quad \frac{\pi}{2}]$است.	atan(a) *180/pi
atan2(y,x)	آرک تانژانت که x,y مختصات نقطه در صفحه می‌باشند.	atan2(1,2)*180/pi
sinh(x)	سینوس هایپربولیک x، $\sinh(x)=\frac{1}{2}(e^{x}-e^{-x})$	sinh(0.5)
cosh(x)	کسینوس هایپربولیک x، $\cosh(x)=\frac{1}{2}(e^{x}+e^{-x})$	cosh(0.5)
tanh(x)	تانژانت هایپربولیک x، $\tanh(x)=\frac{\sinh(x)}{\cosh(x)}$	tanh(0.5)
asinh(x)	معکوس سینوس هایپربولیک x، $a\sinh(x)=\ln(x+\sqrt{x^2+1})$	asinh(0.5211)
acosh(x)	معکوس کسینوس هایپربولیک x ، $a\cosh(x)=\ln(x-\sqrt{x^2+1}), x\geq 1$	acosh(1.1276)
atanh(x)	معکوس تانژانت هایپربولیک x، $a\tanh(x)=\frac{1}{2}\ln(\frac{1+x}{1-x}), \lvert x\rvert \leq 1$	atanh(0.5)

یکی از دستورات کاربردی و مهم، دستور meshgrid است. این دستور روی صفحه XY یک شبکه دوبعدی ایجاد می‌کند. بردارهای ورودی به این دستور مشخص کننده تقسیمات در جهت‌های x و y هستند. با استفاده از این دستور می‌توان ماتریسهای x و y برای رسم نمودارهای سه بعدی را ایجاد کرد.

```
>> x=[1 2 3 4];
>> y=[-1 -2 -3 -4 -5];
```

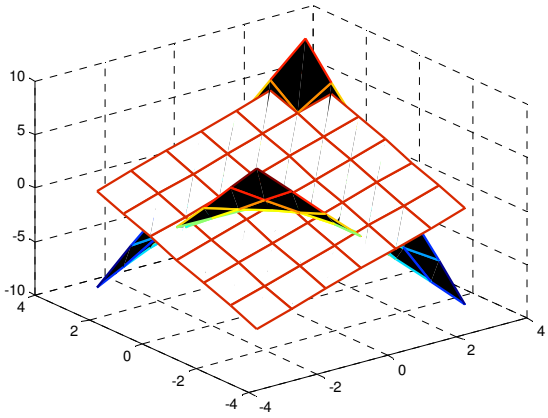


شکل(۱-۵) شبکه دو بعدی در صفحه XY

```
>> [u,v]=meshgrid(x,y)
u =
     1     2     3     4
     1     2     3     4
     1     2     3     4
     1     2     3     4
     1     2     3     4
v =
    -1    -1    -1    -1
    -2    -2    -2    -2
    -3    -3    -3    -3
    -4    -4    -4    -4
    -5    -5    -5    -5
```

سایز ماتریس‌های u وv برابر بوده و برابر حاصل‌ضرب طول بردار y در x است. اگرx وy یکی باشند می‌توان یکی از آنها را نوشت. بطور مثال برای رسم دو صفحه z1=x+y و z2=xy دستورات زیر را وارد می‌کنیم:

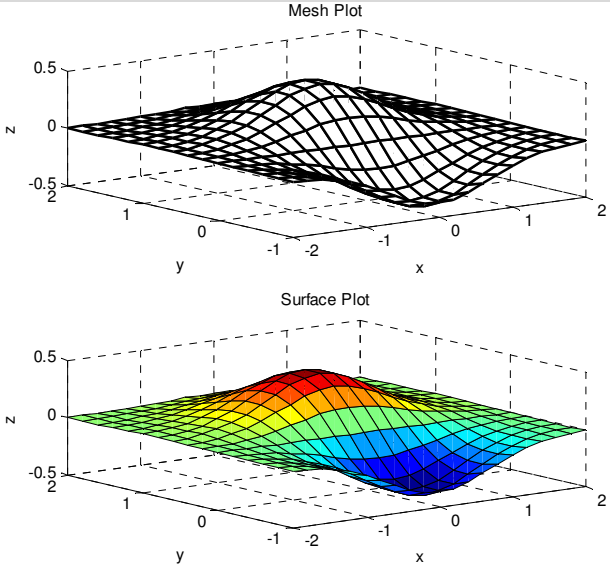
```
>>s=-3:3;
>>[x,y]=meshgrid(s)
>>z1=x+y ;
>>z2=x.*y;
>>mesh(x,y,z1)
>>hold on    دو نمودار را روی هم می‌اندازد
>>mesh(x,y,z2)  برای رسم سطوح سه بعدی از این دستور استفاده می‌شود
```



شکل(۱-۶) صفحات **z1=x+y** و z2=xy

مثال: نمودار تابع دو متغیره $f(x,y)=ye^{-(x^2+y^2)}$ را رسم کنید.

```
>> x = -2:0.2:2;
>> y = -1:0.2:2;
>> [X,Y] = meshgrid(x,y);
>> Z = Y.*exp(-(X.^2 + Y.^2));
>> figure(1)     شماره شکل را مشخص می‌کند
>> subplot(2,1,1),mesh(X,Y,Z)
>> title('Mesh Plot'),xlabel('x'),ylabel('y'),zlabel('z')
>> subplot(2,1,2),surf(X,Y,Z)
>> title('Surface Plot'),xlabel('x'),ylabel('y'),zlabel('z')
```



$f(x,y)=ye^{-(x^2+y^2)}$ شکل(۱-۷) تابع دو متغیره

۱–۹ داده‌های ورودی و خروجی

۱–۹–۱ انواع داده‌ها: داده‌هایی که در MATLAB با آنها سروکار داریم مختلف هستند. در جدول (۱–۸) چند نوع از این داده‌ها توضیح داده شده است.

جدول(۱–۸) انواع داده‌های مورد استفاده در MATLAB

نوع داده	توضیحات	مثال
Logical	آرایه‌هایی که مولفه‌های آن (True)1 و یا (False) 0 می‌باشند.	`A > 4`
Char	آرایه‌های کاراکتری و رشته‌ای	`'Hello'`
Cell	آرایه‌هایی سلولی که مولفه‌های آن آرایه‌های دیگر را می‌تواند شامل شود.	`{17 'Hello' eye(2)}`

Structure	آرایه‌های ساختمانی	`a.day=12;`
		`a.color = 'Red'`
		`a.mat = magic(3)`

Function Handle		`@humps`
User class	این کلاس که توسط کاربر تعریف می‌شود، با استفاده از توابع MATLAB ایجاد می‌گردد.	`Inline('sin(x)')`
Java class		`Java.awt.Frame`
Single	آرایه‌های عددی با دقت Single که فضای کمتری نسبت به داده‌های Double نیاز دارند.	`3*10^8`
Double	آرایه‌های عددی با دقت Double که مرسوم ترین نوع متغیر در MATLAB هستند.	`3*10^308` `5+6i`
Int8,16,32,64	آرایه‌ای از اعداد صحیح با علامت که طول آنها ۸،۱۶،۳۲،۶۴ بیت است.	`Int16(100)`
Unit8,16,32,64	آرایه‌ای از اعداد صحیح بدون علامت که طول آنها ۸،۱۶،۳۲،۶۴ بیت است.	`Unit16(100)`

۱–۹–۲ خروجی‌های رشته‌ای: خروجی‌های رشته‌ای مجموعه‌ای از کاراکترهای الفبایی یا عددی هستند که در بین یک جفت آپوستروف ’,‘ قرار می‌گیرند. هر کاراکتر رشته یک عضو در یک بردار را اشغال می کند. یک رشته در MATLAB از نوع char است و هر کاراکتر در دو بایت از حافظه ذخیره می‌شود.

```
>> Text='my name is ali';
```

```
>> U =Text(4:10)
U =
name is
>> m='abcd 1234';
>> size(m)
ans =
     1      9
>> m(1)
ans = a
>> m(2:6)
ans = bcd 1
>> length(m)
ans = 9
>> findstr(m,'2')
ans = 7
```

دستور findstr درون یک رشته را برای یک کلمه جستجو کرده و موقعیت را مشخص می‌کند.

```
findstr('I am working in the field of mechatronic','field')
ans =
    21
```

دستور ischar برای چک کردن آرایه‌های کاراکتری به‌کار می‌رود. اگر متغیر ورودی یک کاراکتر باشد، این تابع یک و در غیر این صورت صفر را بر می‌گرداند.

متغیرها با استفاده از دستور double از نوع char به double تبدیل می‌شوند.

```
a='MATLAB2010'
a =
MATLAB2010
b=double(a)
b =
    77    65    84    76    65    66    50    48    49    48
```

دستور char عکس عمل بالا را انجام داده و متغیرها را از نوع double به نوع char تبدیل می‌کند.

```
char(b)
ans =
MATLAB2010
```

۱-۹-۳ وارد کردن داده‌های اسکالر و برداری با استفاده از دستور input: برای وارد کردن یک کمیت عددی، برداری، رشته‌ای و یا ماتریسی می‌توان از دستورInput استفاده کرد.

```
>> x=input('your name =')
your name ='aliakbar'
>> x =aliakbar;
>> y=x(1:3)
y = ali
```

۱-۹-۴ مقایسه رشته‌ها: برای مقایسه رشته ها می‌توان از دستورات متعددی استفاده کرد. دستور strcmp بررسی می‌کند که آیا دو رشته با هم برابرند یا خیر؟ این دستور به حروف بزرگ و کوچک حساس است.

```
>> str1='hello';
```

```
>> str2='Hello';
>> str3='help';
>> strcmp(str1,str2)
ans=0
```

در حالی که دستور strcmpi رشته‌ها را بدون در نظر گرفتن بزرگی و کوچکی حروف برابری آنها، مقایسه می کند.

```
>> strcmp(str1,str2)
ans=1
```

روش دیگر استفاده از عملگرهای مقایسه است.

```
>> S1='ann';
>> S2='ban';
>> S1<S2
ans =
     1     0     0
```

روش دیگر استفاده از دستور strncmp است. این دستور تشخیص می‌دهد که آیا n کاراکتر اول دو رشته مشابه هستند یا خیر؟ این دستور به حروف بزرگ و کوچک حساس است.

```
>> strncmp(str1,str3,2)
```

دستور strncmpi همان کار strncmp را انجام می‌دهد ولی نسبت به حروف بزرگ و کوچک حساس نیست.

```
>> strncmpi(str1,str3,4)
```

۵-۹-۱ آرایه‌های رشته‌ای: آرایه‌های رشته‌ای، آرایه‌هایی هستند که مولفه‌های آنها رشته‌های کاراکتری می باشند. امکان ایجاد آرایه‌های کاراکتری دو بعدی وجود دارد ولی بدین منظور، سطرهای آرایه باید دارای طول یکسانی باشند. اگر یکی از سطرها کوتاهتر از دیگری باشد آرایه کاراکتری تولید اشتباه می‌کند، زیرا برخلاف قانون بیان شده است.

```
>> V=['my name is ali  ';'ali is a student']
V =
my name is ali
ali is a student
>> size(V)
ans =
     2    16
```

آسان‌ترین راه برای ایجاد آرایه کاراکتری استفاده از char است.

```
>>name=char('ali','nasrin')
```

سایز متغیر name دو در شش است، یعنی در انتهای اسم ali سه فضای خالی گذاشته شده است. دستور deblank هرگونه فضای خالی اضافی را از انتهای یک رشته هنگام استخراج آن از درون یک آرایه پاک می‌کند.

```
>>deblank(name(1,:))
ans =
ali
```

دستور strcat دو یا چند رشته را به صورت افقی به هم متصل می‌کند. این تابع به فضاهای خالی درون رشته‌ها کاری ندارد ولی فضاهای خالی بین دو رشته را حذف می کند.

```
>>myname=strcat('Ali    ','Alamdari')
myname =
AliAlamdari
```

ولی دستورstrvcat دو یا چند رشته را به صورت عمودی به هم وصل می‌کند. خروجی هر دو دستور زیر یکسان می‌باشد.

```
>>strvcat(['Hello';'Yes  '])
>>strvcat('Hello','Yes  ')
```

دستور isletter مشخص می‌کند که یک کاراکتر حرف است یا خیر و در غیر این صورت صفر بر می‌گرداند.

```
>>isletter('AUT2002')
ans =
     1     1     1     0     0     0     0
```

دستور isspace به ازای فضاهای خالی یک بر می گرداند.

```
>>isspace('I can')
ans =
     0     1     0     0     0
```

دستور upper تمامی کاراکترهای حروف درون یک رشته را به حروف بزرگ و دستور lower تمامی کاراکترهای حروف درون یک رشته را به حروف کوچک تبدیل می‌کند.

```
>>lower(myname)
ans =
alialamdari
>>upper(myname)
ans =
ALIALAMDARI
```

یکی دیگر از دستورات کاربردی strrep است، این دستور عمل جستجو و جایگزینی را انجام می‌دهد. شکل کلی آن به صورت زیر است:

result=strrep(str,srch,repl)

که str همان رشته‌ای است که قرار است بررسی شود، srech رشته‌ای است که باید جستجو شود و repl رشته جایگزین شونده است.

```
>>strrep('Alamdari Ali','Ali','Hamid')
ans = Alamdari Hamid
```

برای تبدیل مقدار عددی به رشته از تابع num2str و برای تبدیل یک مقدار اسکالر به یک آرایه کاراکتری از دستور int2str استفاده می‌کنیم.

```
>> num = 20
>> disp(['your score is = ',num2str(num)])
your score is = 20
```

دستور num2str قدرت بیشتری به کاربر برای کنترل فرمت و شکل خروجی می‌دهد.

```
num2str(pi,15)
ans =
3.14159265358979
```

۱-۹-۶ آرایه‌های ساختمانی (Structure)

تاکنون آرایه‌هایی دیده‌اید که یا عددی هستند یا به صورت کاراکتری. ولی در قسمت Structure می‌توان انواع داده‌های عددی و کاراکتری را در فیلدهای مختلف یک ساختمان قرار داد. برای جدا کردن Structure و field باید آنها را با نقطه از هم جدا کرد.

```
>>student.name='aliakbar alamdari'
>>student.id='8126092'
>>student.marks=[18  19  17]
student =
     name: 'aliakbar alamdari'
       id: '8126092'
    marks: [18 19 17]
```

برای دیدن کل structure باید نام متغیر ساختمانی را بنویسیم:

```
>>student
>>student.marks(2)
ans =
    19
```

برای اضافه کردن المان دوم به ساختمان به صورت زیر عمل می‌کنیم:

```
>>student(2).name = 'nasrin alamdari';
>>student(2).id='831502021';
>>student(2).marks = [17   16   19];
```

سایز structure یک در دو است.

```
>> student
student =
1x2 struct array with fields:
    name
    id
    marks
```

برای نمایش نمرات دانشجویان باید هر دو student را فراخوانی کرد:

```
>>student(1:2).marks
ans =
    18    19    17
ans =
    17    16    19
>> student(1).name(1:2)
ans =
al
```

۱-۹-۷ آرایه‌های سلولی: آرایه‌های سلولی یک ظرف حاوی انواع داده‌ها می‌باشند. به‌طور مثال، آرایه‌های عددی، رشته‌ای ، ساختاری، و خود سلول‌ها ... . ممکن است یک سلول از آن، حاوی آرایه اعداد حقیقی بوده، سلول دیگر حاوی آرایه‌ای متشکل از رشته ها و دیگری حاوی آرایه‌ای از اعداد مختلط باشد.

روش اول برای قرار دادن یک عدد و یا رشته در یک آرایه سلولی استفاده از آکولاد {} به‌صورت زیر است:

```
>>c{1,1}=rand(3);
>>c{1,2}=char('ali','alamdari');
```

```
>>c{2,1}=13;
>>c{2,2}=student;
>> c
c =
    [3x3 double]    [2x8 char  ]
    [        13]    [1x2 struct]
```

برای دسترسی به مولفه‌های آرایه‌های سلولی بصورت زیر عمل می‌کنیم:

```
>>c{1,1}
ans =
    0.8147    0.9134    0.2785
    0.9058    0.6324    0.5469
    0.1270    0.0975
```

روش دوم برای ایجاد آرایه‌های سلولی به‌صورت زیر است:

```
>> C={rand(3),char('ali');13,student}
C =
    [3x3 double]    'ali'
    [        13]    [1x2 struct]
>> r =C{2,1}
r =
    13
```

نکته: اگر بخواهیم نوع محتویات سلول را ببینیم، از علامت پرانتز و اگر بخواهیم محتویات ساختار داده‌ای را ببینیم، از آکولاد استفاده می کنیم.

با دستور Cell می توان یک آرایه با ابعاد دلخواه را ایجاد کرده و سپس اعضا را مقدار دهی کرد.

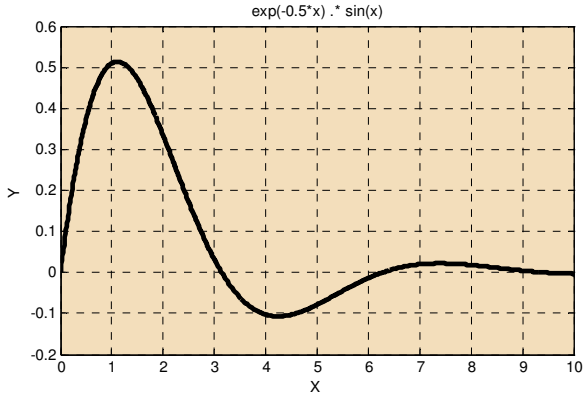
```
c=cell(2,2)
c =
     []     []
     []     []
c{1,1}='ali'
c =
    'ali'     []
      []      []
c{2,2}=12
c =
    'ali'      []
      []    [12]
```

دستور eval(s): این دستور رشته‌های حاوی اعداد را به مقادیر عددی تبدیل می‌کند.

```
>>a=input('how old are u? ','s')
    how old are u? 20
    a =20
>>a=eval(a)
a =
    20
>> f = input( 'Enter function (of x) to be plotted: ', 's')
```

با نوشتن 's' دیگر نیازی نیست که موقع ورود داده آپوستروف بگذارید. در صورتی‌که s گذاشته شود، باید عبارت در داخل یک جفت کوتیشن قرار بگیرد.

```
>> Enter function (of x) to be plotted:
exp(-0.5*x) .* sin(x)
>> x = 0:0.01:10;
>>plot(x, eval(f)),grid
```



شکل(۱-۸) منحنی exp(-0.5*x) * sin(x)

نکته: برای تبدیل اعداد از یک فرمت عددی به فرمت عددی دیگر از دستورات زیر استفاده می‌کنیم:

جدول(۱-۹) دستورات تبدیل داده‌ها

دستور	توضیح	مثال
bin2dec	مقدار باینری را به دسیمال تبدیل می‌کند.	>> bin2dec('10101')
hex2dec	مقدار هگز را به دسیمال تبدیل می‌کند.	>> hex2dec('ad4')
hex2num	اعداد هگز را که به صورت رشته هستند به اعداد با دقت مضاعف تبدیل می‌کند.	>> hex2num('afb54dd13')
dec2base base2dec	تبدیل اعداد اعشاری به هر پایه دیگـری از ۲ تا ۳۶ و یا برعکس را انجام می دهند.	>> b=dec2base(13,3) >> base2dec(b,3)
dec2bin	مقادیر دسیمال را به باینری تبدیل می‌کند.	>>dec2bin(87)

۱-۱۰ ذخیره سازی ، بازیابی داده‌ها و انتقال داده‌ها از Excel به MATLAB

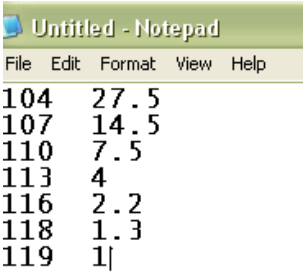
به جدول (۱-۱۰) توجه کنید.

جدول(۱-۱۰) دستورات ذخیره‌سازی و بازیابی داده‌ها

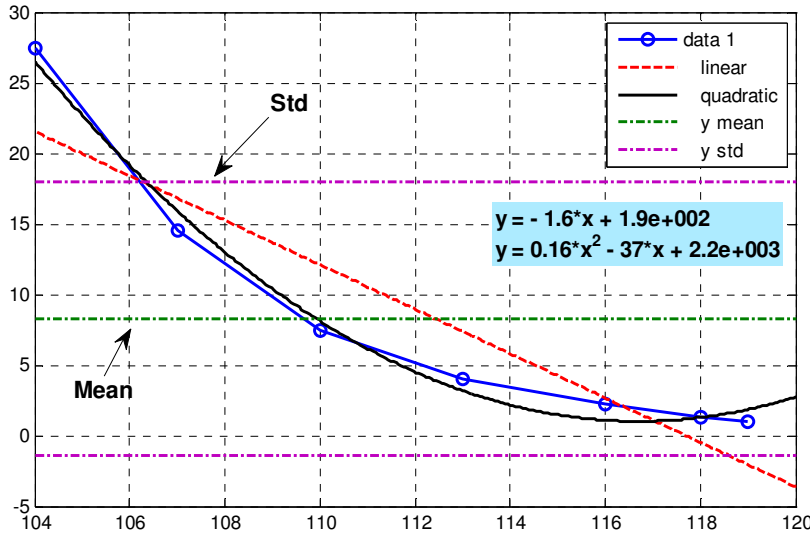
دستورات	توضیحات
save	مقادیر موجود در فضای کاری را در فایل matlab.mat ذخیره می‌کند.

save data	تمام مقادیر فضای کاری را در فایل data.mat ذخیره می‌کند.
save data_1 x y	متغیرهای x,y را در فایل data_1.mat ذخیره می‌کند.
load data_1	مقادیر فضای‌کاری که در فایل data_1 ذخیره شده بودند را باز می‌کند. این دستور فایل را خط به خط خوانده و هر مقدار داده توسط یک فاصله یا کاما جدا می‌شود. تعداد ستون داده‌ها در هر سطر و تعداد سطرها در هر ستون باید برابر باشد. هنگام ایجاد یک بردار سطری، اطلاعات بدون استفاده از فاصله وارد می‌شود.
delete file	باعث پاک شدن فایل در دایرکتوری جاری می‌شود.
open file	برای باز کردن فایل‌ها استفاده می‌شود.
what	نام تمامی m-fileها و matای موجود در current directory را ارائه می‌دهد.
diary file	تمامی textهای نوشته شده در پنجره command به‌جز علامت << را در فایل text به‌نام diary ذخیره می‌کند.
copyfile	برای کپی کردن فایل‌ها استفاده می‌شود.
movefile	برای جابه‌جا کردن فایل‌ها استفاده می‌شود.
lookfor	به دنبال یک m-file مشخص شده می‌گردد.
helpwin	پنجره help را باز می‌کند.
whos	لیست متغیرهای موجود در فضای کاری را می دهد.
cd	برای تغییر فضای کاری جاری استفاده می‌شود.
exit, quit	برای خروج از نرم افزار MATLAB استفاده می‌شود.

به‌طور مثال در یک فایل Notepad داده‌های زیر را وارد کرده، سپس آنها را با نام numbers.text ذخیره کنید.

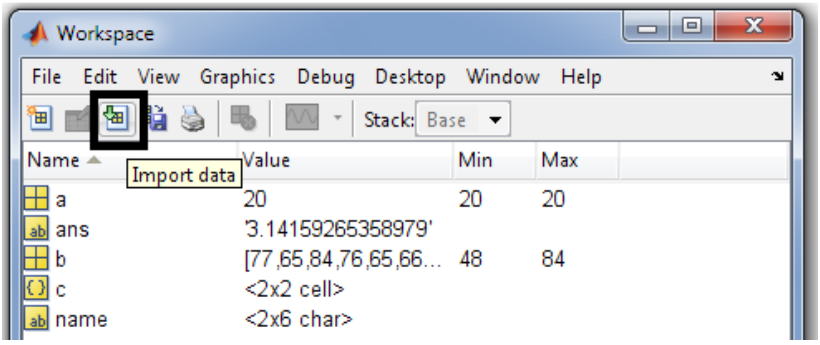



```
>> open('numbers.txt')
>> load numbers.txt
>> temp = numbers(:,1);
>> value = numbers(:,2);
>> plot (temp,value)
>> save data temp value
>> delete data.mat
```



شکل(۱-۹) رسم داده‌های فایل Notepad

برای انتقال داده‌ها از Excel به فضای‌کاری MATLAB، ابتدا باید از نوار ابزار Workspace گزینه Import data را انتخاب کرده، سپس از صفحه باز شده فایل Excel مورد نظر را انتخاب کرده و روی open کلیک کنید تا داده‌ها به محیط کاری MATLAB انتقال پیدا کنند. نام داده‌ها را می‌توانید قبل و بعد از کلیک کردن روی Finish عوض کنید. وقتی داده‌ها به فضای کاری MATLAB انتقال پیدا کردند، همان عملیاتی را که روی فایل Notepad مثال قبل انجام دادیم، تکرار می‌کنیم تا بتوانیم تحلیل روی داده‌های Excel را انجام دهیم.



شکل(۱-۱۰) وارد کردن داده‌ها از فایل Excel

۱-۱۱اعداد مختلط

اعداد مختلط کاربرد گسترده‌ای در علوم کاربردی، ریاضی، و مهندسی دارند. این اعداد را می‌توان به سه صورت نمایش داد:

۱- نمایش کارتزینی

۲- نمایش قطبی

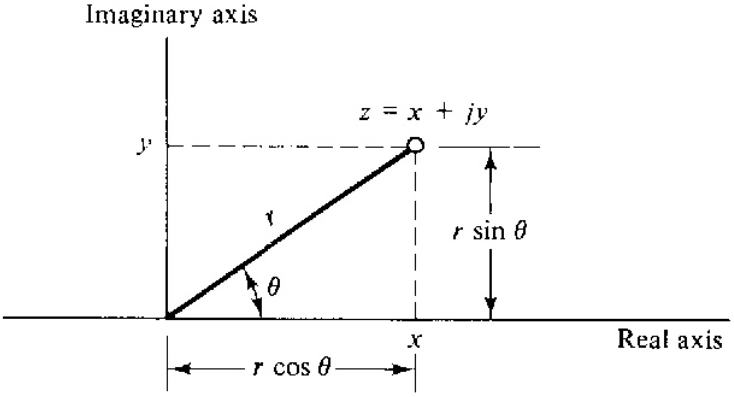
۳- نمایش نمایی

در نمایش کارتزینی، عدد مختلط z از دو قسمت حقیقی x و موهومی y تشکیل شده است. قسمت موهومی، به صورت جذر عدد ۱- تعریف شده است.

```
>> j = sqrt(-1);
```

$z = x + jy$

$x = \mathrm{Re}[z];\ y = \mathrm{Im}[z]$



شکل(۱-۱۱) نمایش اعداد مختلط

در MATLAB، به صورت پیش فرض متغیرهای i و j به‌عنوان اعداد موهومی می‌باشند. برای تولید این اعـداد از سه روش زیر استفاده می‌شود:

```
>> z = 1 + j*2
>> z = 1 + 2j
>> z = complex(1,2)
```

برای دسترسی به قسمت حقیقی و موهومی، دستورات زیر به‌کار می‌رود:

```
>> x = real(z)
>> y = imag(z)
```

در نمایش قطبی، از r به عنوان شعاع و از θ به عنوان زاویه عدد مختلط استفاده می‌شود.

$z = r\cos\theta + jr\sin\theta$

مقادیر r و θ با روابط و توابع MATLAB در جدول (۱-۸) محاسبه می‌شوند.

جدول (۱-۱۱) محاسبه مقادیر **r** و **θ**

MATLAB	تعریف ریاضی	
Abs(z)	$\lvert z\rvert = r = \sqrt{x^2 + y^2}$	بزرگی

زاویه	$\theta = \tan^{-1}(\frac{y}{x}) = \sin^{-1}(\frac{y}{r}) = \cos^{-1}(\frac{y}{r})$	**Angle(z)**

```
>> z = 3 + 4j;
>> r = abs(z)
>> theta = angle(z)
>> theta = (180/pi)*angle(z)
>> z1 = r*(cos(theta) + j*sin(theta))
z1 = 1.0000+ 1.7321i
```

در نمایش نمایی نیز ازرابطه اویلر استفاده می‌کنیم:

```
>> z = 3 + 4j
z = 3.0000+ 4.0000i
>> r = abs(z)
r = 5
>> theta = angle(z)
theta = 0.9273
>> z = r * exp(j*theta)
z = 3.0000+ 4.0000i
```

اعداد مختلط را می‌توان به راحتی با هم جمع، تفریق، و یا در هم ضرب وتقسیم کرد:

```
>> z1 = 1 + 2j
z1 = 1.0000+ 2.0000i
>> z2 = 4 + 3j
z2 = 4.0000+ 3.0000i
>> z3 = z1 + z2
z3 = 5.0000+ 5.0000i
>> z1 = 1 +0.5j; z2 = 2 + 1.5j;
>> z4 = z1 * z2
z4 = 1.2500+ 2.5000i
```

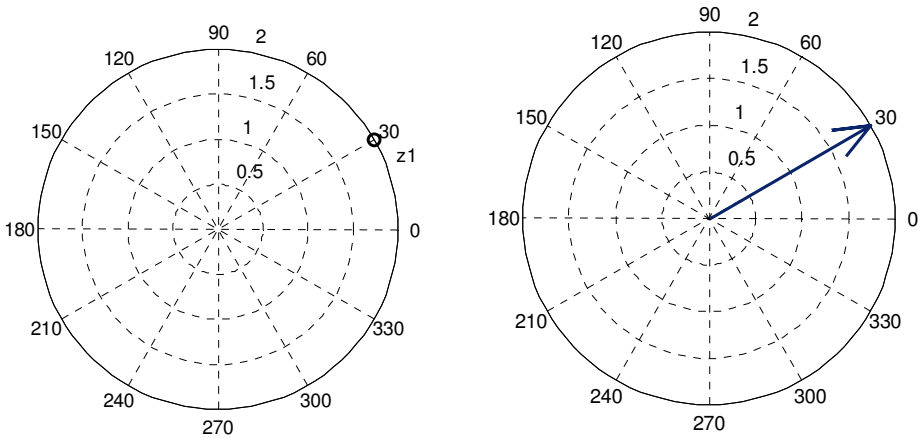
برای به‌دست آوردن مزدوج عدد مختلط از دستور conj استفاده می‌کنیم:

```
>> zconj = conj(z)
```

برای نمایش و رسم اعداد مختلط روی صفحه می‌توان از دستورات plot، polar، و compass استفاده کرد.

```
>> z = 1 + 0.5j;
>> plot(z,'*')
>> axis([-3 3 -3 3]);
>> text(real(z)+0.1,imag(z),'z');
>> xlabel('Real Part');
>> ylabel('Imaginary Part');
>> title('Complex Numbers');
```

```
>> z1 = sqrt(3) + j;
>> r1 = abs(z1);
>> phi1 = angle(z1);
>> polar(phi1,r1,'o')
>> text(real(z1)+0.1,imag(z1),'z1')
>> compass (z1)
```

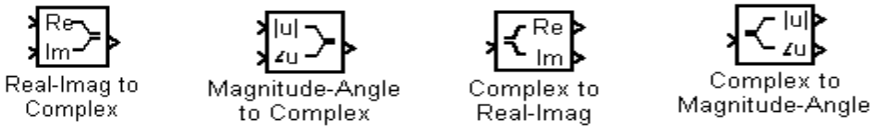


شکل(۱-۱۲) نمایش و رسم اعداد مختلط به‌صورت قطبی

۱-۱۲ نمایش اعداد مختلط در سیمولینک

در سیمولینک بلوک‌هایی تعریف شده‌اند، که از آنها برای انجام عملیات خاصی روی اعداد مختلط استفاده می شود. برای دسترسی به این بلوک‌ها ابتدا وارد قسمت سیمولینک شوید. به این منظور، دستور simulink را در پنجره دستورات تایپ کنید.

```
>> simulink
```



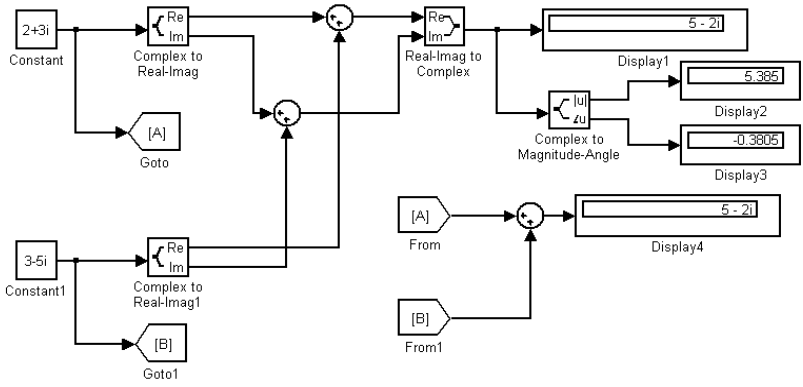
شکل(۱-۱۳) بلوک‌های سیمولینک برای انجام عملیات خاص روی اعداد مختلط

صفحه جدیدی باز کنید و بلوک‌ها را مطابق جدول (۱-۹) به این صفحه منتقل کرده و به‌هم متصل کنید. سپس روی Start simulation درنوار ابزار کلیک کنید تا شبیه سازی آغاز گردد. حال می‌توانید خروجی‌ها را در بلوک های Display مشاهده نمائید.

جدول (۱-۱۲) مسیر بلوک‌ها در سیمولینک

تعداد	مسیر بلوک
1	Simulink>>math operation>>Complex to magnitude-Angle

2	Simulink>>math operation>>Complex to Real-Image
1	Simulink >>Sinks>>Display
1	Simulink>>math operation>> Real-Image to Complex
2	Simulink >>Signal Routing>>Goto
2	Simulink >> Sources>> Constant
3	Simulink >> Math operations>> Sum
2	Simulink >>Signal Routing>>From



شکل(۱-۱۴) نحوه اتصال بلوک‌ها به‌یکدیگر

❖ تمرینات تکمیلی

۱- اگر x=10، y=20 و z=30 باشند، مقادیر a، b و c زیر را به‌دست آورید.

$$a=5x^2-6y+7z,\ b=\frac{3y^2}{4x-5z^3},\ c=(1+\frac{1}{x^2})^{-1}$$

۲- برای یک پوسته کروی با شعاع‌های خارجی متغیر $r_1=3,4,5,6,\ldots,10$ و شعاع داخلی $r_1=2$، مقادیر حجم پوسته کروی را با توجه به رابطه $v=\frac{4}{3}\pi(r_1^3-r_2^3)$ به‌دست آورید. سپس نمودار تابع غیرخطی $v=f(r_1)$ را بر حسب r_1 رسم کنید.

۳- اگر $A=\begin{bmatrix}6&9&5&1\\8&7&2&3\\1&3&4&4\\5&2&8&2\end{bmatrix}, B=\begin{bmatrix}4&8\\3&7\\2&3\\5&1\end{bmatrix}$، در این‌صورت

۳-۱ از دو ستون میانی ماتریس A، با استفاده از عملگر کولن ماتریس E1 را بسازید.

۳-۲ با استفاده از سطر اول و دوم و ستون دوم و سوم ماتریس A ماتریس E2 را بسازید.

۳-۳ با کنارهم گذاشتن ماتریس‌های B, E1 در کنار هم ماتریس E3 را بسازید.

۳-۴ حاصل‌ضرب مولفه‌های A_{24}, B_{12} را به‌دست آورید.

۴- در صورتی‌که ماتریس A برابر $\begin{bmatrix}12.11&-7.9&9.23\\5.06&6.35&21.7\\-3.34&2.67&14.38\end{bmatrix}$ باشد، در این صورت

۴-۱ لگاریتم طبیعی قدرمطلق مولفه‌های ماتریس A را به‌دست آورید

۴-۲ لگاریتم مبنای ۱۰ قدرمطلق مولفه‌های ماتریس A را به‌دست آورید.

۴-۳ جذر مولفه‌های ماتریس A را به‌دست آورید.

۴-۴ کسینوس هایپربولیک مولفه‌های ماتریس A را به‌دست آورید.

۴-۵ هر مولفه ماتریس A را به عدد صحیح بزرگتر گرد کنید.

۴-۶ مجموع مولفه‌های هر ستون ماتریس A را به‌دست آورید.

۴-۷ حاصل‌ضرب مولفه‌های هر سطر ماتریس A را به‌دست آورید.

۴-۸ بزرگترین و کوچکترین مقدار هر سطر ماتریس A را به‌دست آورید.

۴-۹ هر ستون ماتریس A را به صورت صعودی مرتب نمائید.

۴-۱۰سایز ماتریس A را بیابید.

۴-۱۱میانگین مقادیر هر ستون ماتریس را به‌دست آورید.

۵- مجموعه معادلات جبرخطی زیر را حل نمائید.

$$1-\begin{cases}6x-3y+4z=41\\12x+5y-7z=-26\\-5x+2y+6z=14\end{cases}$$

مقادیر x، y و z را به‌دست آورید.

$$2-\begin{cases}R_1i_1+R_2i_2-v_1=0\\-R_2i_2+R_3i_3+R_5i_5=0\\R_4i_4-R_3i_3+v_2=0\\-i_1+i_2+i_3+i_4=0\\-i_4-i_3+i_5=0\end{cases}$$

V1=5, V2=10, R1=470, R2=300, R3=560, R4=100, R5=1000

مقادیر جریان‌های i_1,i_2,i_3,i_4,i_5 را به‌دست آورید.

۶- مقادیر ویژه ماتریس‌های زیر را بیابید.

$$A=\begin{bmatrix}3 & 4\\4 & -2\end{bmatrix}$$

$$B=\begin{bmatrix}13 & -3 & 5\\0 & 4 & 0\\-15 & 9 & -7\end{bmatrix}$$

$$E=\begin{bmatrix}-1 & 2 & 1 & 3\\1 & 2 & 2 & -1\\0 & 4 & 3 & 2\\-1 & 6 & 4 & 5\end{bmatrix}$$

۷- رتبه ماتریس زیر را محاسبه کنید.

۸- سری‌های داده شده زیر را در بازه نشان داده شده τ رسم کنید. ۲۰۰ مرحله را برای جمع سری‌ها بکار ببرید. حل را بدون استفاده از دستور sum به‌دست آورید.

$$1-f(\tau)=\frac{2}{\pi}+\frac{4}{\pi}\sum_{n=1}^{N}\frac{1}{1-4n^2}\cos(2n\pi t)\qquad -1\le\tau\le 1$$

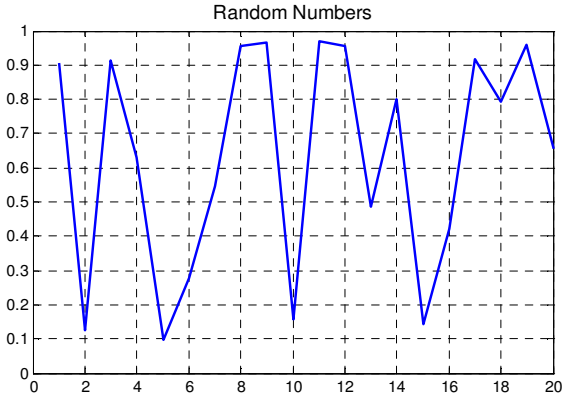
$$2-f(\tau)=\frac{4}{\alpha^2}\sum_{n=1,3,5}^{N}\frac{\sin(\alpha n\pi)}{(n\pi)^2}\sin(n\pi\tau)\qquad -2\le\tau\le 2\ ,\ \alpha=0.25$$

فصل دوم

ترسیم نمودارهای دو بعدی و سه بعدی

۲-۱ رسم نمودار با استفاده از دستور Plot

تابع plot متداول‌ترین تابع رسم نمودارهای دوبعدی می‌باشد. این تابع مجموعه‌ای از داده‌ها را بر روی صفحه مختصات رسم کرده و نقاط تعیین شده را با خطوط مستقیم به هم وصل می‌کند. plot(y) نمودار داده‌های y را رسم می‌کند و به‌طور پیش فرض، مقادیر x را اعداد طبیعی درنظر می‌گیرد. به عنوان مثال نمودار plot(rand(1, 20)) به‌صورت زیر خواهد بود:

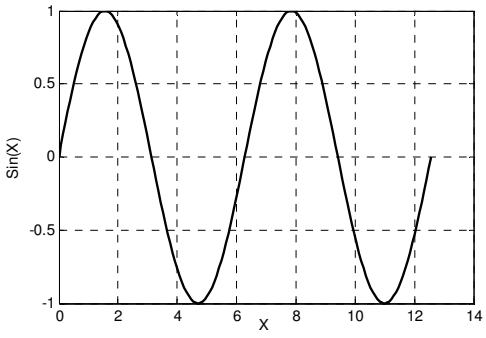


شکل (۲-۱) رسم اعداد تصادفی با تابع plot(y)

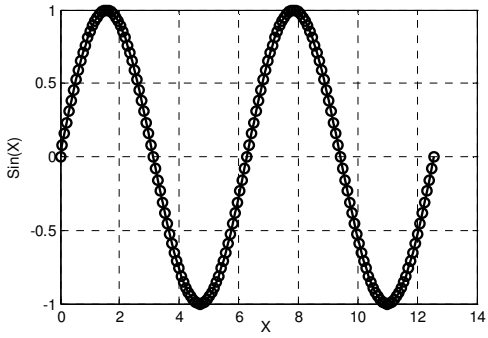
فرم رایج رسم نمودار به صورت plot(x,y) می‌باشد که در آن نمودار y برحسب x رسم می‌شود. توجه داشته باشید که اندازه بردار y و x برابرند.

```
>> x = 0:pi/40:4*pi;
>> y=sin(x);
>> plot(x, y)
```

در مثال بالا x محور افقی و y محور عمودی می‌باشد. در تابع plot آرگومان اول محور افقی و آرگومان دوم محور عمودی را مشخص می‌کند. تابع plot پنجره گرافیکی figure را باز کرده، سپس اندازه محورهای مختصات را مطابق داده‌ها تنظیم می‌کند. بعد از رسم نقاط روی صفحه، آنها را با خط راست به یکدیگر وصل می‌کند.



شکل (۲-۲) رسم منحنی سینوس



شکل (۲-۳) رسم منحنی سینوس با مارکر

تابع plot را می‌توان به همراه آرگومان سومی نیز به‌کار برد. این آرگومان که پس از x و y می‌آید یک رشته کاراکتری است که مشخص کننده نوع خطوط و رنگ آنها می‌باشد. این رشته شامل یک یا چند کاراکتر از جدول (۲-۱) می‌باشد.

جدول (۲-۱) انواع رنگ‌ها، مارکرها، و نوع خطوط در MATLAB

Specifier	Line Style
-	Solid line (default)
--	Dashed line
:	Dotted line
-.	Dash-dot line

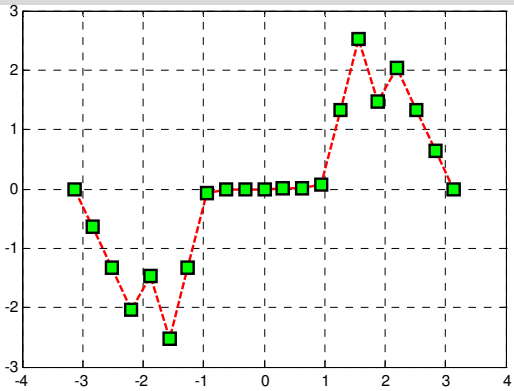
Specifier	Marker Type
+	Plus sign
o	Circle
*	Asterisk
.	Point
x	Cross
'square' or s	Square
'diamond' or d	Diamond
^	Upward-pointing triangle
v	Downward-pointing triangle
>	Right-pointing triangle
<	Left-pointing triangle
'pentagram' or p	Five-pointed star (pentagram)
'hexagram' or h	Six-pointed star (hexagram)

Specifier	Color
r	Red
g	Green
b	Blue
c	Cyan
m	Magenta
y	Yellow
k	Black
w	White

```
>> plot(x, sin(x),'ko-')
```

دستور plot می‌تواند آرگومان‌های دیگری نیز داشته باشد؛ مثلاً تغییر اندازه خطوط منحنی‌ها، تعیین اندازه و رنگ مارکرها که در مثال زیر این دستورات آورده شده‌اند.

```
>> x=-pi:pi/10:pi;
>> y=tan(sin(x))-sin(tan(x));
>> plot(x,y,'--rs','Linewidth',2,'Markersize',10,...
'MarkerFaceColor','g','MarkerEdgeColor','k')
```



شکل (۲-۴) رسم نمودار y=tan(sin(x))-sin(tan(x))

۲-۲ بررسی چند دستور مفید در رسم منحنی‌ها

جدول (۲-۲) دستورات مفید در رسم نمودارها

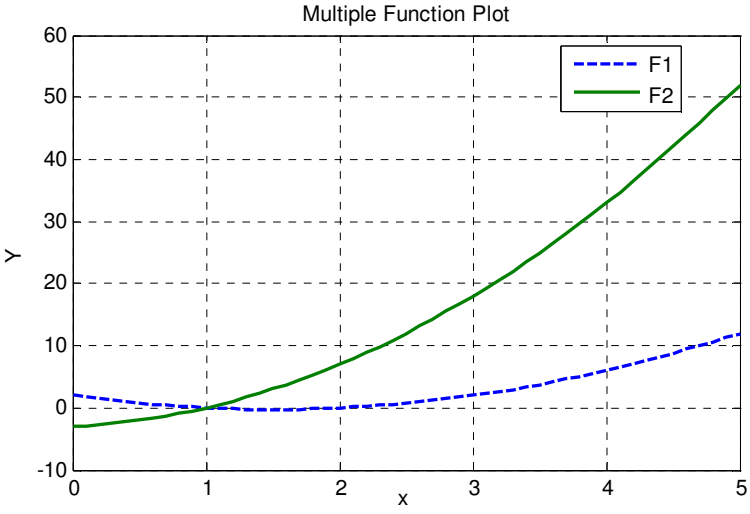
دستورات	توضیحات
`xlabel(' ')`	برچسب محور x ها را مشخص می‌کند.
`ylabel(' ')`	برچسب محور y ها را مشخص می‌کند.
`title(' ')`	عنوان را در بالای نمودار قرار می‌دهد.
`grid on,off`	grid on خطوط شبکه‌ای نمودار را فعال می‌کند و grid off آنها را غیرفعال می‌کند.
`text(x,y,' ')`	برای نوشتن متن روی نمودار از این دستور استفاده می‌شود. ۲ آرگومان اول مختصات متن می‌باشند.
`gtext(' ')`	در صورتی‌که مختصات متن را ندانیم از این دستور استفاده کرده و با ماوس در مکان مورد نظر کلیک می‌کنیم.
`hold on ,off`	این دستور اجازه می‌دهد چند منحنی در یک نمودار قرار بگیرند.
`figure(n)`	پنجره شماره n را برای رسم نمودار تولید می‌کند.
`close(n),close all`	پنجره و شکل شماره n را می‌بندد.
`axis([xmin xmax ymin ymax])`	با استفاده از این دستور می‌توانیم محورهای مختصات را در جهت xوy تغییر بدهیم.
`legend(' ')`	در صورتی‌که چند نمودار در یک پنجره باشند، از این دستور به‌عنوان راهنمای نمودار استفاده می‌شود.

توجه: دستور clc برای پاک کردن صفحه ورود دستورات، دستور clf برای پاک کردن تصویر موجود در یک شکل، و clear برای پاک کردن متغیرهای موجود در فضای کاری MATLAB می‌باشد.

۲-۳ رسم چند نمودار در یک صفحه

در مثال زیر دو تابع درجه دوم در یک پنجره رسم شده‌اند.

```
>> x = 0:0.1:5;
>> f(:,1) = x'.^2 -3*x' + 2;
>> f(:,2) = 2*x'.^2 +x' -3;
>> plot(x,f)
>> title('Multiple Function Plot')
>> xlabel('X'),ylabel('Y')
>> Grid on
>> legend('F1','F2')
```



شکل (۲-۵) رسم چند نمودار در یک صفحه

در این برنامه مقادیر دو تابع درجه دو در ستون‌های ماتریس f قرار گرفته‌اند. سپس با دستور plot ماتریس f بر حسب x رسم شده است. در برنامه بالا برای روی هم انداختن دو نمودار از دستور Hold on نیز می‌توان استفاده کرد.

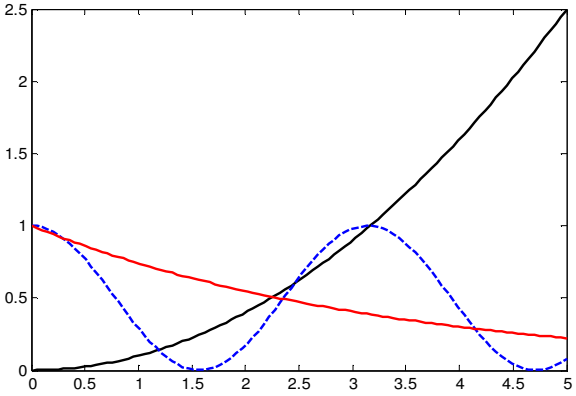
```
>> x = 0:0.1:5;
>> f1 = x.^2 -3*x + 2;
>> f2 = 2*x.^2 +x -3;
>> plot(x,f1)
>> hold on
>> plot(x,f2)
```

روش دیگر برای رسم چند نمودار در یک پنجره استفاده از فرمت زیر می‌باشد:

plot(x1, y1, x2, y2, x3, y3, ...)

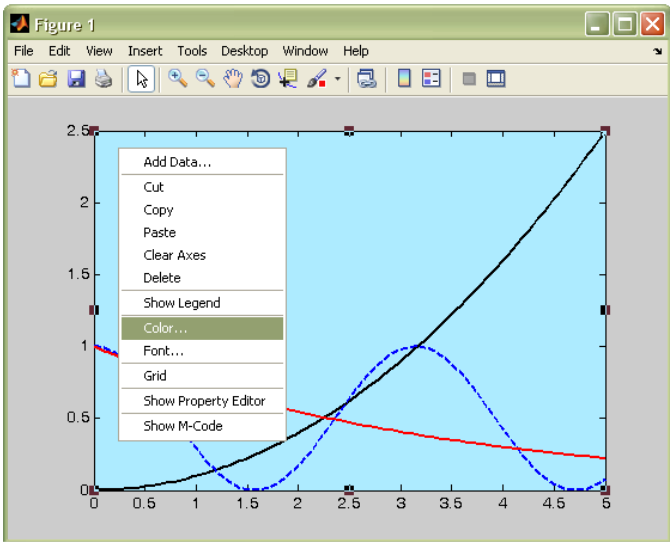
axis([xmin, xmax, ymin, ymax])

```
>>x=linspace(0,5);
>>plot(x,0.1*x.^2,'k',x,cos(x).^2,'b--',x,exp(-
0.3*x),'r')
```



شکل (۲-۶) رسم چند نمودار در یک صفحه

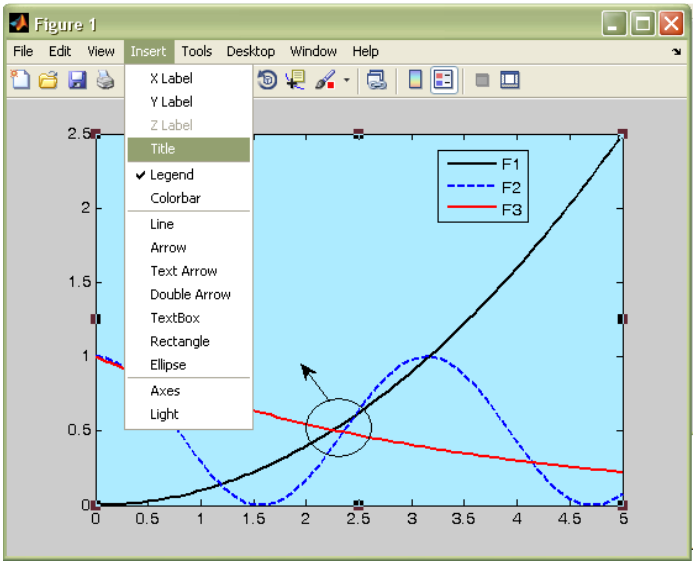
برای ویرایش شکل و نوشتن فرمولها روی شکل، برچسب گذاری محورها، تغییر رنگ منحنی‌ها، تغییر رنگ زمینه، تغییر خطوط، علائم راهنما و... ابتدا ازمنوی Tools گزینه Edit plot را انتخاب کنید. برای تغییر رنگ زمینه روی صفحه، کلیک راست کرده سپس گزینه color را انتخاب نموده و یکی از رنگها را انتخاب کنید.



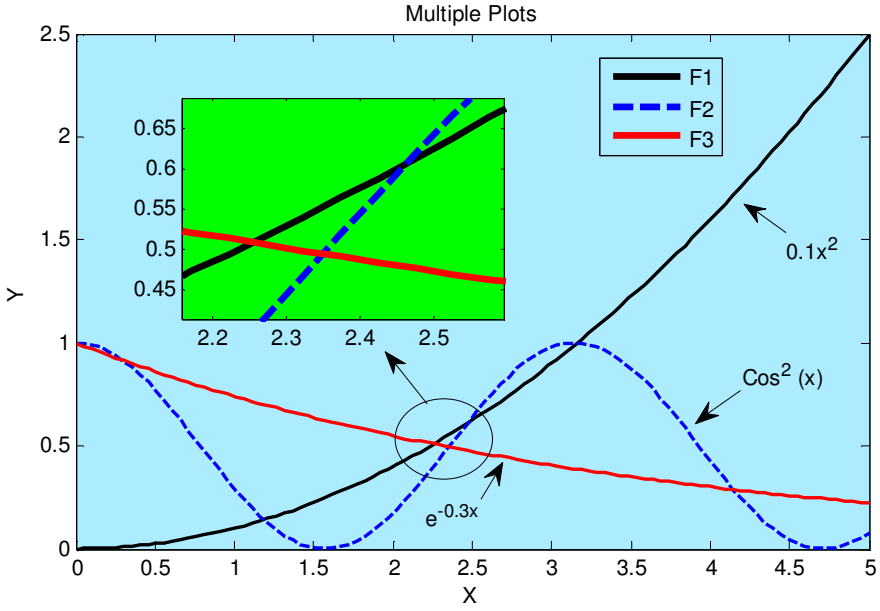
شکل (۲-۷) ویرایش نمودار، تغییر رنگ زمینه

برای برچسب گذاری روی محورهای x وy و عنوان نمودار، مطابق شکل زیر، به ترتیب از منوی Insert گزینه-های xlabel، ylabel و title را انتخاب کنید و برچسبهای مناسب را برای محورها بنویسید. سپس از همین

مسیر دستور legend را انتخاب کنید تا علائم راهنما بر روی شکل گذاشته شود. برای کشیدن بردار و بیضی دستورات Text Arrow, Ellipse را انتخاب نمائید و جای آنها را در شکل مشخص کنید.



شکل (۲-۸) ویرایش نمودار

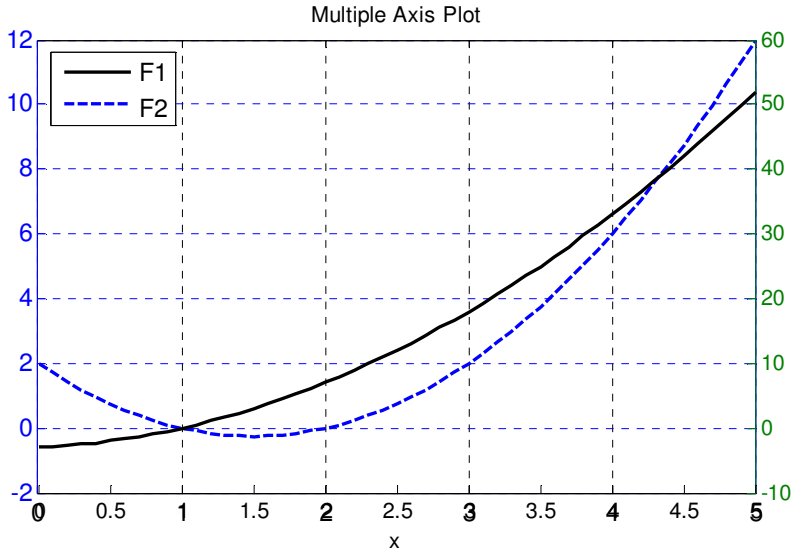


شکل (۲-۹) ویرایش نمودار، نوشتن text

برای رسم چند نمودار در یک شکل ولی با محورهای y می‌توان از دستور plotyy استفاده کرد:

```
>> x = 0:0.1:5;
```

```
>> f1 = x.^2 -3*x +2;
>> f2 = 2*x.^2 +x -3;
>> plotyy(x,f1,x,f2)
>> title('Multiple Axis Plot') , xlabel('x') ,grid
```



شکل (۲-۱۰) رسم دو منحنی با محورهای متفاوت

روش دیگر برای رسم چند نمودار در یک پنجره استفاده از دستور subplot می‌باشد. این دستور، پنجره را به چند قسمت تقسیم کرده و هر نمودار را در یکی از این قسمت‌ها رسم می‌کند. فرمت نوشتاری این دستور به صورت subplot(m,n,k) می‌باشد. این دستور پنجره را به یک ماتریس m در n تقسیم می‌کند و k امین خانه آن را انتخاب می‌کند. شماره هر خانه به صورت ردیفی تعیین می‌گردد. به‌عنوان نمونه به مثال زیر توجه کنید در خانه اول ایجاد شده توسط subplot ، منحنی cos(t) برحسب sin(t) رسم شده است.

```
t = 0:pi/20:2*pi;
[x,y] = meshgrid(t);
subplot(2,2,1)
plot(sin(t),cos(t))
axis equal
```

در خانه دوم ایجاد شده توسط subplot ، منحنی cos(x) +sin(x) برحسب t رسم شده است.

```
subplot(2,2,2)
z = sin(x)+cos(y);
plot(t,z)
axis([0 2*pi -2 2])
```

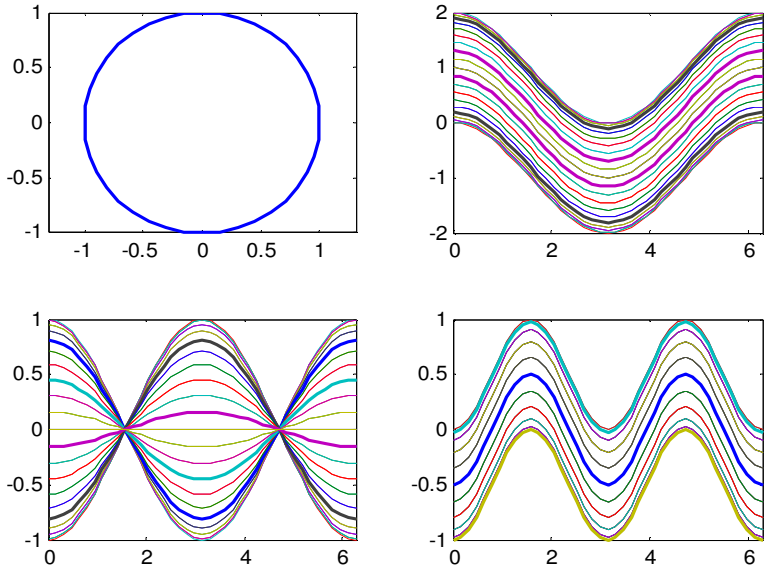
در خانه سوم ایجاد شده توسط subplot ، منحنی cos(y) *sin(x) برحسب t رسم شده است.

```
subplot(2,2,3)
z = sin(x).*cos(y);
plot(t,z)
```

```
axis([0 2*pi -1 1])
```

در خانه چهارم ایجاد شده توسط subplot ، منحنی (sin(x)^2)-(cos(y)^2) برحسب t رسم شده است.

```
subplot(2,2,4)
z = (sin(x).^2)-(cos(y).^2);
plot(t,z)
axis([0 2*pi -1 1])
```



شکل (۲-۱۱) رسم چند نمودار در یک پنجره با دستور subplot

مثـال: چنـد جملـه‌ای مرتبـه پـنج $f(x)=x^5-10x^4+35x^3-50x^2+24$ و سـیگنال میـرای $x(t)=te^{-t}\cos(8\pi t)$ را در یک پنجره رسم کنید.

```
>> f=[1 -10 35 -50 0 24];
```

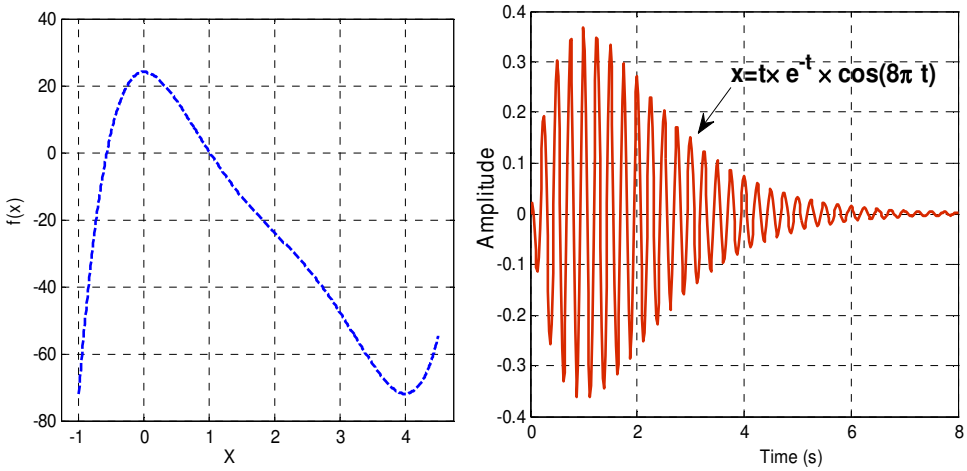
یک چند جمله‌ای در MATLAB به صورت یک بردار سطری است که مولفه‌های آن ضرایب چندجمله‌ای می باشند. این ضرایب به‌صورت نزولی مرتب شده اند.

```
>> x=-1:0.01:4.5;
>> y=polyval(f,x);
```

تابع polyval مقدار چندجمله‌ای f را در نقاط x محاسبه می‌کند.

```
>> subplot(1,2,1)
>> plot(x,y)
>> xlabel('X'), ylabel('f(x)'), grid on
>> t = linspace(0, 8, 401);
>> x = t.*exp(-t).*cos(8*pi*t);
>> subplot(1,2,2)
>> plot(t,x)
```

```
>> xlabel('Time (s)'), ylabel('Amplitude'), grid on
```



شکل (۲-۱۲) رسم منحنی‌های $f(x) = x^5 - 10x^4 + 35x^3 - 50x^2 + 24$ و $x(t) = te^{-t}\cos(8\pi t)$

۲-۴ رسم منحنی‌های لگاریتمی

برای رسم نمودارهای لگاریتمی و نیمه لگاریتمی از دستورات semilogx,semilogy,loglog(x,y) استفاده می‌کنیم.

```
% Generate subplots of a polynomial
x = 0:0.5:10;
y = 5*x.^3-2*x.^2+30;
```

در خانه اول ایجاد شده توسط subplot ، محور x و y هر دو خطی می‌باشند.

```
subplot(2,2,1),plot(x,y)
title('Polynomial - linear/linear (plot)')
ylabel('y'),grid
```

در خانه دوم ایجاد شده توسط subplot، محور x لگاریتمی و محور y خطی می‌باشد.

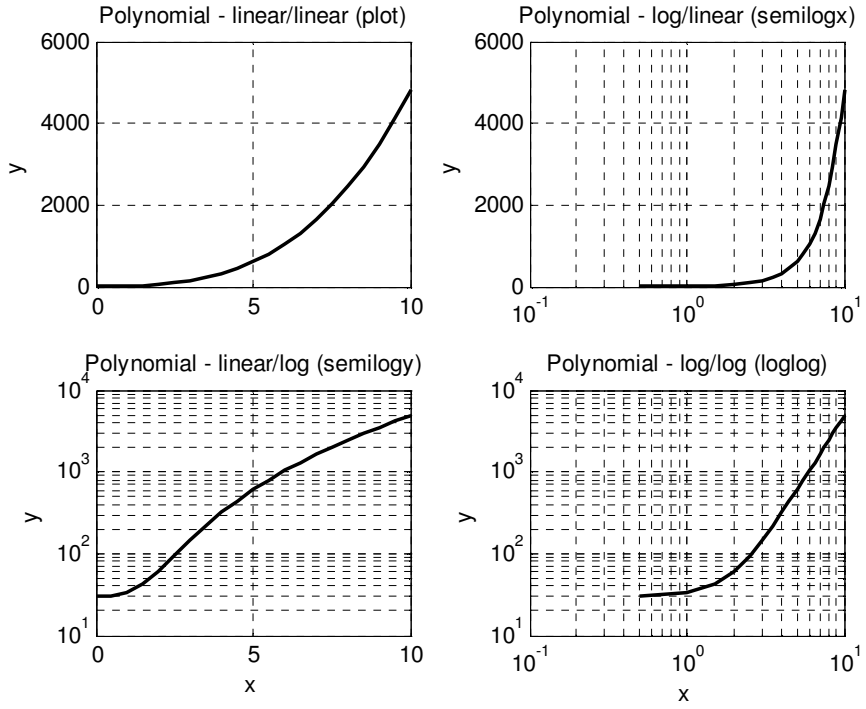
```
subplot(2,2,2),semilogx(x,y)
title('Polynomial - log/linear (semilogx)')
ylabel('y'),grid
```

در خانه سوم ایجاد شده توسط subplot ، محور y لگاریتمی و محور x خطی می‌باشد.

```
subplot(2,2,3),semilogy(x,y)
title('Polynomial - linear/log (semilogy)')
xlabel('x'),ylabel('y'),grid
```

در خانه چهارم ایجاد شده توسط subplot ، محورهای x و y هر دو لگاریتمی می‌باشند.

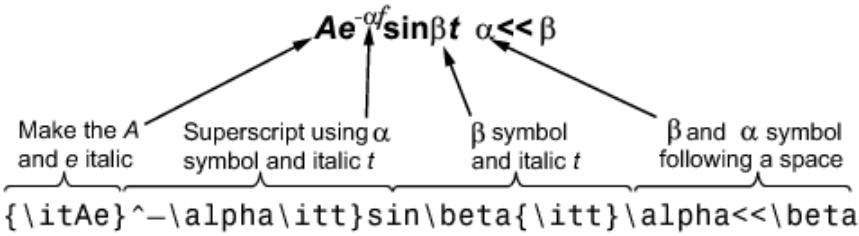
```
subplot(2,2,4),loglog(x,y)
title('Polynomial - log/log (loglog)')
xlabel('x'),ylabel('y'),grid
```



شکل (۲-۱۳) رسم منحنی‌های لگاریتمی

۲-۵ نوشتن متون ریاضی سیمبولیک بر روی نمودارها

برای نوشتن یک متن ریاضی سیمبولیک با املای مخصوص از پیشوند \ استفاده می‌کنیم. برخی از سمبل‌ها در جدول زیر آورده شده‌اند. نیازی به حفظ کردن این سمبل‌ها نیست. برای دسترسی به این علائم از help نرم‌افزار استفاده کنید و Annotation Textbox Properties را جستجو کنید. فهرست کامل این علائم در این جدول (۲-۳) موجود می‌باشد.



جدول (۲-۳) سمبل‌های کاربردی و مفید برای نوشتن متن روی نمودارها

رشته کاراکتری	سمبل	رشته کاراکتری	سمبل
\alpha	α	\upsilon	υ
\beta	β	\phi	φ
\gamma	γ	\chi	χ
\delta	δ	\psi	ψ
\epsilon	ε	\omega	ω
\zeta	ζ	\Gamma	Γ
\eta	η	\Delta	Δ
\theta	Θ	\Theta	Θ
\vartheta	ϑ	\Lambda	Λ
\iota	ι	\Xi	Ξ
\kappa	κ	\Pi	Π
\lambda	λ	\Sigma	Σ
\mu	μ	\Upsilon	ϒ
\nu	ν	\Phi	Φ
\xi	ξ	\Psi	Ψ
\pi	π	\Omega	Ω
\rho	ρ	\forall	∀
\sigma	σ	\exists	∃
\varsigma	ς	\ni	∋
\tau	τ	\cong	≅

به عنوان مثال برای نوشتن متن نوشته شده در شکل بالا($x(t) = te^{-t}\cos(8\pi t)$) از دستور زیر استفاده شده است.

```
>> gtext('x=t\times e^{-t}\times cos(8\pi t)')
```

۲-۶رسم منحنی‌های سه بعدی

برای رسم نمودارهای سه بعدی از دستورات mesh، surf، plot3 و contour استفاده می‌کنیم.

۲-۶-۱رسم منحنی تابع دو متغیره $f(x,y)=80y^2e^{-x^2-0.3y^2}$, $-2.1\le x\le 2.1, -6\le y\le 6$ با توابع mesh,contour

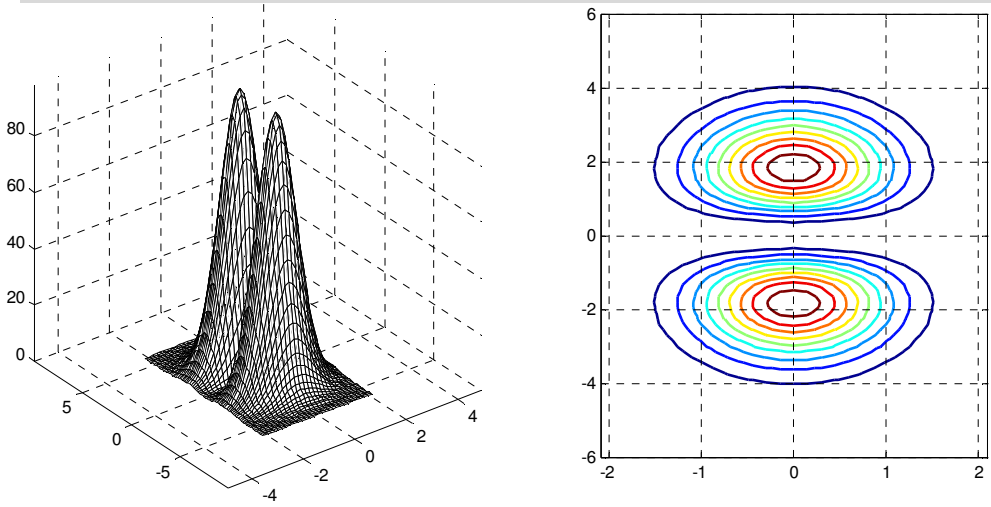
برای رسم سطوح سه بعدی مش بندی شده از این دستور استفاده می شود الگوی نوشتاری این دستور به صورت زیر است:

mesh(x,y,z)

دستور contour نیز کانتور ماتریس z را رسم می‌کند و الگوی نوشتاری این دستور به صورت زیر است:

contour(x,y,z,linespec)

```
>> [x, y] = meshgrid(-2.1:0.15:2.1,-6:0.15:6);
>> z = 80 * y.^2 .* exp(-x.^2 - 0.3*y.^2);
>> subplot(1,2,1)
>> mesh(x,y,z)
>> subplot(1,2,2)
>> contour(x,y,z)
```



شکل (۲-۱۴) رسم تابع دو متغیره $f(x,y)=80y^2e^{-x^2-0.3y^2}$ با تابع کانتور

۲-۶-۲ رسم منحنی $f(x,y)=Cos(x).Cos(y)$ با تابع surf:

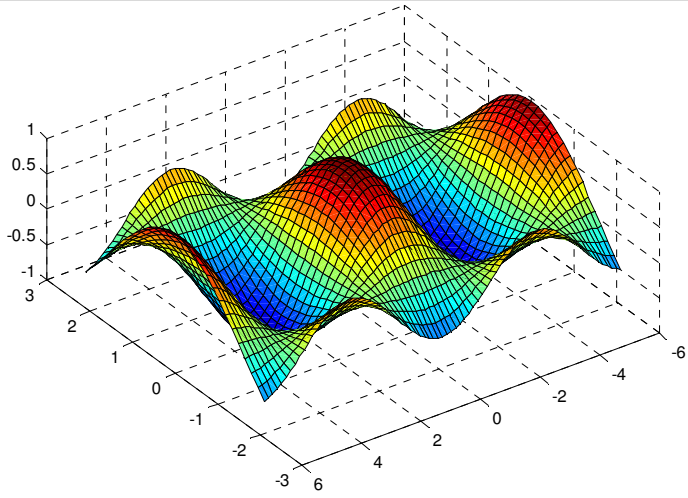
برای رسم سطوح سه بعدی از دستور surf استفاده می‌کنیم الگوی نوشتاری این دستور به صورت زیر می باشد:

surf(z)

surf(x,y,z)

که سایز ماتریس های x,y,z باید با هم برابر باشد.

```
>> [x,y] = meshgrid(-2.1:0.15:2.1,-6:0.15:6);
>> z=cos(x).*cos(y);
>> surf(x,y,z)
```



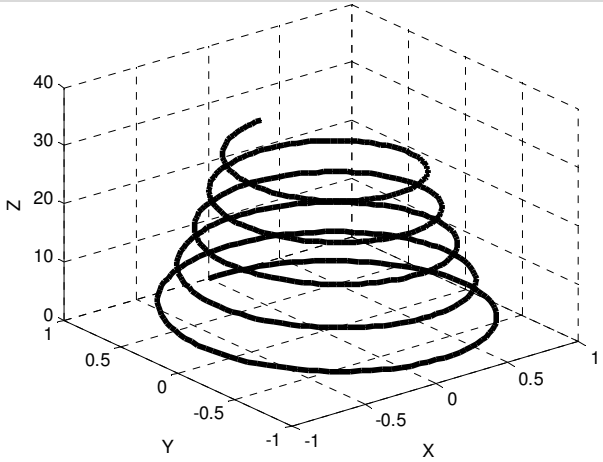
شکل (۲-۱۵) رسم منحنی $f(x,y)=Cos(x).Cos(y)$ با تابع surf

۲-۶-۳ رسم منحنی $\{x(t)=e^{-0.02t}Sin(t),\ y(t)=e^{-0.02t}Cos(t),\ z(t)=t\}$ با تابع plot3 :

برای رسم منحنی های سه بعدی پارامتریک ازدستور plot3 استفاده می‌شود. الگوی نوشتاری این دستور به صورت زیر است:

plot3(x1,y1,z1,linespec)
plot3(x1,y1,z1,x2,y2,z2,…)

```
>> t=0:pi/50:10*pi;
>> plot3(exp(-0.02*t).*sin(t), exp(-0.02*t).*cos(t),t)
```

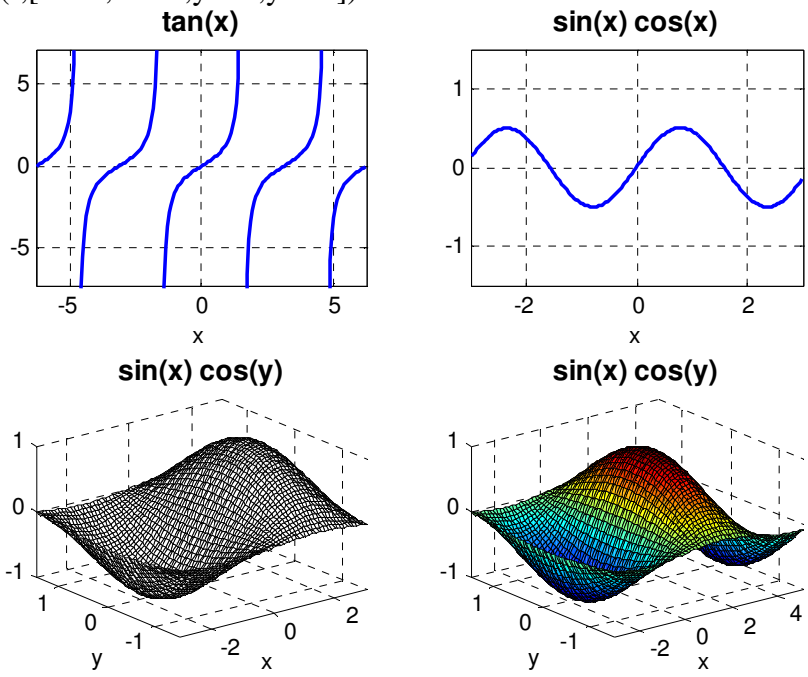


شکل (۲-۱۶) رسم منحنی با تابع plot3

۷–۲ رسم توابع با استفاده از ezplot,ezmesh,ezsurf

MATLAB برای رسم توابع مجموعه‌ای از دستورات ارائه می‌کند که با کاراکتر (easy- to- use) ’ez‘ آغاز می‌شوند. این دستور به‌طور خودکار معادله تابع را در عنوان نمودار قرار می‌دهد. فرمت‌های این توابع به صورتهای زیر می‌باشند:

ezplot(f)
ezplot(f,[min,max])
ezplot(f,[xmin,xmax,ymin,ymax])



شکل (۲–۱۷) رسم منحنی با تابع ezplot

ezmesh(f,[xmin,xmax,ymin,ymax])
ezsurf(f,[xmin,xmax,ymin,ymax])

در صورتی که محدوده x,y تعیین نشده با شد به صورت پیش فرض $\pi<x<2\pi,-2\pi<y<2\pi$ در نظر گرفته می شود. به‌طور مثال:

```
>> ezplot('tan(x)')
>> ezplot('sin(x)*cos(x)',[-3 3 -1.5 1.5])
>> ezmesh('sin(x)*cos(y)',[-3 3 -1.5 1.5])
>> ezsurf ('sin(x)*cos(y)',[-3 5 -1.5 1.5])
```

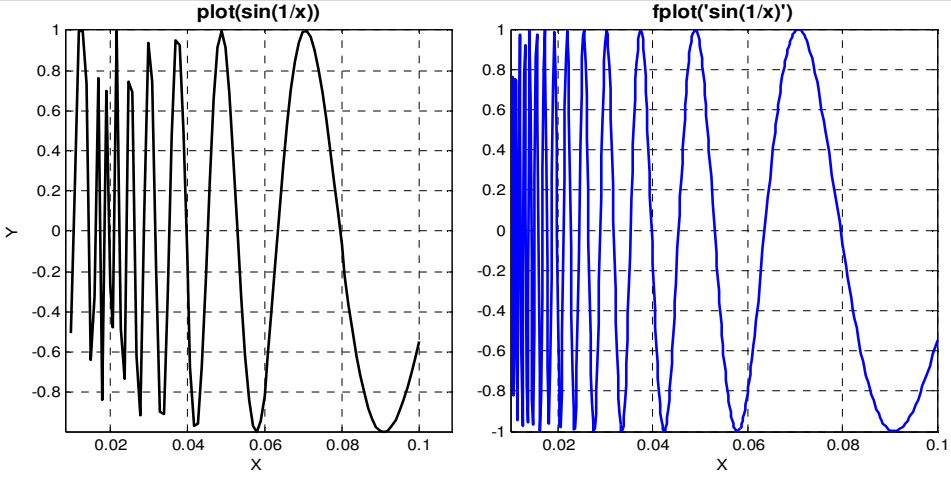
۲-۸توابع خاص برای رسم نمودارها

علاوه بر رسم منحنی‌ها با استفاده از توابع گفته شده، MATLAB توابعی دیگر برای رسم منحنی‌ها پیشنهاد می‌دهد که در ادامه توضیح داده می‌شوند.

۲-۸-۱ دستور fplot: در صورتی‌که تغییرات در تابع مورد نظر در بعضی بازه‌ها زیاد باشد، استفاده از دستور plot کارآمد نخواهد بود. بنابراین MATLAB دستور دیگری به نام fplot را پیشنهاد می‌کند که مقادیر تابع در این نقاط را سریعتر ارزیابی می‌کند. الگوی نوشتاری دستور fplot به صورت‌های زیر است:

fplot(fun,limits,linespec)
[X,Y]=fplot(fun,limits,linespec)

```
>> x = 0.01:.001:.1;
>> plot(x, sin(1./x))
>> fplot('sin(1/x)', [0.01 0.1])
```

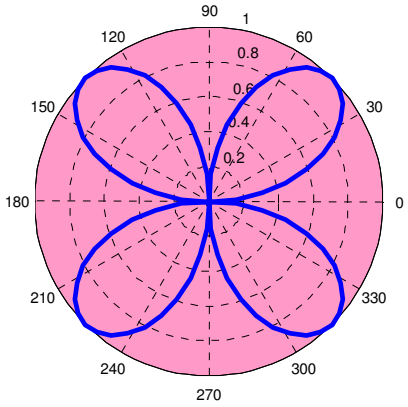


شکل (۲-۱۸) رسم منحنی با fplot

۲-۸-۲دستور polar: نقطه (x,y) را در مختصات قطبی به صورت (r,θ) نمایش داده می‌دهد. الگوی نوشتاری این دستور به‌صورت زیر است.

polar(theta, r)
polar(theta,r,'linespec')

```
>>theta = 0:pi/40:2*pi;
>>polar(theta, sin(2*theta))
>>grid
```

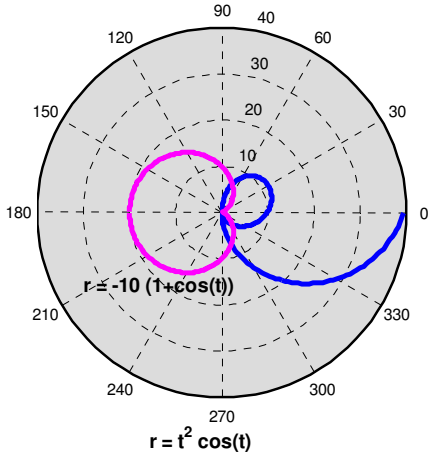


شکل (۲-۱۹) رسم منحنی با استفاده از تابع polar

۲-۸-۳ دستور ezpolar: این تابع نیز مشابه تابع polar می‌باشد با این تفاوت که این تابع نیازی به تعریف بازه برای متغیر ندارد و برای θ به‌طور پیش فرض مقداری بین 0تا 2*pi در نظر گرفته می‌شود، به‌جز در حالتی‌که بخواهیم بازه را تغییر دهیم که در این صورت باید از الگوی دوم استفاده شود. الگوی نوشتاری این دستور به‌صورت‌های زیر است:

ezpolar(f)
ezpolar(f,[a,b])

```
>> ezpolar('t^2*cos(t)')
>> hold on
>> ezpolar('-10*(1+cos(t))',[0,2*pi])
```



شکل (۲-۲۰) رسم منحنی با استفاده از تابع ezpolar

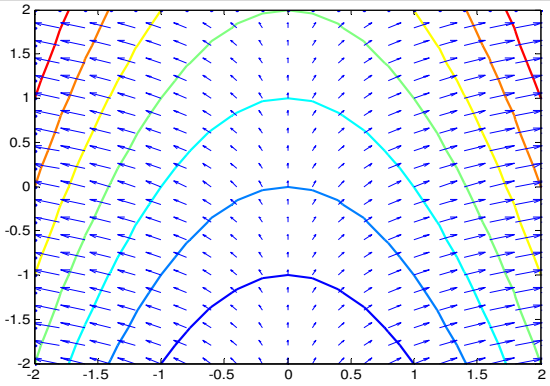
۲-۸-۴دستور quiver و quiver3: از این تابع برای رسم میادین برداری و گرادیان‌ها استفاده می‌شود. الگوی نوشتاری دستورهای quiver و quiver3 به صورت های زیر است:

quiver(x,y,u,v)
quiver(u,v)
quiver(x,y,u,v,linespec)
quiver3(x,y,z,u,v,w)
quiver3(z,u,v,w)
quiver3(x,y,z,u,v,w,linespec)

در قسمت linspec می‌توان نوع، مارکر و رنگ خطوط را تغییر داد.

مثال۱: تابع دو متغیره $V = x^2 + y$ را درنظر بگیرید. گرادیان این تابع عددی برابر $\nabla V = (2x,1)$ است. برای رسم این میدان برداری دستورات زیر را وارد می‌کنیم.

```
>>[x y] = meshgrid(-2:.2:2, -2:.2:2);
>>V = x.^2 + y;
>>dx = 2*x;
>>dy = dx; % dy same size as dx
>>dy(:,:) = 1; % now dy is same size as dx but all 1's
>>contour(x, y, V), hold on
>>quiver(x, y, dx, dy)
```



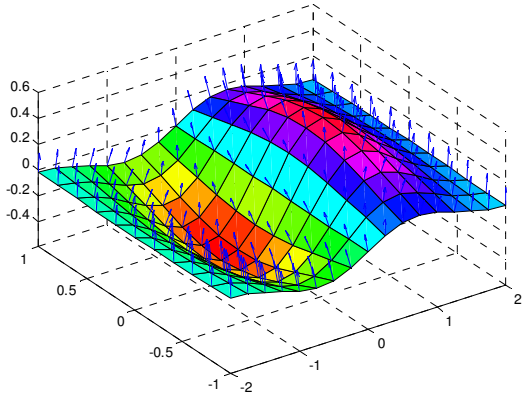
شکل (۲-۲۱) رسم میادین برداری

مقدار گرادیان تابع عددی نیز در صورتی‌که گام‌ها در جهت x,y برابر 0.02 باشد، به‌صورت زیر به‌دست می‌آید:

```
>>[dx1 dy1] = gradient(V, 0.02, 0.02);
```

مثال ۲: بردار نرمال بر سطح تابع دو متغیره $z = xe^{(-x^2-y^2)}$ را رسم کنید.

```
[X,Y] = meshgrid(-2:0.25:2,-1:0.2:1);
Z = X.* exp(-X.^2 - Y.^2);
[U,V,W] = surfnorm(X,Y,Z);
quiver3(X,Y,Z,U,V,W,0.5);
hold on
surf(X,Y,Z);
view(-35,45)
axis ([-2 2 -1 1 -.6 .6])
```



شکل (۲-۲۲) بردار نرمال بر سطح تابع

دستور surfnorm بردارهای سه بعدی نرمال بر سطح را محاسبه می‌کند. الگوی نوشتاری این دستور به صورت زیر است:

[dx,dy,dz]=surfnorm(x,y,z)

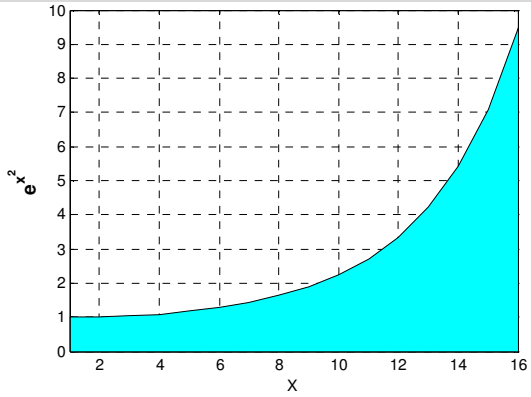
۲-۸-۵دستور area: از این تابع برای رسم و نشان دادن سطح زیر منحنی استفاده می‌شود. الگوی نوشتاری این دستور به صورت زیر می باشد:

area(y)
area(x,y,basevalue)

این دستور همانند دستور plot(x,y) است؛ ولی در این دستور سطح بین 0و y با رنگ خاصی پر می شود. در حالت پیش فرض مقدارbasevalue برابر صفر است ولی می‌توان این مقدار را تغییر داد. در صورتی‌که y ماتریس باشد،area(x,y) ستون‌های ماتریس y را رسم می‌کند و سطح زیر نمودار را پر می‌کند.

y=[1 2 3; 4 5 6;7 8 9;1 2 3]; area(y);

```
>>x = 0:0.1:1.5;
>>area(exp(x.^2))
```



شکل (۲-۲۳) رسم و نشان دادن سطح زیر منحنی با تابع area

۲-۸-۶دستور bar و bar3: از این تابع برای رسم منحنی به‌صورت ستونی استفاده می‌شود. الگوی نوشتاری دستور bar ,bar3 به صورت‌های زیر می‌باشد:

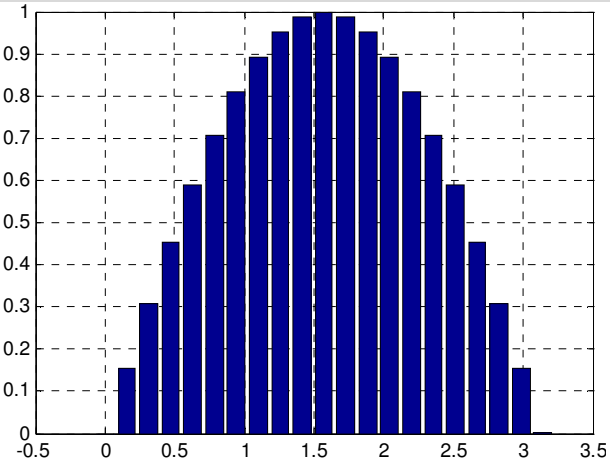
bar(y)
bar(x,y)
bar(x,y,width,'style,'bar color')

که در آن style یکی از حالت‌های 'hist,'histc','stacked','grouped' می‌باشد.

bar3(x,y,width,'style',linespec)

که در آن style یکی از حالت‌های 'stacked','grouped','detached'می‌باشد.

```
>> bar3(10*rand(7,3),'detached')
>>x = 0:pi/20:pi;
>> bar(x,sin(x))
```



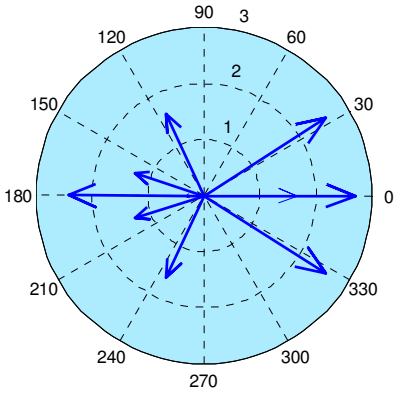
شکل (۲-۲۴) رسم منحنی با استفاده از تابع bar

۲-۸-۷ دستور compass: از این تابع برای رسم بردارهای قطبی و بردارهای مکانی استفاده می‌شود. الگوی نوشتاری این دستور به صورت‌های زیر می‌باشد.

compass(u,v)
compass(z)
compass(z,linespec)

در قسمت linspec می‌توان نوع، مارکر، و رنگ خطوط را تغییر داد. این دستور بردارهایی با مولفه‌های (u,v) را رسم می‌کند. u,v,z هر سه در دستگاه مختصات کارتزین بوده و روی یک دایره شبکه‌بندی شده رسم می‌شوند.

```
>> z = eig(randn(10,10));
>> compass(z)
```



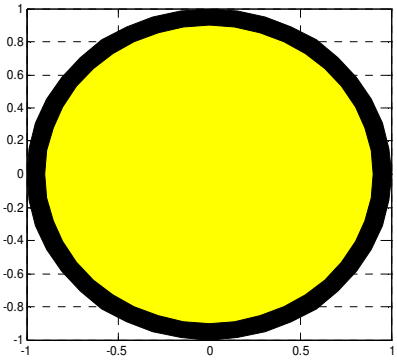
شکل (۲-۲۵) رسم بردارهای قطبی با استفاده از تابع compass

۲-۸-۸ دستور fill: از این تابع برای پرکردن سطح داخلی چندضلعی‌ها استفاده می‌شود. الگوی نوشتاری دستور fill به صورت های زیر می باشد:

```
fill(x,y,colorspec)
fill(x1,y1,c1,x2,y2,c2,…)
```

به‌طور مثال:

```
>> t = 0:pi/20:2*pi;
>> fill(cos(t),sin(t),'k',0.9*cos(t),0.9*sin(t),'y')
```



شکل (۲-۲۶) پر کردن سطح داخلی با استفاده از تابع fill

۲-۸-۸ دستور stairs: از این تابع برای رسم منحنی به صورت پله‌ای یا دیجیتالی استفاده می‌شود. الگوی نوشتاری دستور stairs به صورت های زیر می باشد.

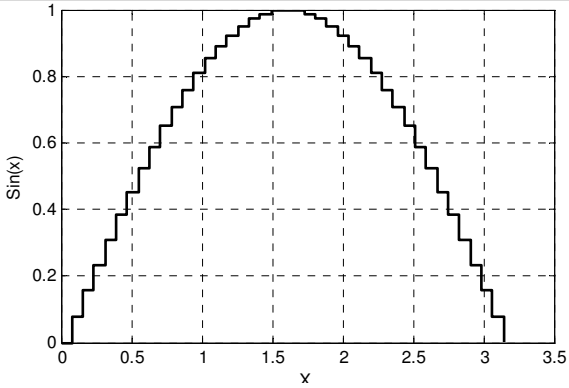
```
stairs(y)
stairs(x,y)
stairs(x,y,linespec)
```

در قسمت linspec می‌توان نوع، مارکر، و رنگ خطوط را تغییر داد.

```
[xb,yb]=stairs(y)
```

این دستور گرافی رسم نمی‌کند بلکه مقادیر xb,yb را بر می‌گرداند که می‌توان به این طریق با استفاده از دستور plot(xb,yb) نمودار پله‌ای را رسم کنیم.

```
>>x = 0:pi/40:pi;
>> stairs(x,sin(x),'k-','linewidth',2)
```



شکل (۲-۲۷) رسم منحنی به صورت پله‌ای با استفاده از تابع stairs

۲-۸-۹ دستور errorbar: دستور errorbar برای نمایش میزان انحراف داده ها استفاده می‌شود. الگوی نوشتاری این دستور به صورت زیر است:

errorbar(Y,E)

این دستور ماتریس Y را رسم کرده و میزان انحراف آن را با استفاده از ماتریس E رسم می کند. نمودار خطا برای هر درایه Y با توجه درایه متناظر از E رسم شده و به صورت یک خط متقارن در بالا و پایین آن رسم می‌شود.

errorbar(X,Y,E)

برای رسم کردن ماتریس Y در مقابل ماتریس X از این شکل دستور استفاده می‌شود. ماتریسهای X,Y,E باید هم اندازه باشند. اگر هر سه به صورت بردار باشند، نمودار خطا به صورت فاصله E(i) برای هر (x(i)),Y(i)) در بالا و پایین نقطه رسم می‌شود. اگر X,Y,E به صورت ماتریس باشند هر نمودار به صورت فاصله‌ی E(i,j) در بالا و پایین نقطه (x(i),j),Y(i,j)) رسم می‌شود.

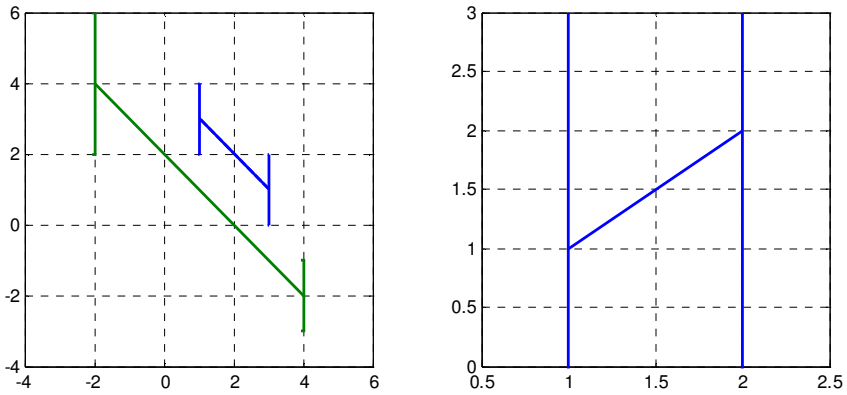
```
errorbar([3 4;1 -2],[1 -2;3 4],[1 1;1 2])
plot([3 4;1 -2],[1 -2;3 4])
```

گاهی اوقات لازم است که میزان انحراف از بالا و پایین یکسان نباشد. برای این حالت از این دستور استفاده می‌شود.

errorbar(X,Y,L,U)

ماتریسهای X,Y,L,U باید هم اندازه باشند. ماتریس L مشخص‌کننده میزان انحراف از پایین و U میزان انحراف از بالا را نشان می‌دهد.

```
errorbar([1,2],[1,2],[1,2],[2,1])
```



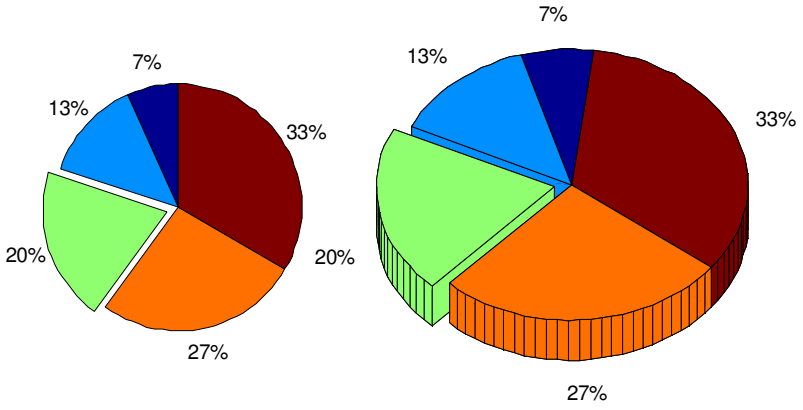
شکل (۲-۲۸) مقایسه خروجی دستور errorbar و دستور plot

۲-۸-۱۰ دستورpie و pie3: از این دستور برای رسم نمودارهای پای (pie) یا گراف‌ها وچارت‌های دایروی استفاده می شود. الگوهای نوشتاری این دستورات به صورت‌های زیر است:

```
pie(x)
pie(x,explode)
pie(x,explode,labels)
pie3(x)
pie3(x,explode,labels)
```

با استفاده از بردار explode می توان یک تکه یا چند تکه را از بقیه نمودار جدا کرده، و با استفاده labels می‌توان در کنار هر تکه متنی را به عنوان برچسب هر تکه نوشت.

```
pie([1,2,3,4,5],[0 0 1 0 0])
```



شکل (۲-۲۹) نمودار پای(pie)

❖ تمرینات تکمیلی

۱- نمودار توابع زیر را در یک گراف رسم نمائید.

$$x_1(t)=\frac{2+\sin(t)}{2-\cos\frac{t}{4}}e^{-0.05t} \qquad 0\le t\le 30$$

$$x_2(t)=\frac{2+\sin(t)}{2-\cos\frac{t}{4}}e^{-0.2t} \qquad 0\le t\le 30$$

۲- نمودار توابع غیرخطی زیر را رسم کنید.

$$x(t)=\left|\frac{\sin 50t}{t}\right|\cos(t+\pi)$$

$$y(t)=\left|\frac{\sin 50t}{t}\right|\sin(t+\pi)$$

$$-10\pi\le t\le 10\pi$$

سپس نمودارهای رسم شده را Hold کرده و تابع $z=10e^{z}$ را در صورتی‌که z از z = 0.01+0.5i تا z = - 1+50i تغییر کند، رسم کنید.

۳- تابع $y=\sin^4 x\cos x+e^{-|x|}\cos^4 x$ را با تابع ezplot رسم کنید.

۴- اگر $\varepsilon=1\times10^{-10}, -10\le x\le 10, -10\le y\le 10$، تابع $z(x,y)=\dfrac{\sin(\sqrt{x^2+y^2+\varepsilon})}{\sqrt{x^2+y^2+\varepsilon}}$ را رسم نمائید.

۵- تابع گسسته $x(n)=25\cos(\pi n+5)e^{0.1n}$ را در بازه $0\le n\le 40$ رسم کنید.

۶- معادله یک بیضی در سیستم مختصات قطبی به‌صورت $r=\dfrac{a(1-e^2)}{1-e\cos\theta}$ است. در صورتی‌که a=2 و e=0.5، در آن صورت نمودار r بر حسب θ را رسم کنید.

۷- نمودار نقاطی را که توسط معادلات اختلافی زیر حاصل می‌شود رسم کنید.

$$x_{k+1}=y_k(1+\sin(0.7x_k))-1.2\sqrt{|x_k|}$$

$$y_{k+1}=0.21-x_k$$

فصل سوم

چند جمله‌ای‌ها و محاسبات سیمبولیک در MATLAB

۳-۱ محاسبات ریاضی چندجمله‌ای‌ها

یک چندجمله‌ای در MATLAB به صورت یک بردار سطری است که مولفه‌های آن ضرایب چندجمله‌ای می باشند. این ضرایب به‌صورت نزولی مرتب شده اند. به‌عنوان مثال، چندجمله‌ای $A(s)=s^3+4s^2-7s-10$ در MATLAB به شکل زیر معرفی می‌گردد:

```
a = [1 4 -7 -10];
```

برای ارزیابی و رسم چندجمله‌ای $A(s)=s^3+4s^2-7s-10$ در بازه [3 -1]دستورات زیر را وارد کنید:

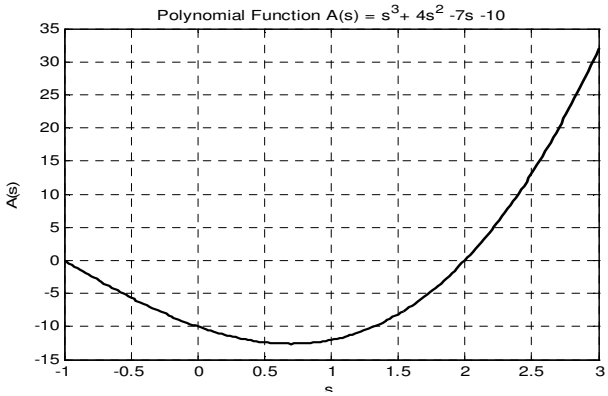
```
s = linspace(-1,3,201);
a = [1 4 -7 -10];
A = polyval(a,s);
```

تابع polyval مقدار چندجمله‌ای a را در نقاط s محاسبه می‌کند. الگوی نوشتاری این دستور به صورت زیر می باشد.

y = polyval(p,x)

که p ضرایب چندجمله‌ای و x نقاطی هستند که در چندجمله‌ای قرار می‌گیرند.

```
plot(s,A)
title('Polynomial Function A(s) = s^3+ 4s^2 -7s -10')
xlabel('s')
ylabel('A(s)')
```



شکل (۳-۱) رسم چندجمله‌ای $A(s)=s^3+4s^2-7s-10$

تابع polyval مقدار چندجمله‌ای را در هر نقطه حساب می‌کند. به‌عنوان مثال برای پیدا کردن مقدار چندجمله‌ای a در نقطه s=5 دستور زیر را می‌نویسیم:

```
>> polyval(a,5)
ans = 180
```

جمع و تفریق دو چند جمله‌ای a وb به‌راحتی قابل انجام است.

```
>> a = [1 -3 0 -1 2];
>> b = [0 4 -2 5 -16];
>> c = a + b
c =
  1  1  -2  4  -14
```

برای ضرب و تقسیم چندجمله‌ای‌ها از دستورات conv و deconv استفاده می‌شود. الگوی نوشتاری این دستورات به صورت‌های زیر می باشد.

```
w = conv(u,v)
[q,r] = deconv(v,u) %v = conv(u,q)+r
```

که u وv ضرایب چند جمله‌ای هستند.

$$w(k)=\sum_j u(j)v(k+1-j)$$

در مثال زیر a و b دو چندجمله‌ای هستند که با استفاده از دستور conv در یکدیگر ضرب شده‌اند.

```
>> a = [1 -3 0 -1 2];
>> b = [4 -2 5 -16];
>> c = conv(a,b)
c =
     4   -14    11   -35    58    -9    26   -32
%~~~~~~~~~~~~~~~~~~~~~~~~~
```

$$H(s)=\frac{N(s)}{D(s)}=\frac{s^3+5s^2+11s+13}{s^2+2s+4}$$

$$H(s)=Q(s)+\frac{R(s)}{D(s)} \Rightarrow H(s)=s+3+\frac{s+1}{s^2+2s+4}$$

```
>> n = [1 5 11 13];
>> d = [1 2 4];
>> q = deconv(n,d)
q =
     1     3
```

در مثال بالا دو چندجمله‌ای n وd بر یکدیگر تقسیم شده اند. برای محاسبه این تقسیم از دستور deconv استفاده شده است. در صورتی‌که آرگومان‌های خروجی دستور deconv دوتا باشند مقدار R,Q برگردانده می شوند در غیراین صورت اگر تنها یک آرگومان خروجی داشته باشیم، فقط مقدار Q برگردانده می‌شود.

```
>> [q r] = deconv(n,d)
q =
     1     3
r =
```

```
     0     0     1     1
```

ریشه‌های چند جمله‌ای $A(s)=s^3+4s^2-7s-10$ را می‌توان با استفاده از دستور roots به‌دست آورد:

```
>> a = [1 4 -7 -10];
>> r= roots(a)
r =
   -5.0000
    2.0000
   -1.0000
```

با دانستن ریشه‌های چند جمله‌ای می‌توان ضرایب چندجمله‌ای را با دستورpoly به‌دست آورد. الگوی نوشتاری این دستور به صورت‌های زیر می باشد. در صورتیکه A ماتریس باشد، p ضرایب چند جمله‌ای مشخصه ماتریس A خواهد بود.

p = poly(A)

$\det(\lambda I - A) = c_1\lambda^n + ... + c_n\lambda^n + c_{n+1}$

الگوریتم محاسبه به صورت زیر می باشد:

```
z = eig(A);
c = zeros(n+1,1); c(1) = 1;
for j = 1:n
    c(2:j+1) = c(2:j+1)-z(j)*c(1:j);
end
```

در صورتی‌که A بردار باشد، در آن صورت p ضرایب چند جمله‌ای خواهد بود که A ریشه های آن می‌باشد.

```
>> A= poly(r)
A =
    1.0000    4.0000   -7.0000  -10.0000
```

۲-۳ محاسبه مشتق و انتگرال چندجمله‌ای‌ها

مشتق چندجمله‌ای را می‌توان با استفاده از تابع polyder محاسبه کرد. الگوی نوشتاری این دستور به صورت های زیر می باشد، که در الگوی اول p ضرایب چند جمله‌ای است.

k = polyder(p)
k = polyder(a,b)
[q,d] = polyder(b,a)

```
>> s=polyder(a)
s =
     3     8    -7
```

مشتق حاصلضرب دو چندجمله‌ای با دستور polyder(a,b) محاسبه می‌شود:

```
>> a = [1 2];
>> b = [1 3];
>> dc = polyder(a,b)% der(A*B)
dc =
     2     5
```

در صورتی‌که تعداد آرگومان‌های خروجی تابع polyder برابر ۲ باشد، تابع polyder(a,b) مشتق حاصل تقسیم دو چندجمله‌ای را محاسبه می‌کند.

$$H(s)=\frac{s+2}{s+3} \Rightarrow \frac{dH(s)}{ds}=\frac{(s+3)-(s+2)}{(s+3)^2}=\frac{1}{(s+3)^2}=\frac{1}{s^2+6s+9}$$

```
>> b = [1 2];
>> a = [1 3];
>> [q,d] = polyder(b,a)
q =
      1
d =
     1     6     9
```

برای محاسبه انتگرال چندجمله‌ای از دستور polyint استفاده می‌شود. الگوی نوشتاری این دستور به صورت های زیر می‌باشد. که در آن p ضرایب چند جمله‌ای و k عدد ثابت انتگرال گیری می‌باشد. در صورتی که اشاره نشود، به صورت پیش فرض برابر صفر خواهد بود.

polyint(p,k)
polyint(p)

به طور مثال برای محاسبه انتگرال چند جمله ای a می نویسیم:

```
>>a=[1 5 18 24 39 7];
>>i=polyint(a)
i =
0.1667    1.0000    4.5000    8.0000   19.5000    7.0000         0
```

۳-۳ بسط به کسرهای جزئی با دستور residue

بسط به کسرهای جزئی با استفاده از دستور residue انجام می‌شود. الگوی نوشتاری این دستور به صورت‌های زیر می باشد:

[r,p,k] = residue(b,a)
[b,a] = residue(r,p,k)

که a,b ضرایب چندجمله‌ای های صورت و مخرج و p،r به ترتیب بردار ستونی باقیمانده، ریشه و k بردار سطری شامل ترم مستقیم یا خارج قسمت می باشد. در این میان ممکن است ریشه‌های مخرج حقیقی متمایز، تکراری، و یا مختلط باشند که در ادامه برای هر یک از این حالت‌ها یک مثال آورده شده است.

مثال۱: مخرج با ریشه‌های حقیقی مختلف

$$H(s)=\frac{c_1}{s-r_1}+\frac{c_2}{s-r_2}+.....+\frac{c_n}{s-r_n}$$

$$H(s)=\frac{s+2}{s^3+4s^2+3s} \Rightarrow H(s)=-\frac{1/6}{s+3}-\frac{1/2}{s+1}+\frac{2/3}{s}$$

```
>> b = [1 2];
>> a = [1 4 3 0];
```

```
>> [c,r] = residue(b,a)
c =
   -0.1667
   -0.5000
    0.6667
r =
    -3
    -1
     0
~~~~~~~~~~~~~~~~~~
```

مثال۲: مخرج با ریشه‌های مختلط

$$H(s)=\frac{s^2-2s+1}{s^3+3s^2+4s+2} \Rightarrow H(s)=\frac{-1.5+2j}{s+1-j}+\frac{-1.5-2j}{s+1+j}+\frac{4}{s+1}$$

```
>> b = [1 -2 1];
>> a = [1 3 4 2];
>> [c,r] = residue(b,a)
c =
  -1.5000 + 2.0000i
  -1.5000 - 2.0000i
   4.0000
r =
  -1.0000 + 1.0000i
  -1.0000 - 1.0000i
  -1.0000
~~~~~~~~~~~~~~~~~~
```

مثال ۳: مخرج با ریشه‌های تکراری

$$H(s)=\frac{c_1}{s-r_1}+\frac{c_2}{(s-r_1)^2}+\ldots+\frac{c_p}{(s-r_1)^p}+\frac{c_{p+1}}{s-r_{p+1}}+\ldots\frac{c_n}{s-r_n}$$

$$H(s)=\frac{5s-1}{s^3-3s-2} \Rightarrow H(s)=\frac{1}{s-2}-\frac{1}{s+1}+\frac{2}{(s+1)^2}$$

```
>> b = [5 -1];
>> a = [1 0 -3 -2];
>> [c,r] = residue(b,a)
c =
    1.0000
   -1.0000
    2.0000
r =
    2.0000
   -1.0000
   -1.0000
~~~~~~~~~~~~~~~~~~
```

مثال۴: توابع گویا که درجه صورت بیشتر از درجه مخرج می‌باشد.

$$H(s)=Q(s)+\frac{R(s)}{A(s)}$$

$$H(s)=\frac{s^3+2s-4}{s^2+4s-2}\Rightarrow H(s)=s-4+\frac{20.6145}{s+4.4495}-\frac{0.6145}{s-0.4495}$$

```
>> b = [1 0 2 -4];
>> a = [1 4 -2];
>>[c,r,q] = residue(b,a)
c =
   20.6145
   -0.6145
r =
   -4.4495
    0.4495
q =
     1    -4
```

۳-۴ محاسبات ریاضی سیمبولیک

در این مبحث، اجرای محاسبات سیمبولیک یا نمادین با استفاده از MATLAB بررسی می‌شود، به‌طوری‌که بتوان عملیات و محاسبات را به راحتی در این نرم افزار انجام داد. برای اجرای محاسبات سیمبولیک در ابتدا باید متغیرها و ثابت‌های مورد استفاده معرفی شوند. به‌عنوان مثال برای تعریف متغیرهای x,y یا هر متغیر دیگری، از دستور syms استفاده می‌کنیم:

```
>> syms x y
```

در صورتی‌که x,y اعداد حقیقی و z غیرحقیقی باشد آنها را به صورت زیر معرفی می‌نمائیم:

```
>> x = sym('x','real');
>> y = sym('y','real');
>> syms x y real
>> syms z unreal
```

برای تعریف ثوابت سیمبولیک نیز از sym استفاده می‌کنیم:

```
pi = sym(’pi’);
a = sym('alpha')
rho = sym('(1 + sqrt(5))/2')
f = rho^2 - rho - 1
sqroot2 = sym('sqrt(2)');
```

مزیت استفاده از ثابت‌های نمادین یا سیمبولیک در این است که دقت محاسباتی تا مرحله ارزیابی کاهش پیدا نخواهد کرد.

```
>>f = sym('f(x)')
>>df = (subs(f,'x','x+h') - f)/'h'
df =
(f(x+h)-f(x))/h
```

عملگرهای ^ / * - + همانند محاسبات عددی، در محاسبات سیمبولیک نیز مورد استفاده قرار می‌گیرند.

```
>> syms s t A
>> f = s^2 + 4*s + 5
f =
s^2+4*s+5
>> g = s + 2
g =
s+2
>> h = f*g
h =
(s^2+4*s+5)*(s+2)
>> z= exp(-s*t)
z =
exp(-s*t)
>> y = A*exp(-s*t)
y =
A*exp(-s*t)
```

در مثال بالا در ابتدا متغیرهای نمادین s,t,A معرفی شده، سپس از آنها در عملیات ریاضی استفاده شد تا متغیرهای نمادین f،g، z، y و h را خلق کنند. باید توجه داشت که متغیر x به عنوان متغیر مستقل پیش فرض می‌باشد. بقیه متغیرها نیز می‌توانند به‌عنوان متغیر پیش فرض درنظرگرفته شوند. دستوری که برای پیدا کردن متغیرهای مستقل استفاده می‌شود، دستور findsym می‌باشد.

```
findsym(S)
>> findsym(f)
ans =
s
>> findsym(y)
ans =
A, s, t
```

حتی می‌توان ماتریس‌های سیمبولیک نیز ایجاد کرد:

```
>> n = 3;
>> syms x
>> B = x.^((0:n)'*(0:n))
B =
[ 1, 1, 1, 1]
[ 1, x, x^2, x^3]
[ 1, x^2, x^4, x^6]
[ 1, x^3, x^6, x^9]
```

عملیات جبری به سادگی در قسمت جعبه ابزار سیمبولیک امکان پذیر است. برای انجام این محاسبات جبری از دستورات زیر استفاده می‌کنیم که در ادامه به توضیح هریک می‌پردازیم.

expand(S), factor(S), simplify(S), [n,d] = numden(S), subs(S,old,new)

۱- **عملیات جمع و تفریق سیمبولیک:** عملیات جمع و تفریق سیمبولیک به راحتی در MATLAB قابل انجام است و نیازی به دستوری خاص ندارد.

```
>> syms s
>> A = s^4 -3*s^3 -s +2;
>> B = 4*s^3 -2*s^2 +5*s -16;
>> C = A + B
C =
s^4+s^3+4*s-14-2*s^2
```

۲- **عملیات ضرب سیمبولیک:** ازدستور expand برای بسط سیمبولیک چندجمله ای ها استفاده می‌شود.

```
>> syms s
>> A = s+2;
>> B = s+3;
>> C = A*B
C =
(s+2)*(s+3)
>> C = expand(C)
C =
s^2+5*s+6
```

۳- **عملیات فاکتورگیری:** برای فاکتورگیری از عبارات سیمبولیک از دستور factor استفاده می‌شود.

```
>> syms s
>> D = s^2 + 6*s + 9;
>> D = factor(D)
D =
(s+3)^2
>> P = s^3 - 2*s^2 -3*s + 10;
>> P = factor(P)
P =
(s+2)*(s^2-4*s+5)
```

۴- **مخرج مشترک گرفتن،** در عبارت زیر با استفاده از دستور numden صورت و مخرج کسر حاصل از عملیات ریاضی کسرها را به‌دست می‌آوریم. الگوی نوشتاری این دستور به صورت زیر است:

```
[N,D] = numden(A)
```

$$H(s) = -\frac{1/6}{s+3} - \frac{1/2}{s+1} + \frac{2/3}{s}$$

```
>> syms s
>> H = -(1/6)/(s+3) -(1/2)/(s+1) +(2/3)/s;
>> [N,D] = numden(H)
N =
s+2
D =
(s+3)*(s+1)*s
>> D = expand(D)
```

```
D =
s^3+4*s^2+3*s
```

پاسخ به صورت $H(s)=\frac{s+2}{s^3+4s^2+3s}$ خواهد بود. N صورت کسر و D مخرج کسر نهایی می‌باشند.

۵- **حذف جملات و عبارتهای یکسان از صورت و مخرج**: عبارت کسری زیر را در نظر بگیرید. برای ساده سازی کسر از دستور simplify استفاده می‌شود.

$$H(s)=\frac{s^3+2s^2+5s+10}{s^2+5}=\frac{(s+2)(s^2+5)}{s^2+5}=s+2$$

```
>> syms s
>> H = (s^3 +2*s^2 +5*s +10)/(s^2 + 5);
>> H = simplify(H)
H =
s+2
>> factor(s^3 +2*s^2 +5*s +10)
ans =
(s+2)*(s^2+5)
```

۶- **جایگزینی متغیرها و عبارت‌ها**: نسبت بین دو چندجمله‌ای زیر را در نظر بگیرید. برای جایگزینی مقادیر به جای متغیر s می توان از دستور subs استفاده کرد. الگوی نوشتاری این دستور به صورت زیر است:

```
R = subs(S,old,new)
```

که new عدد یا متغیرجدید و old متغیری است که قرار است old جایگزین آن شود.

$$H(s)=\frac{s+3}{s^2+6s+8}\Rightarrow G(s)=H(s)\,|_{s=s+2}=\frac{s+5}{s^2+10s+24}$$

```
>> syms s
>> H = (s+3)/(s^2 +6*s + 8);
>> G = subs(H,s,s+2)
G =
(s+5)/((s+2)^2+6*s+20)
>> G = collect(G)
G =
(s+5)/(s^2+10*s+24)
```

محاسبه معکوس تابع: برای محاسبه معکوس یک تابع از دستور finverse استفاده می‌شود. الگوی نوشتاری این دستور به صورت زیر می باشد.

```
g = finverse(f)
g = finverse(f,v)
```

که v متغیر سیمبولیک مستقل می‌باشد. در حالت اول g(f(x))=x و در حالت دوم g(f(v))=v است.

```
>>syms x u v
>>finverse(1/tan(x))
ans =
```

```
atan(1/x)

>>finverse(exp(u-2*v),u)
ans =
2*v+log(u)
```

۷- **ترکیب توابع**: برای ترکیب توابع در MATLAB از دستور compose استفاده می‌کنیم. الگوی نوشتاری این دستور به صورت‌های زیر می باشد.

```
compose(f,g)
compose(f,g,z)
compose(f,g,x,z)
compose(f,g,x,y,z)
```

در حالت اول f=f(x) و g=g(y) و خروجی برابر f(g(y)) است. در حالت دوم خروجی برابر f(g(z)) و در حالت‌های بعدی x متغیر مستقل تابع f و y متغیر مستقل g است.

```
syms x y z t u;
f = 1/(1 + x^2);
g = sin(y);
h = x^t;
p = exp(-y/u);

compose(f,g)        ->  1/(1+sin(y)^2)
compose(f,g,t)      ->  1/(1+sin(t)^2)
compose(h,g,x,z)    ->  sin(z)^t
compose(h,g,t,z)    ->  x^sin(z)
compose(h,p,x,y,z) ->  exp(-z/u)^t
compose(h,p,t,u,z) ->  x^exp(-y/z)
```

در پردازش سیمبولیک می‌توان مشتق یک بردار ستونی را نسبت به یک بردار سطری با استفاده از دستور jacobian انجام داد. الگوی نوشتاری این دستور به صورت زیر است:

```
jacobian(f,v)
```

که حاصل این عبارت مشتق عدد یا بردار f نسبت به v است.

```
>>syms r l f
>>x = r*cos(l)*cos(f); y = r*cos(l)*sin(f); z = r*sin(l);
>>J = jacobian([x; y; z], [r l f])
>>detJ = simple(det(J))
J =
[    cos(l)*cos(f), -r*sin(l)*cos(f), -r*cos(l)*sin(f)]
[    cos(l)*sin(f), -r*sin(l)*sin(f),  r*cos(l)*cos(f)]
[          sin(l),         r*cos(l),                0]

detJ =
-cos(l)*r^2
```

برای تبدیل نمایش چندجمله‌ای‌ها ازحالت عددی به حالت نمادین و بالعکس می‌توان از دستورات زیـر اسـتفاده کرد:

sym2poly(P), poly2sym(p,s)

به‌عنوان مثال چندجمله‌ای $A(s)=s^3+4s^2-7s-10$ را در نظر بگیرید که از حالت عددی به حالت نمـادین تبدیل می‌شود و به‌جای متغیر نمادین، s قرار می‌گیرد. البته می‌توان متغیرهای دیگری نیز جایگزین آن نمود.

```
>> a = [1 4 -7 -10];
>> A = poly2sym(a,s)
A =
s^3+4*s^2-7*s-10

>> syms s
>> B = 4*s^3 -2*s^2 +5*s -16;
>> b = sym2poly(B)
b =
4 -2 5 -16
```

محاسبات مثلثاتی را نیز می‌توان به صورت سیمبولیک در MATLAB انجام داد. به‌طور مثال:

```
>> syms theta phi
>> A = sin(theta + phi)
>> A = expand(A)
A =
sin(theta)*cos(phi)+cos(theta)*sin(phi)
```

در بسیاری از کاربردها می‌خواهیم نتیجه عددی یا نموداری از محاسبات سیمبولیک به‌دست آوریم. برای انجام این کار از دستور double استفاده می‌کنیم.

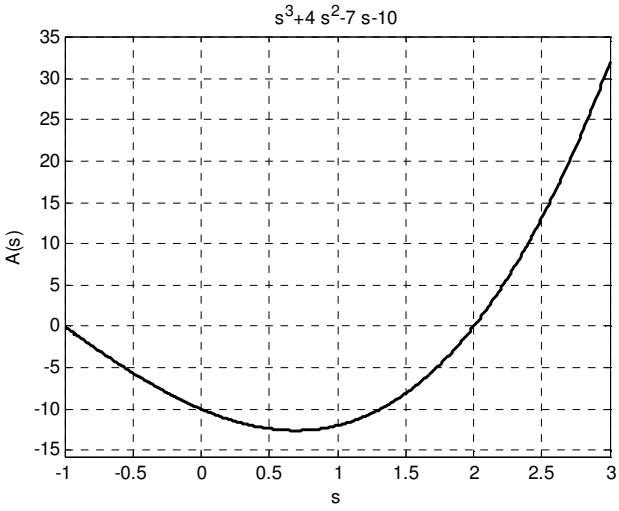
```
>> E = s^3 -14*s^2 +65*s -100;
>> F = subs(E,s,7.1)
F =
13671/1000
>> G = double(F)
G =
13.6710
```

عبارات سیمبولیک را می‌توان با دستور ezplot رسم کرد. فرمت این دستور به‌صورت زیر می‌باشد:

ezplot(f), ezplot(f,xmin,xmax)

به‌عنوان مثال چند جمله‌ای $A(s)=s^3+4s^2-7s-10$ را در بازه $[-1\ 3]$ رسم می‌کنیم:

```
syms s
a = [1 4 -7 -10];
A = poly2sym(a,s)
ezplot(A,-1,3), ylabel('A(s)')
```



شکل (۳-۲) رسم چندجمله‌ای‌ها با استفاده از ezplot

۳-۵ حل معادلات جبری با استفاده از دستور solve

جعبه ابزار سیمبولیک برای حل معادلات جبری و غیرجبری از دستور Solve استفاده می‌کند. فرمت استفاده از این دستور به صورت زیر می‌باشد:

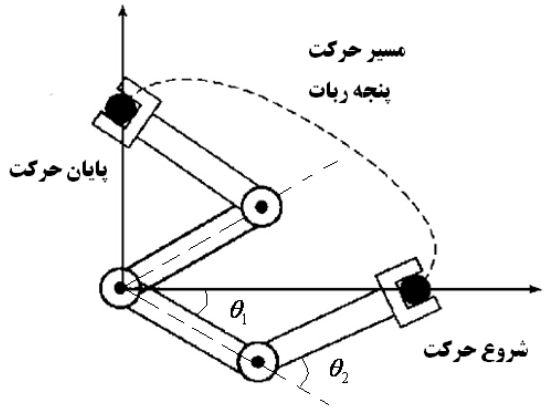
solve(E1, E2,...,EN)
solve(E1, E2,...,EN, var1, var2,...,varN)

که در آن E1، E2،...و EN عبارات سیمبولیک و var1، var2،...و varN متغیرهای مورد استفاده در عبارت‌های سیمبولیک می‌باشند که در ابتدا معرفی شده اند. جوابهای به‌دست آمده ریشه‌های معادلات تحت شرایط E1 = 0، E2 = 0، . . . و EN = 0 هستند.

```
>> syms s
>> P = s^3 -2*s^2 -3*s + 10;
>> s = solve(P)
s =
[ -2]
[ 2+i]
[ 2-i]
~~~~~~~~~~~~~~~~~
>> syms theta x z
>> E = z*cos(theta) - x;
>> theta = solve(E,theta)
theta =
acos(x/z)
~~~~~~~~~~~~~~~~~~
>> syms x
```

```
>> E = exp(2*x) + 4*exp(x) -32;
>> x = solve(E)
x =
[ log(-8)]
[ log(4)]
~~~~~~~~~~~~~~~~~~
>>solve('a*x^2 + b*x + c')
ans =
 -1/2*(b-(b^2-4*a*c)^(1/2))/a
 -1/2*(b+(b^2-4*a*c)^(1/2))/a
~~~~~~~~~~~~~~~~~~
>>solve('a*x^2 + b*x + c','b')
ans =
-(a*x^2+c)/x
~~~~~~~~~~~~~~~~~~
>>S = solve('x + y  = 1','x - 11*y = 5')
>> S.x
ans =4/3
>> S.y
ans = -1/3
~~~~~~~~~~~~~~~~~~
>> A = solve('a*u^2 + v^2', 'u - v = 1', 'a^2 - 5*a + 6');
>> a2 = [A.a(2), A.u(2), A.v(2)]
~~~~~~~~~~~~~~~~~~
>>syms theta
>>E = cos(2*theta)-sin(theta);
>> solve(E,theta)
→ ans =
 -1/2*pi
  1/6*pi
  5/6*pi
```

مساله حرکت روبات دو درجه آزادی را در نظر بگیرید. برای به‌دست آوردن زوایای اولیه بازوهای ربات در صورتیکه موقعیت نوک پنجه ربات $(x, y) = (6.5, 0)$ باشد، دستورات زیر را وارد می‌کنیم:



شکل (۳-۳) روبات دولینکی

```
>> syms theta1 theta2
>> E1 = 4*cos(theta1)+3*cos(theta1+theta2)-6.5;
>> E2 = 4*sin(theta1)+3*sin(theta1+theta2);
>> [theta1, theta2] = solve(E1,E2)
theta1 =
[ atan(9/197*55^(1/2))]
[ atan(-9/197*55^(1/2))]
theta2 =
[ -atan(3/23*55^(1/2))]
[ -atan(-3/23*55^(1/2))]
>> theta1 = double(theta1*(180/pi))
>> theta2 = double(theta2*(180/pi))
theta1 =
18.7170
-18.7170
theta2 =
-44.0486
44.0486
```

که جواب‌های مورد قبول $\theta_1 = -18.7^0, \theta_2 = 44^0$ می‌باشند.

۳-۶ مشتق‌گیری و انتگرال‌گیری از توابع سیمبولیک

از عبارات سیمبولیک می‌توان مشتق‌گیری و انتگرال‌گیری کرد. برای انجام **عملیات مشتق‌گیری** از دستورات زیر استفاده می‌کنیم:

diff(E), diff(E,v), diff(E,n), diff(E,v,n)= $\frac{d^n E}{d^n v}$

که درآن، E، عبارت سیمبولیک، v، متغیر سیمبولیک که مشتق‌گیری نسبت به آن انجام می‌شود، و n، مشتق مرتبه n اُم از عبارت نمادین E می‌باشد. پس می‌توان عبارت آخر را به این‌صورت بیان کرد: "مشتق مرتبه n اُم تابع E نسبت به متغیر v ".

```
>> p = s^3 + 4*s^2 -7*s -10;
>> d = diff(p)
>> e = diff(p,2)
```

$(u(t))_{t=3} = \text{vpa('Heaviside(3)')}$

$u'(t) = \text{diff('Heaviside(t)')}$

برای انجام عملیات انتگرال‌گیری از عبارات سیمبولیک از دستورات زیر استفاده می‌کنیم:

$\text{int(E), int(E,v), int(E,a,b)} = \int_{x=a}^{x=b} E(x)dx$, $\text{int(E,v,a,b)} = \int_{x=a}^{x=b} E(v)dv$

که در آن، E، عبارت سیمبولیک، v، متغیر سیمبولیک که انتگرال‌گیری نسبت به آن انجام می‌شود. a و b نیز حد بالا و پائین انتگرال‌گیری می‌باشند.

```
>> syms x n a b t

>> int(x^3 +4*x^2 + 7*x + 10)
ans =
1/4*x^4+4/3*x^3+7/2*x^2+10*x

>> int(x^3,a,b)
ans =
1/4*b^4-1/4*a^4
```

انتگرال تابع $\int_{-\infty}^{+\infty} e^{-(kx)^2} dx$ به صورت زیر محاسبه می‌شود.

```
>> x=sym ('x')
>> syms  k real;
>> f = exp(-(k*x)^2);
>> int(f,x,-inf,inf)
ans =
signum(k)/k*pi^(1/2)
```

مثال: انتگرال توابع زیر را به‌دست آورید.

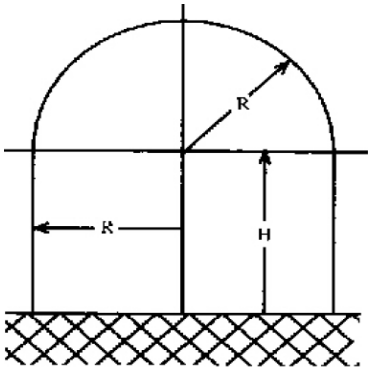
$\int_{-\infty}^{\infty} \delta(t)dt = \text{int('Dirac(t)'}-\text{inf,inf)}$

$\int_{-2}^{3} u(t)dt = \text{int('Heaviside(t)',}-2,3)$

$\int_{-\infty}^{\infty} \text{sign(x)dx} = \text{int('signum(t)',}-\text{inf,inf)}$

$$\int_{-1}^{2} t \times \text{sign}(t)dt = \text{int}('\text{signum}(t)'\times t, -1, 2)$$

مثال: مخزن آبی در شکل (۳-۴) نشان داده شده است. قسمت پائین این مخزن استوانه‌ای و قسمت بالایی آن کروی می‌باشد. شعاع قسمت استوانه‌ای مخزن برابر R و ارتفاع آن برابر H است. مخزن حداکثر ۵۰۰ مترمکعب آب را در خود ذخیره می‌کند. حال می‌خواهیم مقادیر R,H بهینه را طوری پیدا کنیم که هزینه ساخت حداقل شود. از طرفی می‌دانیم هزینه ورق قسمت استوانه $300\$/m^2$ و هزینه قسمت کروی $400\$/m^2$ است.



شکل (۳-۴) مخزن آب

حجم کل مخزن برابر است با:

$$V = \pi R^2 H + \frac{2}{3}\pi R^3 = 500$$

و هزینه ساخت مخزن برابر است با:

$$C = 300(2\pi RH) + 400(2\pi R^2)$$

برای حل این مساله دستورات زیر را وارد می‌کنیم:

```
>> syms R H
>> V = pi*R^2*H + (2/3)*pi*R^3 -500; % Equation for volume
>> H = solve(V,H) % Solve volume for height H
H =
-2/3*(pi*R^3-750)/pi/R^2
```

از معادله حجم، مقدار H را برحسب R به‌دست آورده و در معادله هزینه جایگزین می‌کنیم تا تنها یک متغیر داشته باشیم.

```
>> C = 300*(2*pi*R*H) + 400*(2*pi*R^2); % Equation for cost
>> dCdR = diff(C,R) % Derivative of cost wrt R
dCdR =
400/R^2*(pi*R^3-750)+400*pi*R
```

از معادله هزینه نسبت به R مشتق گرفته و برابر صفر قرار داده و مقدار R مینیمم را به‌دست می‌آوریم.

```
>> Rmins = solve(dCdR,R) % Solve dC/dR for R: Rmin
Rmins =
```

```
[ 5/pi*3^(1/3)*(pi^2)^(1/3)]
[ -5/2/pi*3^(1/3)*(pi^2)^(1/3)+5/2*i*3^(5/6)/pi*(pi^2)^(1/3)]
[ -5/2/pi*3^(1/3)*(pi^2)^(1/3)-5/2*i*3^(5/6)/pi*(pi^2)^(1/3)]
>> Rmins = double(Rmins)
Rmin =
4.9237
-2.4619+ 4.2641i
-2.4619- 4.2641i
>> Rmin = Rmins(1)
Rmin =
4.9237
```

در معادله H برحسب R مقدار R مینیمم را قرار می‌دهیم تا H مینیمم نیز به‌دست آید. سپس با جایگذاری R,H مینیمم در تابع هزینه، مقدار هزینه مینیمم نیز به‌دست می‌آید.

```
>> Hmin = double(subs(H,R,Rmin))
Hmin =
3.2825
>> Cmin = double(subs(C,{R,H},{Rmin,Hmin}))
Cmin =
9.1394e+004
```

نکته: عملیات ریاضی روی ماتریس‌های سیمبولیک همانند ماتریس‌های عددی می‌باشد.

```
>> A = sym([2,1; 4,3])
A =
[ 2, 1]
[ 4, 3]
>> Ainv = inv(A)
Ainv =
[ 3/2, -1/2]
[ -2, 1]
>> C = A*Ainv
C =
[ 1, 0]
[ 0, 1]
```

۳-۷ محاسبه حد و مجانب‌های افقی و قائم

برای محاسبه حد از دستور limit استفاده می‌کنیم که فرمت نوشتاری آن به صورت زیر می‌باشد:

$$\lim_{x \to a} f(x) = \text{Limit(f,x,a)}$$

```
>> syms x
>> f= (1+1/x)^x
>> limit(f,x,inf) → exp(1)
```

مقدار تابع f وقتی x به سمت بینهایت میل می‌کند برابر e خواهد بود.

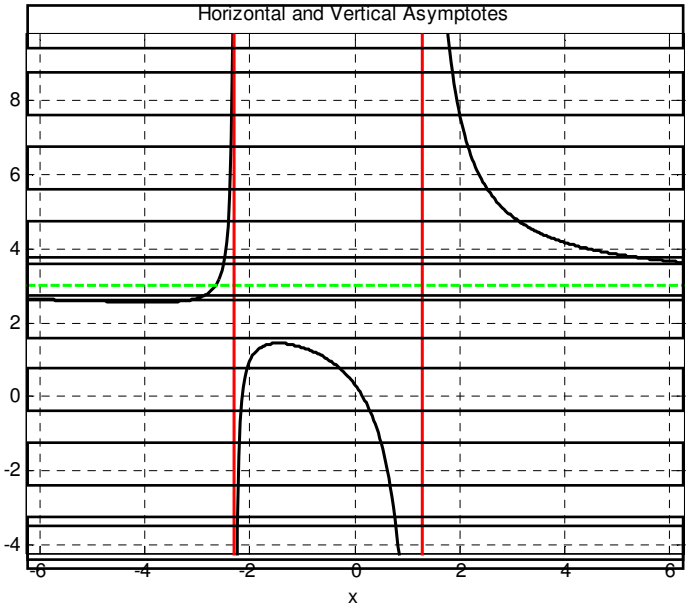
```
>> limit(1/x,x,0,'right')→ inf
```

حد راست تابع

```
>> Limit(1/x,x,0,'left')→ -inf
```
حد چپ تابع

مثال: نمودار تابع $f(x)=\dfrac{3x^2+6x-1}{x^2+x-3}$ را همراه با مجانب‌های افقی و قائم آن رسم کنید.

```
%1-Defining the Function
syms x
num = 3*x^2 + 6*x -1;
den = x^2 + x - 3;
f = num/den
ezplot(f)
% 2-Finding the Asymptotes
g = double(limit(f,inf))
roots = solve(den)
ezplot(f)
hold on
% Plot horizontal asymptote
plot([-2*pi 2*pi], [g g],'g')
% Plot vertical asymptotes
plot(double(roots(1))*[1 1], [-5 10],'r')
plot(double(roots(2))*[1 1], [-5 10],'r')
title('Horizontal and Vertical Asymptotes')
```



شکل (۳-۵) رسم مجانب‌های افقی و قائم

۸-۳ سری تیلور و دنباله‌ها

برای محاسبه مجموع یک دنباله از دستورsymsum استفاده می‌کنیم. الگوی نوشتاری این دستور به صورت‌های زیر است:

```
r = symsum(s)
r = symsum(s,v)
r = symsum(s,a,b)
r = symsum(s,v,a,b)
```

که s عبارت سیمبولیک، k وvمتغیر سیمبولیک، aو b محدوده v را مشخص می کنند یعنی v=a تا v=b

```
>> syms k
>> symsum(k^2,0,10) % ∑_{k=0}^{10} k^2
ans =
385
>>symsum(x^k/sym('k!'), k, 0,inf)
exp(x)
```

در MATLAB، برای محاسبه سری تیلور از دستور taylor استفاده می‌کنیم. دستور taylor(f) نیز سری مک لورن مرتبه ۵ تابع f را محاسبه می‌کند. دستور taylor(f,n) سری مک لورن مرتبه n-1 را محاسبه می کند و دستور taylor(f,n,a) نیز سری تیلور حول نقطه x=a (عدد صحیح)را تا درجه n-1 محاسبه می‌نماید.

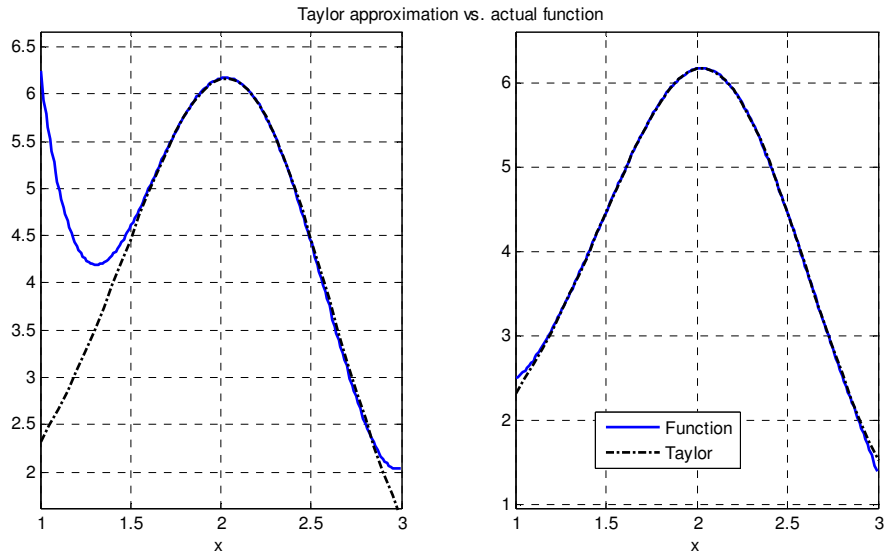
$$\text{Taylor}(f,b,a)=\sum_{n=0}^{b}(x-a)^n\frac{f^n(a)}{n!}$$

```
syms x
taylor(exp(-x))
ans =
1-x+1/2*x^2-1/6*x^3+1/24*x^4-1/120*x^5
~~~~~~~~~~~~~~~~~~~~
f =1/(5+4*cos(x))
B= taylor(f,5)
B =
 1/9+2/81*x^2+5/1458*x^4
~~~~~~~~~~~~~~~~~~~~
>> taylor(sin(x),4,2)
ans =
sin(2)+cos(2)*(x-2)-1/2*sin(2)*(x-2)^2-1/6*cos(2)*(x-2)^3
```

برای مقایسه بسط تیلور یک تابع تا درجه ۱۱ و درجه ۴ با خود تابع واقعی دستورات زیر را وارد کنید. از روی شکل می توان نتیجه گرفت که هر چه تعداد جملات بیشتر شود مقدار دقیق تابع محاسبه خواهد شد.

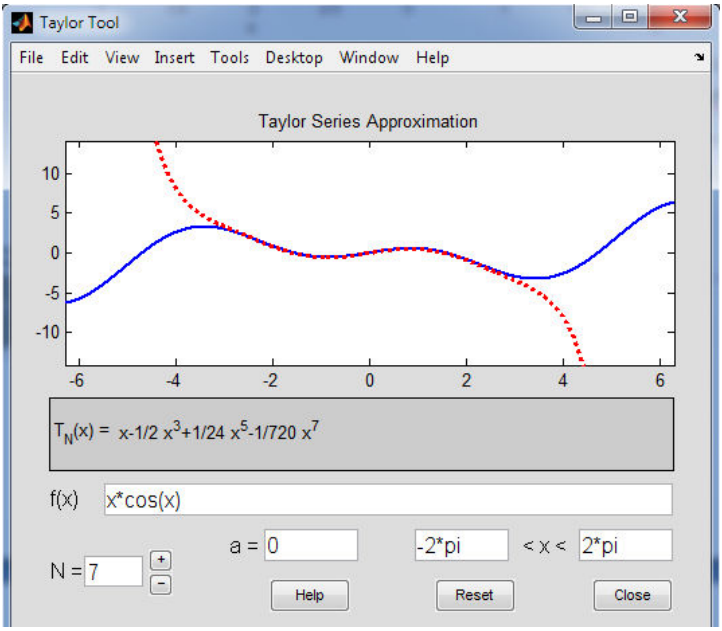
```
syms x
g = exp(x*sin(x));
t1 = taylor(g,5,2);
```

```
t2 = taylor(g,12,2);

xd = 1:0.05:3;
yd = subs(g,x,xd);

subplot(1,2,1)
ezplot(t1, [1,3]);
hold on
plot(xd, yd, 'r-.')

subplot(1,2,2)
ezplot(t2, [1,3]);
hold on
plot(xd, yd, 'r-.')

title('Taylor approximation vs. actual function');
legend('Function','Taylor')
```



شکل (۳-۵) مقایسه تابع اصلی با سری تیلور تابع

برای مقایسه سری با تابع اصلی می توانید از ابزار گرافیکی تیلور استفاده کنید. برای فراخوانی این ابزار در پنجره دستورات، taylortool را تایپ کنید و بعد از اجرا منتظر بمانید تا این پنجره باز شود. در این پنجره می توانید تابع مورد نظر، محدود رسم تابع، نقطه x=a و تعداد جملات را وارد نموده و نتیجه را در همان پنجره مشاهده کنید. هر بار تعداد جملات را تغییر دهید تا مقدار تقریبی تابع را مشاهده کنید.



شکل (۳-۶) ابزار گرافیکی برای رسم و نمایش سری تیلور

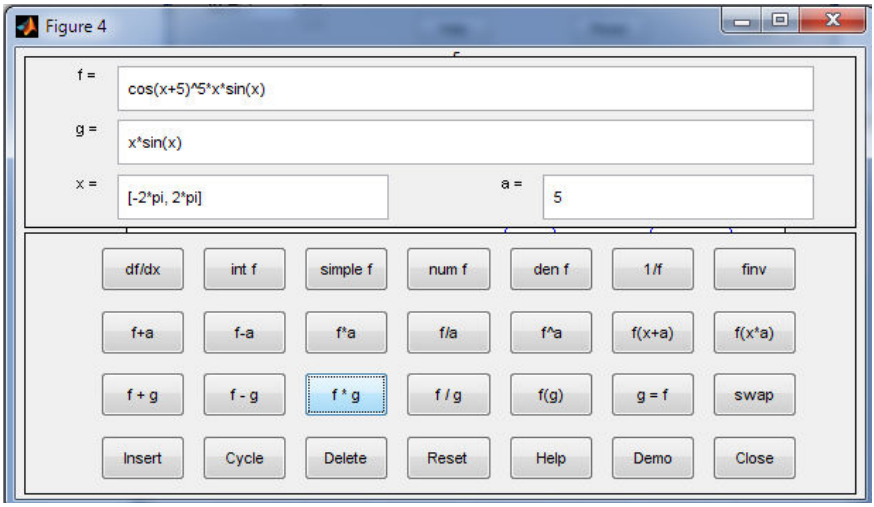
نکته: برای نمایش کد C عبارات سیمبولیک، به‌طور مثال خروجی تیلور تابع x*cos(x) از دستور زیر استفاده می‌کنیم:

```
syms x
f = taylor(x*cos(x));
ccode(f)
```

یکی دیگر از ابزار گرافیکی MATLAB ماشین حساب توابع است. برای استفاده از این ابزار فرمان funtool را در پنجره فرامین تایپ کنید تا این پنجره باز شود.

جدول (۳-۱) توابع موجود در funtool

تعویض جای صورت و مخرج	1/f	محاسبه مشتق تابع	df/dx
معکوس تابع	Finv	محاسبه انتگرال تابع	int f
f(x) + a.	f+a	ساده سازی در صورت امکان	simple f
f(x) ^ a	f^a	استخراج صورت کسر	num f
f(x * a)	f(x*a)	استخراج مخرج کسر	den f
f(x*a)	f(x*a)	f(x + a)	f(x+a)



شکل (۳-۷) ابزار گرافیکی برای محاسبات سیمبولیک روی توابع

۳-۹ حل معادلات دیفرانسیل با استفاده از دستور dsolve

برای حل معادلات دیفرانسیل معمولی به صورت سیمبولیک از دستورات dsolve استفاده می‌کنیم. فرمت نوشتاری این دستور به صورت زیر می‌باشد:

dsolve('eq1,eq2,...', 'cond1,cond2,...', 'v')

که در آن eq2،eq1، ... معادلات دیفرانسیل و v، متغیر مستقل، و cond2،'cond1'،... شرایط مرزی حاکم می‌باشند. بطور مثال برای حل معادله دیفرانسیل $\frac{dy}{dx}=1+y^2$ با شرط مرزی $y(0)=1$ دستورات زیر را وارد می‌کنیم:

```
>> dsolve('Dy=1+y^2','y(0)=1')
ans =
tan(t+1/4*pi)
```

مثال: معادلات دیفرانسیل زیر را حل کنید.

```
>> dsolve('Df = f + sin(t)')
ans =
-1/2*cos(t)-1/2*sin(t)+exp(t)*C1
~~~~~~~~~~~~~~~~~~~~~~
>> dsolve('D2y = -a^2*y', 'y(0) = 1', 'Dy(pi/a) = 0')
ans =
cos(a*t
~~~~~~~~~~~~~~~~~~~~~~
>> dsolve('(Dy)^2 + y^2 = 1','s')
ans =
          1
         -1
```

```
  sin(s-C1)
 -sin(s-C1)
~~~~~~~~~~~~~~~~~~~~~
>> [f,g]=dsolve('Df=3*f+5*g','Dg=-4*f+5*g','f(0)=1,g(0)=7')
f =
exp(4*t)*(34/19*sin(19^(1/2)*t)*19^(1/2)+cos(19^(1/2)*t))
g =
1/5*exp(4*t)*(15/19*sin(19^(1/2)*t)*19^(1/2)+35*cos(19^(1/2)*
t))
~~~~~~~~~~~~~~~~~~~~~
```

$$y^{(4)}+5y''+4y=0 \qquad y(0)=1, y'(0)=-1, y''(0)=1, y^{(3)}(0)=0$$

$$y(\pi)=?$$

```
>>Dsolve('D4y+5*D2y+4*y=0','Dy(0)=-
1','D2y(0)=1','D3y(0)=0','y(0)=1')
ans =
-4/3*sin(t)+5/3*cos(t)+1/6*sin(2*t)-2/3*cos(2*t)
>> syms t
>> subs(f,t,'pi')
ans =-7/3
~~~~~~~~~~~~~~~~~~~~~
```

$$y''-2y'+y=xe^{x}+4\ ,\quad y(0)=1, y'(0)=1$$

```
>> dsolve('D2y-2*Dy+y=x*exp(x)+4','y(0)=1','Dy(0)=1')
ans =
exp(t)*(-3-x*exp(x))+exp(t)*t*(x*exp(x)+4)+x*exp(x)+4
```

در این قسمت به علت اینکه متغیر پیش فرض برنامه t است، جواب بر حسب x,t به‌دست می‌آید. بنابراین متغیر مستقل را تغییر می‌دهیم.

```
>> dsolve('D2y-2*Dy+y=x*exp(x)+4','y(0)=1','Dy(0)=1','x')
ans =
-3*exp(x)+4*x*exp(x)+1/6*x^3*exp(x)+4
```

۳-۱۰ تبدیل لاپلاس و عکس تبدیل لاپلاس

تبدیل لاپلاس یکی از روشهای حل معادلات دیفرانسیل است که در حل معدلات دیفرانسیل معمولی و جزیی کاربرد فروانی دارد. برای لاپلاس گرفتن از دستور laplace استفاده می شود. برای استفاده از این دستور ابتدا باید متغیر مستقل را تعریف کرد.

syms t

البته هر متغیر دلخواه دیگری را نیز می‌توان تعریف کرد و تابع برگشتی هم بر حسب s می باشد.

r = laplace(t)

$$\text{Laplace}[\sin(t)\times t]$$

```
>> syms t
>> f=sin(t)*t;
>> laplace(f)
```

```
ans =
2*s/(s^2+1)^2
~~~~~~~~~~~~~~~~~
```

$$\text{Laplace}[e^{3t}(2\cos 5t - 3\sin 5t)]$$

```
>>syms t
>>f=exp(3*t)*(2*cos(5*t)-3*sin(5*t))
>>laplace(f)
ans =
(2*s-21)/(s^2-6*s+34)
~~~~~~~~~~~~~~~~~
```

$$\text{Laplace}[\int (t-\lambda)^3(\cos\lambda + \sin\lambda)d\lambda]$$

```
>> syms t landa
>> f=int((t-landa)^3*(cos(landa)+sin(landa)),landa,0,t)
f =
-6+t^3+3*t^2-6*t+6*cos(t)+6*sin(t)
~~~~~~~~~~~~~~~~~
```

$$\text{Laplace}[(u(t)-u(t-\pi))\sin t]$$

```
>>f=(heaviside(t)-heaviside(t-pi))*sin(t)
>>laplace(f)
ans =
(1+exp(-s*pi))/(s^2+1)
```

برای لاپلاس معکوس گرفتن هم از دستور ilaplace استفاده می شود.

ilaplace(r)

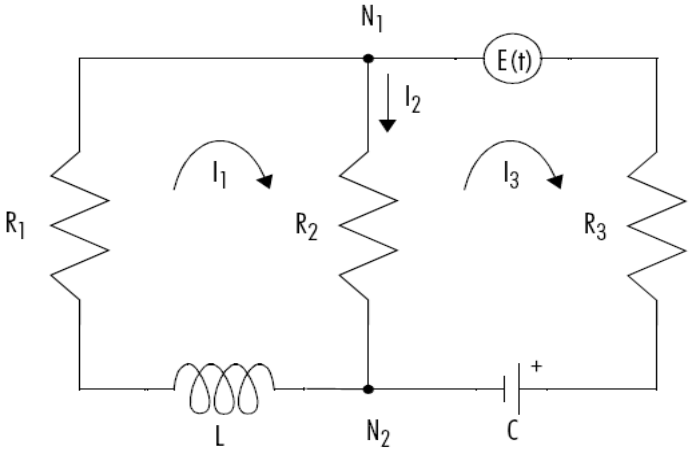
$$F(s) = \frac{2e^{-2s}}{s^2-4}$$

```
>>syms s
>>F=(2*exp(-2*s))/(s^2-4)
>>ilaplace(F)
ans =
heaviside(t-2)*sinh(2*t-4)
~~~~~~~~~~~~~~~~~
```

$$F(s) = (e^{-2\pi s}\frac{s}{(s^2+1)^2})$$

```
>>syms s
>>f=(exp(-2*pi*s)*s)/((s^2+1)^2)
>>ilaplace(f)
ans =
1/2*heaviside(t-2*pi)*(t-2*pi)*sin(t)
```

تبدیل لاپلاس کاربردهای بسیاری در حل مسایل مهندسی دارد از جمله می‌توان به حل معادلات ODE و حل مسایل با مقادیر اولیه اشاره کرد. به طور مثال یک مدار RLC همانند شکل زیر را در نظر بگیرید. با استفاده از قوانین کیرشهف و قانون جریان معادلات دیفرانسیل حاکم بر مدار RLC را به‌دست آورید.



شکل (۳-۸) مدار RLC

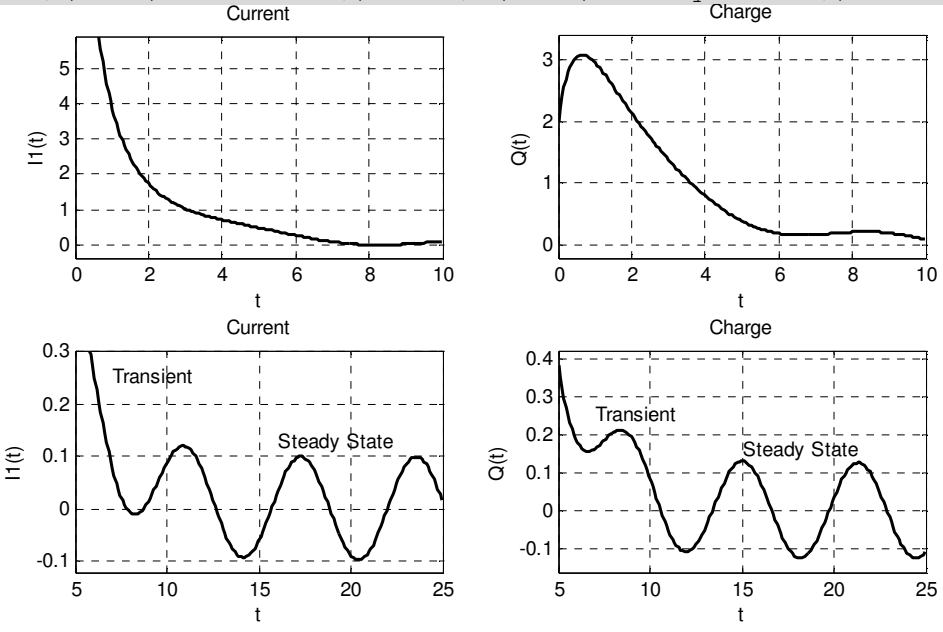
$$\frac{dI_1}{dt}+\frac{R_2}{L}\frac{dQ}{dt}=\frac{R_1-R_2}{L}I_1,\ I_1(0)=I_{10}$$

$$\frac{dQ}{dt}=\frac{1}{R_3+R_2}(E(t)-\frac{1}{C}Q(t))+\frac{R_2}{R_2+R_3}I_1,\ Q(0)=Q_0$$

برنامه زیر را در یک m-file وارد کنید:

```
syms R1 R2 R3 L C real
dI1 = sym('diff(I1(t),t)');
dQ = sym('diff(Q(t),t)');
I1 = sym('I1(t)');
Q = sym('Q(t)');
syms t s
E = sin(t); % Voltage
eq1 = dI1 + R2*dQ/L -(R2 - R1)*I1/L;
eq2 = dQ - (E - Q/C)/(R2 + R3) - R2*I1/(R2 + R3);

L1 = laplace(eq1,t,s)
L2 = laplace(eq2,t,s)

%Now we need to solve the system of equations L1 = 0, L2 = 0
%R1 = 4W(ohms), R2 = 2W, R3 = 3W,C= 1/4 farads, L = 1.6 H
%(henrys), I1(0) = 15 amps, and Q(0) =2amps/sec.

syms LI1 LQ
```

```
NI1 = subs(L1,{R1,R2,R3,L,C,'I1(0)','Q(0)'},
{4,2,3,1.6,1/4,15,2})
NQ =
subs(L2,{R1,R2,R3,L,C,'I1(0)','Q(0)'},{4,2,3,1.6,1/4,15,2})
NI1
=subs(NI1,{'laplace(I1(t),t,s)','laplace(Q(t),t,s)'},{LI1,LQ}
)
NI1 = collect(NI1,LI1)
NQ =
subs(NQ,{'laplace(I1(t),t,s)','laplace(Q(t),t,s)'},{LI1,LQ})
NQ = collect(NQ,LQ)
[LI1,LQ] = solve(NI1,NQ,LI1,LQ)
I1 = ilaplace(LI1,s,t)
Q = ilaplace(LQ,s,t)

subplot(2,2,1); ezplot(I1,[0,10]);
title('Current'); ylabel('I1(t)'); grid
subplot(2,2,2); ezplot(Q,[0,10]);
title('Charge'); ylabel('Q(t)'); grid
subplot(2,2,3); ezplot(I1,[5,25]);
title('Current'); ylabel('I1(t)'); grid
text(7,0.25,'Transient'); text(16,0.125,'Steady State');
subplot(2,2,4); ezplot(Q,[5,25]);
title('Charge'); ylabel('Q(t)'); grid
text(7,0.25,'Transient'); text(15,0.16,'Steady State');
```



شکل (۳-۹) نمودار جریان و شارژ بر حسب زمان

۳-۱۱ تبدیل فوریه و تبدیل Z

تبدیل فوریه و تبدیل z ازتبدیلات بسیار مهم در ریاضیات، پردازش سیگنال، و حوزه دیجیتال می‌باشند. برای انجام این محاسبات توابعی در MATLAB تعریف شده‌اند. تبدیل فوریه با استفاده از تابع fourier، عکس فوریه با استفاده ifourier، تبدیل Z با دستور ztrans، و عکس تبدیل z با دستور iztrans انجام می‌شود.

الگوی نوشتاری دستور ifourier و fourier به صورت زیر است:

```
F = fourier(f)
F = fourier(f,v)
F = fourier(f,u,v)


f = ifourier(F)
f = ifourier(F,u)
f = ifourier(F,v,u)
```

و الگوی نوشتاری دستور ztrans و iztrans به صورت زیر است:

```
F = ztrans(f)
F = ztrans(f,w)
F = ztrans(f,k,w)

f = iztrans(F)
f = iztrans(F,k)
f = iztrans(F,w,k)
```

مثال:

```
>> syms x
>> fourier(exp(-x^2))
ans =
exp(-1/4*w^2)*pi^(1/2)
~~~~~~~~~~~~~~~~~
>> syms w
>> ifourier(1/(1 + w^2))
ans =
1/2*exp(x)*heaviside(-x)+1/2*exp(-x)*heaviside(x)
~~~~~~~~~~~~~~~~~
>> syms n
>> ztrans(2^n)
ans =
1/2*z/(1/2*z-1)
~~~~~~~~~~~~~~~~~
>> syms n k w
>> ztrans(sin(k*n),w)
ans =
w*sin(k)/(w^2-2*w*cos(k)+1)
```

```
>> syms z
>> iztrans(z/(z-2))
ans =
2^n
```

❖ تمرینات تکمیلی

۱- مشتق توابع زیر را بیابید.

$1) x^3 + \ln x + x^2 y + \cos y - 2 = 0 \rightarrow y' = ?$

$2) y = (x-1)\sqrt[3]{x^2} \rightarrow y' = ?$

$3) y = \frac{\sin x}{x} \rightarrow y' = ?$

۲- بسط تیلور توابع زیر را بیابید.

$1) y = \cos x$

$2) y = \frac{1}{1+x}$

$3) y = (x^3+1)\ln(x^2+1)$

۳- انتگرال توابع زیر را محاسبه کنید.

$1) \int x \sec^2 x dx$

$2) \int e^{-x} \cos 3x dx$

$3) \int_0^{\infty} \frac{dx}{x^2+1+x\sqrt{x^2+1}}$

$4) \int \frac{\sin 2x}{\sqrt{4-\cos^2 x}} dx$

۴- تبدیل لاپلاس توابع زیر را بیابید.

$1) \int_0^{\infty} (x^2 + \sin 2x) e^{-x} dx$

$2) e^{3t}(2\cos 5t - 3\sin 5t)$

$3) (u(t) - u(t-\pi))\sin t$

$4) 1 + \int_0^t (t-\lambda)^2 e^{t-\lambda} d\lambda$

۵- معکوس تبدیل لاپلاس توابع زیر را بیابید.

$1) \frac{4}{s} - \frac{4e^{-2s}}{s^2} + \frac{4e^{-3s}}{s^2}$

$2) \frac{e^{-\pi s}}{(1+s)^2+1}$

3) $\ln(\frac{s+1}{s^2+2s+5})^{\frac{1}{s}}$

4) $e^{-4s}(\frac{2}{s^2}+\frac{5}{s})$

۶- حدهای زیر را محاسبه کنید.

1) $\lim_{n\to\infty}(\sqrt{n+1}-\sqrt{n})(\sqrt{n+\frac{1}{2}})$

2) $\lim_{x\to+\infty}\frac{(\int_0^x e^{t^2}dt)^2}{\int_0^x e^{2t^2}dt}$

3) $\lim_{x\to0}\frac{(a^x-1)\ln(1+\sin x)}{(Arc\tan x)^2}$

4) $\sum_{n=2}^{\infty}\frac{2^n e^{-2}}{(n-2)!}$

5) $\lim_{n\to\infty}\sum_{i=1}^{n}\frac{1}{n}\ln(1+\frac{i}{n})$

فصل چهارم

درون یابی[1] و برازش منحنی‌ها[2]

۴–۱ برازش منحنی و درون‌یابی

برازش منحنی یعنی پیدا کردن تابعی مانند $y = f(x)$ ، به‌گونه‌ای که این منحنی بهترین منحنی‌ای باشد کـه از مجموعه مقادیر x, y بگذرد.

درون یابی به معنی تخمین مقدار y_i متناظر با x_i ای می‌باشد که بین مقادیر x اندازه‌گیری شده است.

۴–۲ برازش منحنی چند جمله‌ای‌ها

مرسوم ترین روش برای گذراندن منحنی از بین نقاط اندازه گیری شده، استفاده از روش MSE[3] است. بـه ایـن منظور پارامترهای تابع باید به گونه‌ای انتخاب شوند که MSE بین منحنی و مقادیر اندازه گیری شـده مینـیمم شود. این روش MMSE[4] خوانده می‌شود.

رگرسیون چندجمله‌ای، فرمی از MMSE است که تابع انتخاب شده به‌صورت چندجمله‌ای بـوده و پارامترهـای انتخابی همان ضرایب چندجمله‌ای هستند. رگرسیون خطی نیز فرمی از رگرسیون چندجملـه‌ای بـا درجـه یـک است، که دو پارامتر معادله خطی که یک خط راست را نشان می‌دهند، طوری انتخاب می‌شوند که میانگین مربع فواصل بین خط و مقادیر اندازه‌گیری شده مینیمم گردد.

$$MSE = \frac{1}{N}\sum_{k=1}^{N}(\hat{y}_k - y_k)^2$$

$$RMSE = \sqrt{MSE}$$

برای اندازه‌گیری درجه کیفی مناسب بودن منحنی تخمین زده شده نسبت بـه داده‌هـای انـدازه‌گیـری شـده، از RMSE استفاده می‌شود که N تعداد داده‌های اندازه‌گیری شده (x_k, y_k) می‌باشد. توجـه داشـته باشـید کـه واحد MSE مربع واحد کمیت‌های اندازه‌گیری شده و واحد RMSE مشابه کمیتهای اندازه‌گیری شده است.

---

[1] Interpolation
[2] Curve Fitting
[3] Mean Squared Error
[4] Minimum Mean Squared Error

فرض کنید بهترین منحنی درجه ۱ که از این نقاط می‌گذرد عبارت است از $\hat{y} = a_1 x + a_2$ که‌این ضرایب کوچکترین MSE را تولید می‌کنند.

$$MSE = \frac{1}{N}\sum_{k=1}^{N}(\hat{y}_k - y_k)^2 = \frac{1}{N}\sum_{k=1}^{N}(a_1 x_k + a_2 - y_k)^2$$

MSE زمانی مینیمم است که مشتق جزئی نسبت به هر یک از ضرایب برابر صفر باشد.

$$\frac{\partial MSE}{\partial a_1} = \frac{1}{N}\sum_{k=1}^{N} 2(a_1 x_k + a_2 - y_k).x_k$$

$$= \frac{2}{N}\sum_{k=1}^{N}(a_1 {x_k}^2 + a_2 x_k - y_k x_k) = \frac{2}{N}[(\sum_{k=1}^{N} {x_k}^2)a_1 + (\sum_{k=1}^{N} x_k)a_2 - (\sum_{k=1}^{N} x_k . y_k)] = 0$$

$$\frac{\partial MSE}{\partial a_2} = \frac{1}{N}\sum_{k=1}^{N} 2(a_1 x_k + a_2 - y_k)$$

$$= \frac{2}{N}[(\sum_{k=1}^{N} x_k)a_1 + Na_2 - (\sum_{k=1}^{N} y_k)] = 0$$

با صرف نظر کردن از ثابت $\frac{2}{N}$ و نوشتن دو معادله بالا به شکل ماتریس می‌توان ضرایب ناشناخته a_1, a_2 را به‌دست آورد.

$$\Rightarrow \begin{bmatrix} \sum_{k=1}^{N} {x_k}^2 & \sum_{k=1}^{N} x_k \\ \sum_{k=1}^{N} x_k & N \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} \sum_{k=1}^{N} x_k . y_k \\ \sum_{k=1}^{N} y_k \end{bmatrix}$$

دستور MATLAB که بتواند ضرایب a_1، a_2 را محاسبه کند، به‌طوری‌که بهترین منحنی درجه یک را داشته باشیم، عبارت است از:

```
polyfit(x,y,1)
```

مثال: بهترین خط گذرنده از نقاط زیر را به‌دست آورید. از طرفی از این رگرسیون خطی برای تخمین y متناظر با نقطه x=25 استفاده کنید.

```
X=[0  1  2  3  4  5]
Y=[0  20  60  68  77  110]
```

روش اول: استفاده از دستور polyfit

```
>> x = 0:5;
>> y = [0 20 60 68 77 110];
>> a = polyfit(x,y,1)
>> yhat = polyval(a,x)
>> err = yhat - y
```

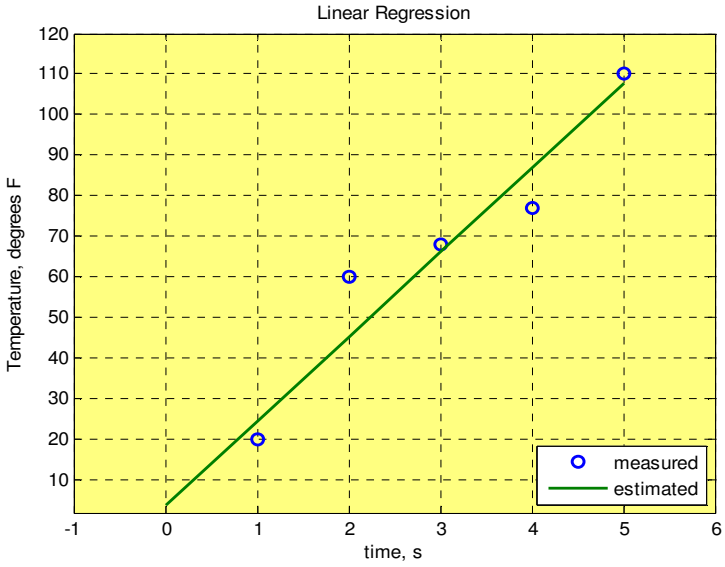
```
>> MSE = mean(err.^2)
>> RMSE = sqrt(MSE)
>> plot(x,y,'o',x,yhat)
>> title('Linear Regression')
>> xlabel('time, s')
>> ylabel('Temperature, degrees F')
>> grid,axis([-1,6,2,120])
>> legend('measured','estimated',4)
```

بهترین منحنی که از این نقاط می‌گذرد عبارت است از:

yhat= 20.8286x + 3.7619

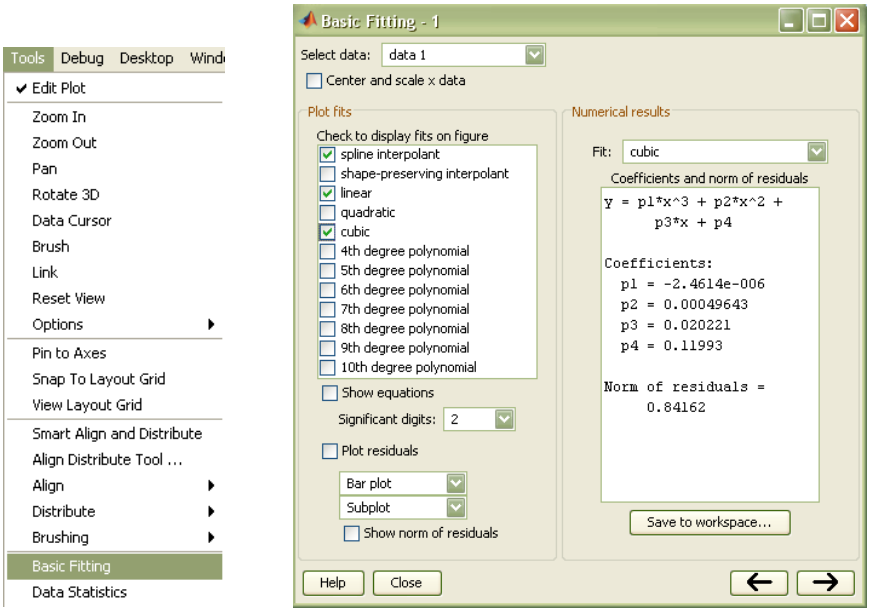
مقدار متناظر با نقطه x=2.5 برابر است با:

```
>> yi = polyval(a,2.5)
>> yi =55.83
```

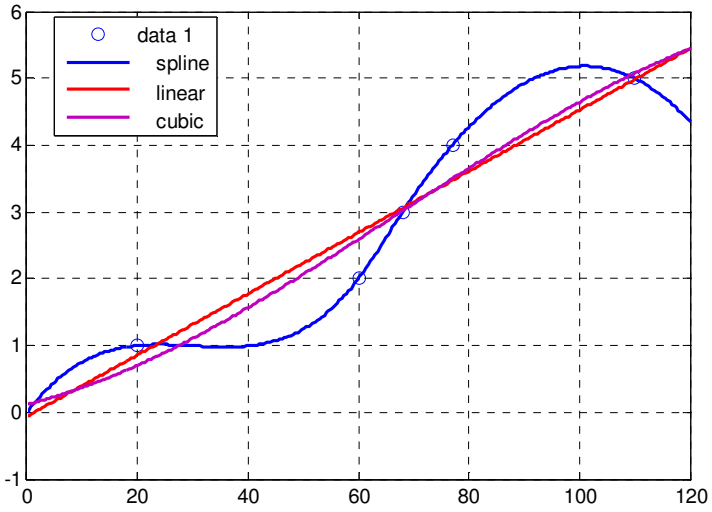


شکل (۴-۱) رگرسیون خطی

روش دوم: بعد از استفاده از دستور plot با انتخاب گزینه Basic Fitting از منوی Tools، در پنجره ظاهر شده روی علامت پیکان مشکی در پائین پنجره کلیک نموده، در پایان گزینه linear و quadratic را انتخاب کنید. همان‌طور که در این پنجره مشاهده می‌کنید ابتدا باید اطلاعات ورودی را در قسمت select data انتخاب کنیم، که در اینجا همان data1 است.



شکل (۴-۲) پنجره Basic Fitting



شکل (۴-۳) رسم بهترین منحنی گذرنده با استفاده از Basic Fitting

۴-۳ رگرسیون چند جمله‌ای

روشی که در بالا برای رگرسیون خطی مطرح شد برای رگرسیون چند جمله‌ای قابل تعمیم است. اگر چند جمله‌ای مرتبه n ام به صورت زیر باشد:

$$f(x) = a_1 x^n + a_2 x^{n-1} + ... + a_{n+1}$$

دستور MATLAB برای رگرسیون چند جمله‌ای به صورت زیر خواهد بود:

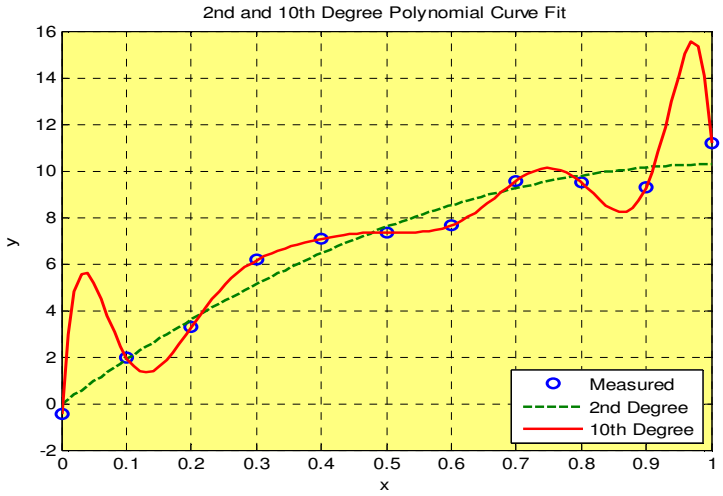
a=polyfit(x,y,n)

بردارهای x و y هم اندازه می‌باشند. خروجی a یک بردار با طول n+1 است. با افزایش درجه تابع، MSE کاهش یافته و تعداد نقاطی که روی منحنی قرار می‌گیرند افزایش می‌یابد؛ به طوری که اگر n+1 نقطه داشته باشیم، یک منحنی درجه n از تمامی نقاط عبور می‌کند. باید توجه شود که این جواب تنها جواب مساله نیست.

مثال: یازده نقطهٔ داده شده زیر را در نظرگرفته و بهترین منحنی درجه ۲ و درجه ۱۰ گذرنده از این نقاط را پیدا کنید.

x = 0:0.1:1;
y = [-0.447 1.978 3.28 6.16 7.08 7.34 7.66 9.56 9.48 9.30 11.2];

```
>> x = 0:0.1:1;
>> y = [-0.447 1.978 3.28 6.16 7.08 7.34 7.66 9.56 9.48 9.30
11.2];
>> a2 = polyfit(x,y,2);
>> disp('a2 coefficients:')
>> disp(a2')
>>a10 = polyfit(x,y,10);
>> disp('a10 coefficients:')
>> format short e
>> disp(a10')
>> xi = linspace(0,1);
>> yi2 = polyval(a2,xi);
>> yi10 = polyval(a10,xi);
>> plot(x,y,'o',xi,yi2,'--',xi,yi10)
>> xlabel('x'), ylabel('y')
>> title('2nd and 10th Degree Polynomial Curve Fit')
>> legend('Measured','2nd Degree','10th Degree',4)
```



شکل (۴-۴) رسم بهترین منحنی گذرنده درجه دو و ده

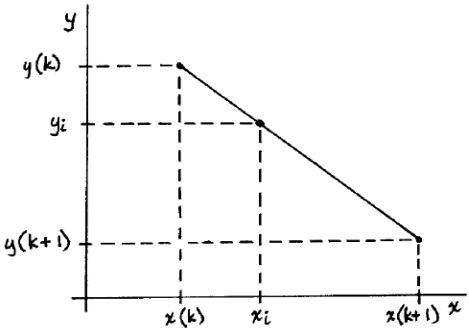
بهترین منحنی درجه ۲ عبارت است از:

$$\hat{y} = -9.8x^2 + 20.12x - 0.03$$

هر چه درجه چند جمله‌ای افزایش یابد تابع تقریب زده شده کمتر هموار شده، و نقاط Max و Min بسیار زیاد خواهد بود.

۴-۴ درونیابی

در درون یابی خطی، تخمین بر مبنای خط راستی است که نقاط مجاور را به هم وصل می‌کند. در درون‌یابی Cubic Spline تخمین بر مبنای پیوند نقاط مجاور با یک چند جمله‌ای درجه ۳ است. روش درون یابی خطی در شکل زیر نمایش داده شده است. نقاط اندازه گیری شده (x(k),y(k)) و (xi(k),yi(k)) هستند که xi(k) بین نقاط x(k),x(k+1) قرار گرفته است.



شکل (۴-۵) درون‌یابی خطی

معادله خط گذرنده بین داده‌ها برابر است با:

$$y_i = y(k) + m(x_i(k) - y(k))$$

$$m = \frac{y(k+1) - y(k)}{x(k+1) - x(k)}$$

$$y_i = y(k) + m(x_i - x(k))$$

دستور MATLAB که این کار را انجام می‌دهد عبارت است از :

yi = interp1(x,y,xi)

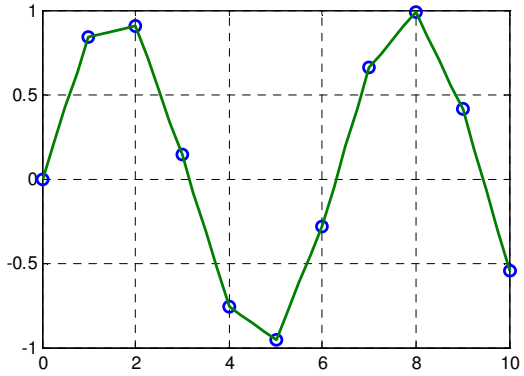
که در آن yi هم سایز xi می‌باشد.

مثال : با داشتن دو بردار زیر، درونیابی خطی را برای مقادیر x=2.6 و x=4.9 انجام دهید.

```
>> x = 0:5;
>> y = [0 20 60 68 77 110];
>> yi = interp1(x,y,[2.6 4.9])
yi = 64.8000    106.7000
```

مثال: در برنامه زیر، درون‌یابی نقاط بین صفر تا ۱۰ انجام شده سپس منحنی نقاط درون‌یابی شده به‌همراه نقاط مورد نظر رسم شده است.

```
>> x = 0:10;
>> y = sin(x);
>> xi = 0:.25:10;
>> yi = interp1(x,y,xi);
>> plot(x,y,'o',xi,yi)
```



شکل (۴-۶) درون یابی با استفاده از Interp1

درون‌یابی با منحنی درجه سه یک منحنی هموار بین دو نقطه می‌باشد:

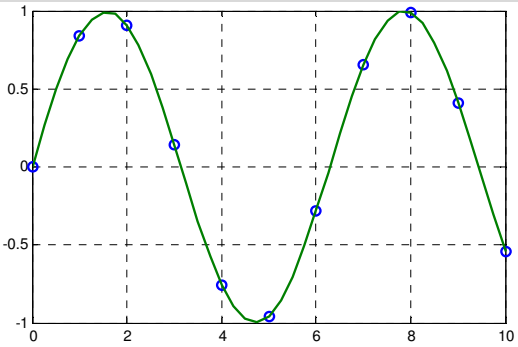
$$y_i = a_1(x_i - x(k))^3 + a_2(x_i - x(k))^2 + a_3(x_i - x(k)) + a_4$$

تابع تعریفی در MATLAB برای پیدا کردن پاسخ برابر y_i به صورت زیر است. توجه شود که طول بردار yi و xi برابر می‌باشند.

yi = spline(x,y,xi)

```
>>x = 0:10;
```

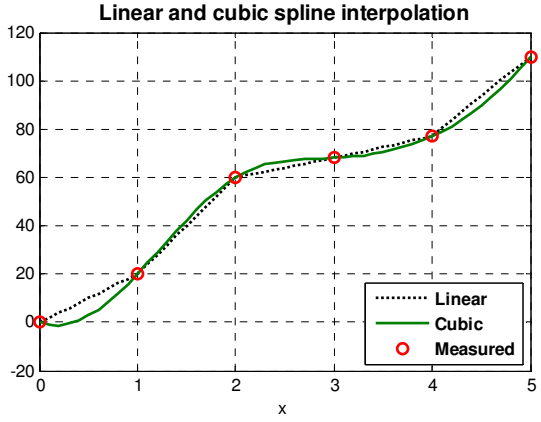
```
>>y = sin(x);
>>xx = 0:.25:10;
>>yy = spline(x,y,xx);
>.plot(x,y,'o',xx,yy)
```



شکل (۴-۷) درون یابی با استفاده از Spline

مثال: مقایسه بین درون‌یابی خطی و درون‌یابی با درجه سه (Cubic Spline)

```
% Comparison of linear and cubic spline interpolation
>> x = (0:5);
>> y = [0,20,60,68,77,110];
>> xi = 0:0.1:5;
>> ylin = interp1(x,y,xi);
>> ycub = spline(x,y,xi);
>> plot(xi,ylin,':',xi,ycub,x,y,'o')
>> legend('Linear','Cubic','Measured',4)
>> title('Linear and cubic spline interpolation')
>> xlabel('x')
>> axis([-1,6,-20,120]),grid
```



شکل (۴-۸) مقایسه بین درون‌یابی خطی و درون یابی با درجه سه

❖ تمرینات تکمیلی

۱- در صورتی‌که $f = 15x^3 - 7x^2 + 2x + 4, \quad g = 9x^2 - 17x + 3$، در آن صورت:

- حاصل‌ضرب f,g را به‌دست آورید.
- خارج قسمت و باقیمانده تقسیم f بر g را به‌دست آورید.
- ریشه‌های چندجمله‌ای g را بیابید.

۲- در صورتی‌که مدول بالک Sic نسبت به درجه حرارت مطابق زیر باشد:

$$B = \left[203_{T=20}, 200_{T=250}, 197_{T=500}, 194_{T=750}, 188_{T=1200}, 186_{T=1400}, 184_{T=1500} \right]$$

- بهترین منحنی درجه ۲،۱و۳ را به‌دست آورید.
- درونیابی را برای مقادیر T=20:10:1500 انجام داده و مقادیر حاصل را رسم کنید.

فصل پنجم

کنترل جریان محاسبات

۵–۱ عملگرهای رابطه‌ای و منطقی

علاوه بر عملگرهای محاسباتی، از عملگرهای منطقی و رابطه‌ای در MATLAB استفاده می‌شود؛ که پاسخ این نوع عملگرها به سوالات مطرح شده به‌صورت True =1 یا False = 0 است. عملگرهای رابطه‌ای و مقایسه‌ای به‌صورت زیر می‌باشند:

۵-۱-۱ عملگرهای رابطه‌ای

عملگرهای رابطه‌ای MATLAB شامل عملگرهای مقایسه‌ای مرسوم می‌باشند. از این عملگرها می‌توان برای مقایسه دو آرایه با یک اندازه یا مقایسه یک آرایه با یک مقدار عددی استفاده کرد. این عملگرها عبارتند از:

جدول(۵–۱) عملگرهای رابطه‌ای

عملگر رابطه‌ای	معنی
<	علامت کوچکتر
<=	علامت کوچکترو مساوی
>	علامت بزرگتر
>=	علامت بزرگتر ومساوی
= =	علامت مساوی
~=	علامت نامساوی

توجه: علامت ' = =' دو مقدار را با هم مقایسه می‌کند. در صورتی که با هم مساوی باشند مقدار ۱ و در غیر این صورت مقدار صفر را بر می‌گرداند. در حالی‌که از '=' برای تخصیص خروجی یک عملگر به یک متغیر استفاده می‌شود.

۵-۱-۲ عملگرهای منطقی

عملگرهای منطقی در MATLAB روشی را برای ترکیب یا قرینه کردن عبارتها ارائه می‌دهند.

جدول(۵–۲) عملگرهای منطقی

عملگر منطقی	معنی
&	AND
|	OR
~	NOT

جدول صحت زیر، وضعیت خروجی را نسبت به حالت‌های مختلف ورودی نشان می‌دهد؛ T(True)و F(False).

جدول(۵-۳) جدول صحت

X1	X2	X1&X2	X1|X2	~X1	XOR(X1,X2)
T	T	T	T	F	F
T	F	F	T	F	T
F	T	F	T	T	T
F	F	F	F	T	F

به مثال‌های زیر توجه کنید:

```
>> 3>5
ans = 0
```

```
>> 3<5
ans = 1
```

```
>> a=[1 2;3 4];
>> b=[1 3;-2 3];
```

```
>> [3 5]<[6 -2]
ans = 1     0
```

```
>> a~=b
ans =
     0     1
     1     1
```

```
>> a>=b
ans =
     1     0
     1     1
```

```
>> c=a>2
c =
     0     0
     1     1
```

```
>> c2=a+(a>=1)
c2 =
     2     3
     4     5
```

```
>> x=[1 2 -1 0 5 4];
```

```
>> x(x>0)
ans = 1     2     5     4
```

```
>> a&b
ans =
     1     1
     1     1
```

```
>> a|b
ans =
     1     1
     1     1
```

```
>> a+(~b)+1
ans =
     2     3
     4     5
```

```
>> a>0 | b>1
ans =
     1     1
     1     1
```

۵-۱-۳ جدول توابع منطقی

در MATLAB به غیر از عملگرهای رابطه‌ای و منطقی، تعدادی از توابع منطقی و رابطه‌ای نیز وجود دارد که در زیر به چند تابع پرکاربرد اشاره شده است:

جدول(۵-۴) جدول توابع منطقی

تابع	توضیحات
any(x)	اگر مقدار موجود در بردار، غیر صفر باشد مقدار ۱ را بر می‌گرداند.
all(x)	اگر تمامی مقادیر موجود در بردار غیر صفر باشند، مقدار ۱ را بر می‌گرداند.
find(x>0)	موقعیت مولفه‌های بزرگتر از صفر بردار x را به‌عنوان خروجی می‌دهد.
exist('a')	در صورتی‌که a جزء متغیرهای فضای‌کاری باشد، خروجی برابر ۱ خواهد بود.
isempty(x)	در صورتی‌که مولفه بردار x تهی باشد، خروجی برابر ۱ خواهد بود.
isinf(x)	در صورتی‌که مولفه بردار x، inf,-inf باشد، خروجی برابر ۱ خواهد بود.
isnan(x)	در صورتی‌که مولفه بردار x , NaN (تعریف نشده) باشد خروجی برابر ۱ خواهد بود.

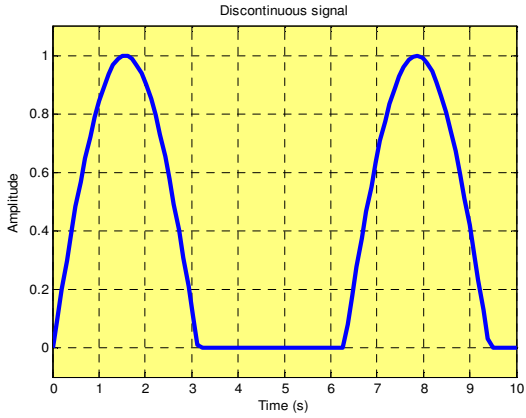
به مثال‌های زیر توجه کنید:

```
>> a=[1 2 -3 4 0 NaN inf 9 -4];
>> any(a>2)
ans =
     1
>> any(a<-5)
ans =
     0
>> all(a>1)
ans =
     0
>> isinf(a)
ans =
     0     0     0     0     0     0     1     0     0
>> isnan(a)
ans =
     0     0     0     0     0     1     0     0     0
>> find(a==inf)
ans =
     7
>> exist('a')
ans =
     1
```

یکی از کاربردهای بردارهای منطقی رسم منحنی‌های ناپیوسته، مشتق ناپذیر است. به‌طور مثال برای رسم منحنی $x(t)=\begin{cases} Sin(t) & Sin(t)>0 \\ 0 & Sin(t)\le 0 \end{cases}$ در محدوده [0 10] از دستورات زیر استفاده می‌کنیم:

```
>> t= linspace(0,10,100);
>> x = sin(t); % compute sine vector
>> x = x.*(x>0); % set negative values of sin(t) to zero
```

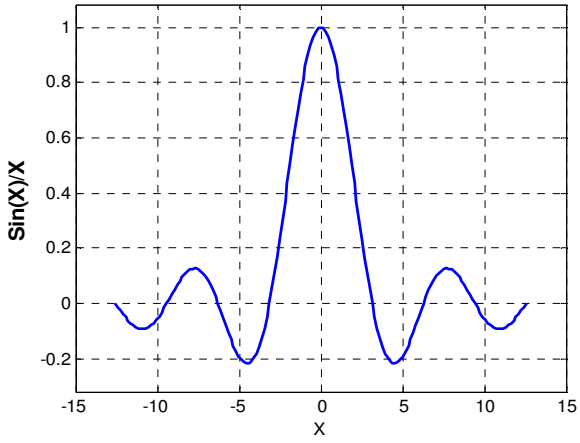
```
>> plot(t,x)
>> xlabel('Time (s)')
>> ylabel('Amplitude')
>> title('Discontinuous signal')
>> axis([0 10 -0.1 1.1])
```



شکل(۵-۱) منحنی سینوس نیم موج

یک راه مناسب برای حل مسأله تقسیم بر صفر، استفاده از توابع منطقی است که می‌توانیم eps (کوچکترین عددی که وقتی به یک اضافه می‌شود، عددی بزرگتر از یک را در کامپیوتر می‌سازد) را جایگزین صفر کنیم. به طور مثال برای رسم منحنی $y(x)=\frac{Sin\ x}{x}$ در بازه $[-4\pi\ \ 4\pi]$ دستورات زیر را می نویسیم:

```
>> x = -4*pi : pi/20 : 4*pi;
>> x = x + (x==0)*eps; % adjust x = 0 to x = eps , eps=epsilon
>> y = sin(x) ./ x; %sin(eps)/eps
>> plot(x, y)
```



شکل(۵-۲) تابع Sinc

مثال: ذره‌ای با سرعت اولیه V_0 تحت زاویه θ_0 نسبت به افق پرتاب می‌شود. ارتفاع و سرعت این ذره با توجه به روابط زیر به‌دست می آید:

$$h(t) = V_0 tSin(\theta) - 0.5gt^2$$

$$V(t) = \sqrt{V_0^2 - 2V_0 gtSin(\theta) + g^2t^2}$$

که در آن g شتاب ناشی از گرانش است. زمان برخورد ذره با زمین عبارت است از:

$$t_g = \frac{2V_0}{g} Sin(\theta)$$

شرایط اولیه برابر $\theta_0 = 40^0, V_0 = 20\frac{m}{s}, g = 9.8\frac{m}{s^2}$ است. بازه زمانی را که ارتفاع بیشتر از ۶ متر و سرعت کمتر از ۱۶ متر برثانیه است، تعیین کنید.

دستورات زیر را دریک m-file وارد کنید:

```
% Set the values for initial speed, gravity, and angle
v0 = 20; g = 9.81; theta = 40*pi/180;
% Compute the time to return to the ground
t_g = 2*v0*sin(theta)/g;
% Compute the arrays containing time, height, and speed.
t = [0:t_g/200:t_g];
h = v0*t*sin(theta) - 0.5*g*t.^2;
v = sqrt(v0^2 - 2*v0*g*sin(theta)*t + g^2*t.^2);
% Determine when the height is no less than 6, and the speed
is no greater than 16.
u = find(h>6 & v <= 16);
% Compute the corresponding times
t_1 = t(u(1))
t_2 = t(u(end))
```

Answers are =

```
>> t_1 =
    0.8518
>> t_2 =
    1.7691
```

با اضافه شدن عملگرهای منطقی و رابطه‌ای، جدول تقدم و تاخر در انجام محاسبات به صورت زیر خواهد بود:

جدول(۵-۵) تقدم و تاخر عملگرها

عملگر	تقدم
پرانتز ( )	۱
(')ترانزپوز .^ ^ توان	۲
Plus, minus, ~(NOT)	۳

۴	* / , \, .* , ./, .\
۵	+ -
۶	:
۷	> < <= >= == ~=
۸	&(AND)
۹	| (OR)

۵-۲ کنترل برنامه

برای این‌که به برنامه نوشته شده این قابلیت اضافه شود که بتواند در حین اجرا تصمیمات لازم را اتخاذ کرده و ترتیب اجرای دستورات را کنترل کند، MATLAB چندین دستور برای کنترل جریان محاسبات معرفی می کند. هر یک از این دستورات با مثال در زیر توضیح داده شده اند.

حلقه های تکرار	• **for- end** • **while -end**
ساختارهای تصمیم	• **if-else -End** • **switch-case**
توقف اجرای برنامه	• **break** • **return, keyboard** • **pause, pause(n)** • **waitforbuttonpress**

۵-۲-۱ شرط If-else-end

حتماً با عملکرد این دستور در زبان‌های برنامه نویسی دیگر آشنا شده‌اید. ساختار کلی این دستور را در زیر ملاحظه می‌کنید.

```
If          شرط ۱
    دستورات ۱
Elseif      شرط۲
    دستورات ۲
Elseif      شرط۳
    دستورات ۳
Else
    دستورات ۴
End
```

همانطور که مشاهده می‌کنید در حالت کلی می‌توان از یک if، بیشمار else if و یک else و end استفاده کرد. البته استفاده از else وelse if اختیاری است. دستور if برنامه را قادر می‌سازد که تصمیم بگیرد چه دستورهایی باید اجرا شوند.

اگر شرط مقابل if درست باشد، دستورات شماره ۱ اجرا می‌شوند؛ در غیر این صورت شرط روبروی else if بررسی می شود، در صورتی که درست باشد دستورات روبروی else if اجرا می‌شود. در غیر این صورت شرط ۳ بررسی می‌شود و الی آخر ... . در صورتی که n شرط بررسی شده درست نبوند، دستورات قسمت else اجرا می‌شوند.

مثال: برنامه‌ای بنویسید که در صورتی‌که کلمه عبور وارد شده از طرف کاربر درست بود، صدایی پخش شود.

```
load handel
x = input('Enter your password: ');
if x == 206
    disp('your password is correct'),sound(y,Fs)
elseif (x > 206 )|(x<206)
    disp('your password is incorrect')
end
```

در برنامه بالا با دستور load فایل handel اجرا می‌شود. در خط بعدی از کاربر کلمه عبور پرسیده می‌شود. در صورتیکه کلمه عبور برابر ۲۰۶ باشد در آن صورت پیغام درست بودن کلمه عبور نمایش داده شده و صدایی پخش می‌شود. در غیر این صورت پیغام نادرست بودن کلمه عبور نمایش داده خواهد شد.

۵-۲-۲ شرط switch-case

از این ساختار برای تصمیم‌گیری چندگانه بر اساس مقادیر مختلف یک متغیر استفاده می‌شود. به‌طورکلی در تصمیم‌گیری‌هایی که ۳ انتخاب بیشتر وجود ندارد از این دستور به جای if استفاده می شود. به‌عنوان مثال فرض کنید متغیری مثل a مقادیر۱،۲و۳ را اختیار می‌کند و می‌خواهید بر اساس مقادیر مختلف a تصمیم‌گیری مختلفی انجام دهید. اگر برابر ۱ بود دستورات ۱، اگر برابر ۲ بود دستورات ۲ و اگر برابر ۳ بود دستورات ۳ اجرا شوند و در صورتی که هیچ کدام از آنها نبود دستورات ۴ (otherwise) اجرا شوند.

```
a = input('a =');
switch a
    case 1
        disp('One')
    case 2
        disp('Two')
    case 3
        disp('Three')
    otherwise
        disp('a>3')
end
```

نکته: بعد از اجرای هر یک از دستورات، روند اجرای برنامه به بعد از end منتقل می شود و سایر caseها کنترل نمی شوند. استفاده از otherwise نیز اختیاری است.

مثال: برنامه‌ای بنویسید که اعداد بین ۰ تا ۹ را به صورت تصادفی بگیرد و سپس پیغام دهد که این عدد زوج، فرد، و یا صفر است.

دستورات زیر را دریک m-file وارد کنید:

```
d = floor(10*rand);
switch d
    case {2, 4, 6, 8}
        disp( 'Even' );
    case {1, 3, 5, 7, 9}
        disp( 'Odd' );
    otherwise
        disp( 'Zero' );
end
```

۵-۲-۳ حلقه for

این حلقه این امکان را به‌وجود می‌آورد که تعدادی از دستورات به تعداد دفعات از قبل تعیین شده تکرار شوند. شکل کلی این دستور به صورت زیر است:

For variable = a

دستورات

End

که در آن a یک بردار است که در هر بار تکرار حلقه یک مقدار بردار a در variable قرار می گیرد. به این ترتیب حلقه به تعداد ستون‌های a تکرار می شود. به‌طور مثال در برنامه فاکتوریل زیر، فاکتوریل اعداد ۱ تا n محاسبه و نمایش داده می‌شوند.

```
n = input('n = ')
fact = 1;
for k = 1:n
 fact = k * fact;
 disp( [k fact] )
end
```

حلقه for را می توان به صورت تودرتو استفاده کرد. به‌طور مثال با استفاده از حلقه‌های تودرتو می‌توان جدول ضرب ایجاد کرد.

```
for i=1:10
    for j=1:10
      S(i,j)=i*j;
    end
end
disp(S)
```

مثال: محاسبه مدت زمان انجام برنامه

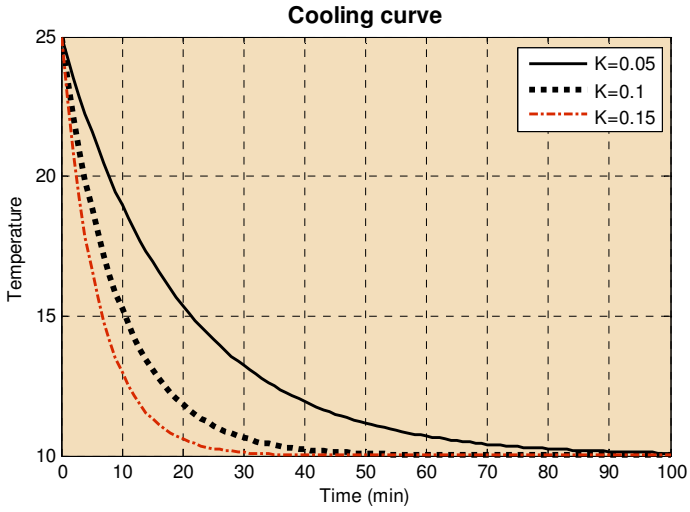
```
t0 = clock;
s = 0;
for n = 1:100000
    s = s + n;
end
```

```
etime(clock,t0)
```

خروجی دستور clock یک بردار یک در شش است که مولفه‌های آن عبارتند از: year, month, day, hour, minute, second. دستور etime برای مقایسه دو آرگومان clock و زمان t است و خروجی آن در مثال بالا براساس ثانیه می‌باشد.

مثال: یک بطری آب با درجه حرارت ۲۵ درجه در داخل یخچال با دمای ۱۰ درجه قرار داده می‌شود. تغییرات دمایی بطری آب را رسم کنید. تغییرات درجه حرارت از رابطه $T_{i+1} = T_i + K \times \Delta t \times (F - T_i)$ به‌دست می‌آید.

```
K = 0.15; % Conduction coefficient
F = 10; % Refrigerator temperature (degrees C)
t = 0:100;
T = zeros(1,101); % Pre allocate temperature vector
dt=1;
T(1) = 25; % Initial temperature (degrees C)
for i = 1:100; % Time in minutes
    T(i+1) = T(i) + K * dt*(F - T(i)); % Compute T
end
disp([ t(1:10:101)' T(1:10:101)' ])
hold on
plot(t,T)
grid,xlabel('Time (min)'),ylabel('Temperature')
title('Cooling curve')
```



شکل(۵-۳) تغییرات دمایی بطری آب با ضرایب انتقال حرارت مختلف

۵-۲-۴ حلقه While

این حلقه چند دستور را به تعداد دفعات نامحدود تکرار می‌کند. این دستورات زمانی تکرار و اجرا می‌شوند که عبارت شرطی درست باشد.

while شرط

دستورات

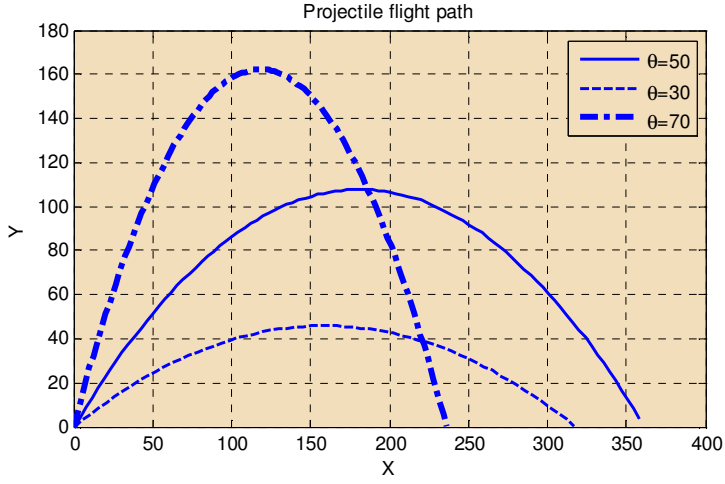
end

مثال: سنگی با سرعت اولیه ۶۰ متر بر ثانیه و تحت شتاب گرانشی ۹/۸ متر بر مجذور ثانیه، تحت زاویه‌ای دلخواه پرتاب می‌شود. مسیر حرکت پرتابه را رسم کنید.

```
dt = 0.1;
g = 9.8;
v = 60;
ang = input( 'Launch angle in degrees: ' );
ang = ang * pi / 180;
xp = zeros(1); yp = zeros(1); % initialize
y = 0; t = 0;
i = 1;
```

تا زمانیکه y>=0 باشد محاسبات x,y انجام می‌شود.

```
while y >= 0
    t = t + dt;
    i = i + 1;
    y = v * sin(ang) * t - g * t^2 / 2;
    if y >= 0
        xp(i) = v * cos(ang) * t;
        yp(i) = y;
    end
end
plot(xp, yp),grid
```



شکل(۵-۴) مسیر حرکت پرتابه تحت زاویه‌های مختلف

مثال: در بازی گوسین عددی صحیح بین ۱-۱۰ به صورت تصادفی انتخاب می‌شود. در صورتی‌که حدس شما از این عدد بزرگتر باشد جمله Too high و در صورتی‌که حدس شما از این عددکوچکتر باشد جمله Too low نمایش داده می‌شود.

```
num = floor(10 * rand + 1);
guess = input( 'Your guess please: ' );
load splat
while guess ~= num
    sound(y, Fs)
    if guess > num
        disp('Too high' )
    else
        disp( 'Too low' )
    end;
    guess = input( 'Your next guess please: ' );
end
disp( 'At last!')
load handel
sound(y, Fs)
```

۵-۲-۵ دستور break

هنگامی که این دستور اجرا می شود کنترل برنامه به اولین خطا بعد از حلقه for منتقل می شود. اگر حلقه‌ها تودرتو باشند این دستور باعث خارج شدن از حلقه جاری می‌شود.

```
for   p=7:8
    for   q=3:5
        for    r=1:2
            if   q == 4
                break
            end
            disp ([p,q,r])
        end
    end
end
```

۶-۲-۵ دستور pause

هنگامی که برنامه در حین اجرا به دستور pause برسد متوقف می ماند تا اینکه شما صفحه کلید را فشار دهید و سپس اجرای برنامه از دستور بعد از pause ادامه می یابد. pause(n) توقف به مدت n ثانیه است.

```
k=0;
for  x=0:.2:1
    if  k>=5
        disp('k>=5'); break
    end
    k=k+1;
```

```
    y=exp(-x);
    disp(['k=',num2str(k),'y=',num2str(y)])
    pause
end
```

در این مثال، برنامه هر بار پس از نشان دادن مقادیر y,k متوقف می شود تـا اینکـه کلیـدی روی صـفحه کلیـد فشرده شود. دستور break نیز باعث خارج شدن از حلقه for می شود و برنامه پایان می یابد.

۵-۲-۷ دستور return

هرگاه روند اجرای برنامه به این دستور برسد مقدار مورد نظر را در command window نمایش داده و ادامه اجرای برنامه متوقف می شود.

۵-۲-۸ دستور keyboard

در صورت استفاده از دستور keyboard در میان برنامه اجرای برنامه هنگامی که به آن دستور می‌رسد بـه‌طـور موقت متوقف شده و به شما اجازه می دهد که عملیات مورد نظرتان را انجام دهید. در چنین حـالتی علامـت k روی صفحه نمایش دیده می شود. پس از اینکه دستور return وارد شد برنامه از جـایی کـه متوقـف شـده بـود ادامه می یابد. این دستور به‌ویژه در مواقعی به کار می رود که بخواهید وسط برنامه مقدار یک متغیر را ببینیـد و آن را تغییر دهید و اثر این تغییر را در اجرای ادامه برنامه مشاهده نمائید.

۵-۲-۹ دستور waitforbuttonpress

با اجرای این دستور برنامه تا زمانی که دکمه ماوس یا یک کلید در صفحه باز شده فشرده شود، متوقف می‌ماند.

```
w=waitforbuttonpress
if w == 0
    disp('button click')
else
    disp('key press')
end
```

۵-۳ ساخت فایل تابعی (Function file)

گاهی اوقات ممکن است نیاز به توابعی با کاربری خاص داشته باشیم که در MATLAB تعریف نشده اند. بنابراین تنها استفاده از توابع تعریف شده در MATLAB جوابگوی ما نخواهند بود. بنابراین نیاز داریم که این توابع را تعریف کنیم و یاگاهی اوقات نیز به‌خاطر راحتی استفاده مجدد از برنامه و کاهش پیچیدگی‌های برنامه های جامع با ایجاد قابلیت خواندن بیشتر، از function fileها استفاده کنیم. یک فایل تابعی مانند یک m-file است با این تفاوت که خط اول آن به صورت زیر است:

Function [outputs] = name[inputs]

در این خط هر یک از خروجی‌ها و ورودی‌ها را در یک متغیر جداگانه قرار می‌دهیم. در صورتی‌که یک خروجی داشته باشیم، دیگر نیازی به براکت نیست. name نیز نام تابع است که بهتر است بین ۹ تا ۲۰ کاراکتر باشد و

نام آنها نباید مشابه توابع موجود و یا توابع قبلاً تعریف شده در MATLAB باشد. برای شروع، یک M-file جدید باز کرده، دستورات زیر را در آن تایپ کنید و آن را با همان اسم پیش فرض f ذخیره کنید.

```
function y = f(x)
y = x^3 + x - 3;
```

حال می‌توانید با فراخوانی تابع f، مقادیر مختلف f را در نقاط مختلف حساب کنید؛ به‌طور مثال f(3.5). در پنجره دستورات عبارت‌های زیر را تایپ کنید:

```
>> f(3.5)
ans =
   43.3750
```

مثال: تابعی بنویسید که ورودی آن ماتریس و خروجی آن موقعیت سطر و ستون بزرگترین عدد موجود در ماتریس باشد.

```
function [r,c] = indmax(x)
[m n] = size(x);
xmax = x(1,1);
r=1; c=1;
for k=1:m
    for i=1:n
        if x(k,i)> xmax
            xmax = x(k,i);
            r=k;
            c=i;
        end
    end
end
```

برای تست برنامه، دستورات زیر را در پنجره دستورات وارد کنید:

```
>> A=3; B=[2 4 3]; C=[3 2; 6 1];
>> [r c]=indmax(A)
>> [r c]=indmax(B)
>> [r c]=indmax(C)
```

نکته: ممکن است توابع برای نمایش داده‌ها در پنجره دستورات یا نوشتن داده‌ها در فایل‌ها ایجاد شوند. در این موارد هیچ مقداری به برنامه فراخوانی شده برنمی‌گردد. ابتدای برنامه این‌گونه توابع به صورت زیر می‌باشد:

Function name(inputs)

به‌طور مثال در برنامه زیر، ورودی تابع یک اسکالر، بردار، و یا یک ماتریس می‌تواند باشد در حالی‌که خروجی برنامه نمایش یک رشته در پنجره دستورات می‌باشد.

```
function testvar(x)
[m,n] = size(x);
if m==n & m==1
    disp(' Argument is a scalar')
elseif m==1 | n==1
    disp(' Argument is a vector')
else
    disp(' Argument is a matrix')
end
```

برای تست برنامه، دستورات زیر را در پنجره دستورات وارد کنید:

```
>> a=2; b=[2 3]; c=[4 5; 6 7];
>> testvar(a)
>> testvar(b)
>> testvar(c)
```

۵-۴ حل عددی معادله $f(x)=0$

همانطور که در کتابهای ریاضیات دیده‌اید سه روش عددی برای حل معادله f(x)=0 مطرح شده است. روش نیوتن، روش سکانت، و روش دو نیم کردن.

۵-۴-۱ حل معادله $f(x)=0$ با استفاده از روش نیوتن

در روش نیوتن تقریب تابع با خطی کردن در یک نقطه انجام می‌شود؛ یعنی

$$f(x) \approx f(x_0) + f'(x_0)(x - x_0)$$

از آنجایی که هدف پیدا کردن x ای است که به ازای آن f(x)=0 باشد، بنابراین:

$$x \approx x_0 - \frac{f(x_0)}{f'(x_0)}$$

برای حل این مساله با حدس اولیه x_0 شروع می‌کنیم تا بتوانیم تقریبا به x^* یا جواب واقعی برسیم. بنابراین یکسری از نقاط را به صورت زیر تعریف می‌کنیم:

$$x_{i+1} \approx x_i - \frac{f(x_i)}{f'(x_i)}$$

باید برای حل این مساله از حلقه‌ها استفاده کنیم. ابتدا در یک m-file تابع mynewton را به صورت زیر می-نویسیم.

```
function x = mynewton(f,f1,x0,n)
format long
format compact
x = x0; % set x equal to the initial guess x0
for i = 1:n
```

```
    x = x - f(x)/f1(x)
end
```

سپس در پنجره فرامین با استفاده از دستور inline تابع $f(x)=x^3-5, f'(x)=3x^2$ را بنویسید (دستور inline در ادامه توضیح داده می‌شود) و در نهایت برنامه mynewton را فراخوانی کنید.

```
f = inline('x^3 - 5');
f1 = inline('3*x^2');
x = mynewton(f,f1,0.1,10)
```

حال اگر یک بازه تلرانسی معین برای $|f(x)|$ داشته باشیم، می‌توانیم بادستورات شرطی if-end این شرط را اضافه کنیم.

```
function x = mynewton2(f,f1,x0,n,tol)
x = x0; % set x equal to the initial guess x0
for i = 1:n
    x = x - f(x)./f1(x);
end
r = abs(f(x));
if r > tol
    warning('The desired accuracy was not attained')
end
```

```
f = inline('x^3 - 5');
f1 = inline('3*x^2');
x = mynewton2(f,f1,0.1,10,0.01)
```

یک راه برای کنترل تعداد مراحل، استفاده از دستور while است. در این مساله مراحل تا زمانی ادامه پیدا می‌کند که $|f(x)|<tol$ به اندازه کافی کوچک شود.

```
function x = mynewtontol(f,f1,x0,tol)
x = x0; % set x equal to the initial guess x0
y = f(x);
while abs(y) > tol
    x = x - y/f1(x);
    y = f(x);
end
```

```
f = inline('x^3 - 5');
f1 = inline('3*x^2');
x = mynewtontol(f,f1,0.1,0.01)
```

۵-۴-۲ حل معادله $f(x)=0$ با استفاده از روش دو نیم کردن

برتری این روش نسبت به روش نیوتن در این است که می‌دانیم جواب $f(x)=0$ در بازه $[a\ b]$ می‌باشد. چون f(a) و f(b) از نظر علامت با هم فرق دارند. بنابراین ما درباره مقدار ماکزیمم خطا اطلاع داریم و مقدار خطا از نصف این فاصله کوچکتر است.

$$\left|x - x^*\right| < (b-a)/2$$

ابتدا یک تابع برای حل این مساله می‌نویسیم و آن را با نام mybisect ذخیره می‌کنیم.

```
function [x e] = mybisect(f,a,b,n)

% Inputs: f -- an inline function
% a,b -- left and right edges of the interval
% n -- the number of bisections to do.
% Outputs: x -- the estimated solution of f(x) = 0
% e -- an upper bound on the error
format long
c = f(a); d = f(b);
if c*d > 0.0
    error('Function has same sign at both endpoints.')
end
disp(' x y'')
for i = 1:n
    x = (a + b)/2;
    y = f(x);
    disp([ x y])
    if y == 0 % solved the equation exactly
        e = 0;
        break
    end
    if c*y < 0
        b=x;
    else
        a=x;
    end
end
e = (b-a)/2;
```

سپس تابع f را با inline تعریف کرده و تابع را اجرا می‌کنیم.

```
f = inline('x^3 - 5');
mybisect(f,1.5,3,20)
 x y'
   2.250000000000000   6.390625000000000
   1.875000000000000   1.591796875000000
   1.687500000000000  -0.194580078125000
   ...
   1.709970474243164  -0.000048004324536
   1.709973335266113  -0.000022907397622
   1.709974765777588  -0.000010358902673
ans =
   1.709974765777588
```

۵-۴-۳ حل معادله $f(x)=0$ با استفاده از روش سکانت

در روش سکانت به دو نقطه اولیه x_0 ,x_1 نیاز داریم که به جواب x^* نزدیک می‌باشند. ترجیحاً علامت $y_0=f(x_0)$ و $y_1=f(x_1)$ باید متفاوت باشند. یکبار که اعداد x_0 ,x_1 تعیین شدند، بقیه اعداد با فرمول زیر به‌دست می‌آیند:

$$x_{i+1}=x_i-\frac{x_i-x_{i-1}}{y_i-y_{i-1}}y_i$$

ابتدا یک تابع برای حل این مساله می‌نویسیم و آن را با نام mysecant ذخیره می‌کنیم.

```
function x = mysecant(f,x0,x1,n)
format long
format compact
% Inputs: f -- the function, input as an inline function
% x0 -- starting guess, a number
% x1 -- second starting guess
% n -- the number of steps to do
% Output: x -- the approximate solution
y0 = f(x0);
y1 = f(x1);
for i = 1:n % Do n times
    x = x1 - (x1-x0)*y1/(y1-y0) % secant formula.
    y=f(x) % y value at the new approximate solution.
    x0=x1;
    y0=y1;
    x1=x;
    y1=y;
end
```

سپس تابع f را با inline تعریف کرده و mysecant را اجرا می‌کنیم.

```
f = inline('x^3 - 5');
mysecant(f,0,2,8)
```

۵-۵ ساخت تابع با استفاده ازدستور inline

می‌توان یک تابع را در پنجره دستورات یا در یک m-file با استفاده از inline ایجاد کرد. مزیت آن این است که به ذخیره شدن در یک فایل جداگانه نیاز ندارد، اما محدودیت‌هایی نیز وجود دارد که نمی‌توان inline دیگری را فراخوانی کرد. شکل عمومی inline به‌صورت زیر می‌باشد:

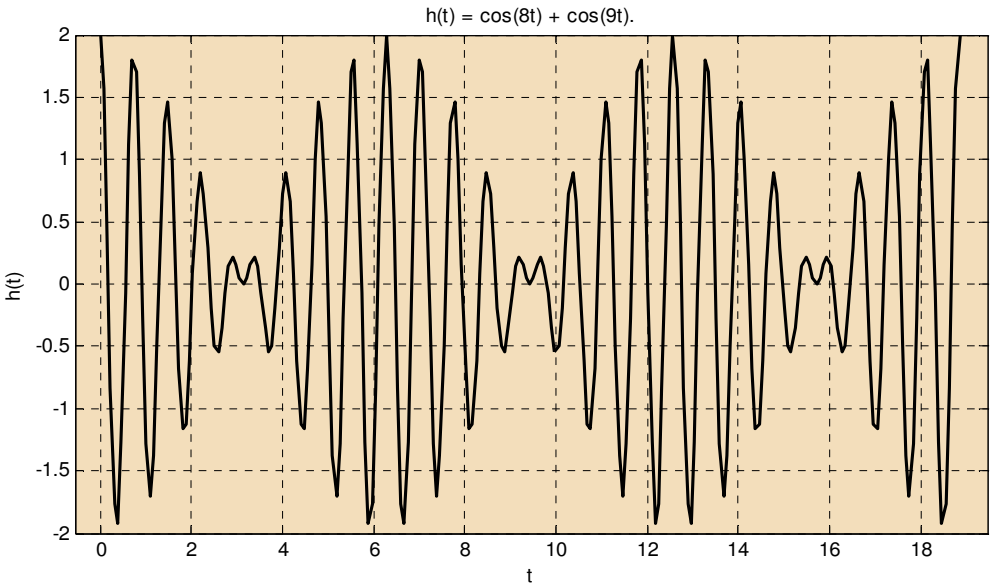
Name of function = inline(‘ function’,’x1’,’x2’,…)

x1,x2,… نام متغیرهای موجود در عبارت می‌باشند. به‌عنوان نمونه در مثال زیر تابع h با استفاده از تابع inline تعریف شده است که متغیر t تنها متغیر آن می‌باشد.

$$h(t)=\cos(8t)+\cos(9t)$$

```
>> h=inline('cos(8*t)+cos(9*t)','t')
```

```
h =
     Inline function:
     h(t) = cos(8*t)+cos(9*t)
>> h(2)
ans =
   -0.2973
>> x = 0 : pi/40 : 6*pi;
>> plot(x,h(x))
```



شکل(۵-۵) منحنی $h(t)=\cos(8t)+\cos(9t)$

```
>>g=inline('sin(t).*cos(x)','t','x')
g =
     Inline function:
     g(t,x) = sin(t).*cos(x)
>>g(2.5,-1)
ans =
    0.3234
```

۵-۶ توابع کاربردی MATLAB برای محاسبه توابع تعریفی کاربر

در MATLAB چندین تابع وجود دارد که توابع تعریفی کاربر را محاسبه می‌کنند. در ادامه مهمترین آنها را شرح می‌دهیم.

۵-۶-۱ پیدا کردن مینیمم نسبی با استفاده از تابع fminbnd

برای پیدا کردن مینیمم نسبی تابع یک متغیره f(x) در فاصله $a \le x \le b$، از تابع fminbnd استفاده می‌شود. فرمت این دستور به صورت زیر است:

[x,fval]=fminbnd('F',a,b)

مقدار x متناظر با نقطه‌ای است که مینیمم نسبی در بازه $[a,b]$ رخ داده است و fval مقدار تابع در نقطه x است. F نیز نام فایل تابعی است که در یک علامت نقل قول قرار گرفته، یا تابعی است که با inline ایجاد شده است. به‌طور مثال برای پیدا کردن مینیمم نسبی تابع cos در بازه [0,4] دستورات زیر را می‌نویسیم:

```
[x,fval]=fminbnd('cos',0,4)
x =
    3.1416
fval =
   -1.0000
>> x=0:0.1:4;
>> plot(x,cos(x))
```

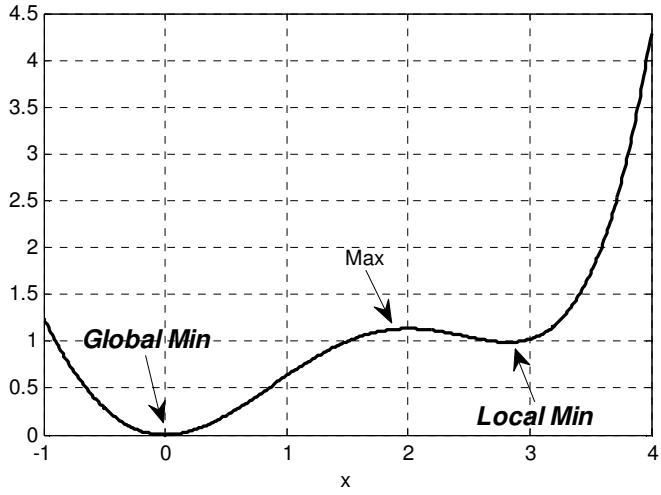
مثال: مینیمم نسبی تابع تک متغیره $y(x) = 0.025x^5 - 0.0625x^4 - 0.333x^3 + x^2$ را در بازه [4 -1] به‌دست آورید.

ابتدا نیاز داریم تابع را در یک فایل تابعی و یا با دستور inline تعریف کنیم:

```
function y=f5(x)
y=0.025*x.^5-0.0625*x.^4-0.333*x.^3+x.^2;
```

با رسم نمودار تابع در بازه [4 -1] متوجه می‌شویم که این تابع دو مینیمم نسبی دارد که هر یک را باید در بازه‌های مختلف به‌دست آوریم:

```
>> x=-1:0.01:4;
>> y=0.025*x.^5-0.0625*x.^4-0.333*x.^3+x.^2;
>> plot(x,y)
>> [x,fval]=fminbnd('f5',-1,4)% Global minimum
x =
  7.1589e-006
fval =
  5.1250e-011
>> xmin=fminbnd('f5',1,4)% Local minimum
xmin =
    2.8236
fval =
    0.9905
```



شکل(۵-۶) نمایش نقاط مینیمم نسبی و مطلق در منحنی $y(x)=0.025x^5-0.0625x^4-0.333x^3+x^2$

۵-۶-۲ حل معادلات غیرخطی با استفاده از fsolve

تابع fsolve حل عددی معادلات غیرخطی مرتبه n $f_n(x_1,x_2,...,x_n)$ با x_n مجهول را با استفاده از فرض اولیه $x_d=[x_1,x_2,...,x_{nd}]$ انجام می‌دهد. فرمت این دستور به صورت زیر می‌باشد:

Fslove('F',x_d)

که در آن، F نام فایل تابعی است که در یک علامت نقل قول قرار گرفته، یا تابعی است که با inline ایجاد شده است وx_d فرض اولیه می‌باشد.

مثال: معادلات غیرخطی زیر را حل کنید.

$$\begin{cases} x_1^2+4x_2^2=5 \\ 2x_1^2-2x_1-3x_2=2.5 \end{cases}$$

برای اینکه ببینیم دستگاه معادلات به چه صورت حل می‌شود، در ابتدا معادلات را به صورت زیر می‌نویسیم:

$$\begin{cases} f_1(x_1,x_2)=0 \\ f_2(x_1,x_2)=0 \end{cases} \Rightarrow \begin{cases} f_1(x_1,x_2)=x_1^2+4x_2^2-5=0 \\ f_2(x_1,x_2)=2x_1^2-2x_1-3x_2-2.5=0 \end{cases}$$

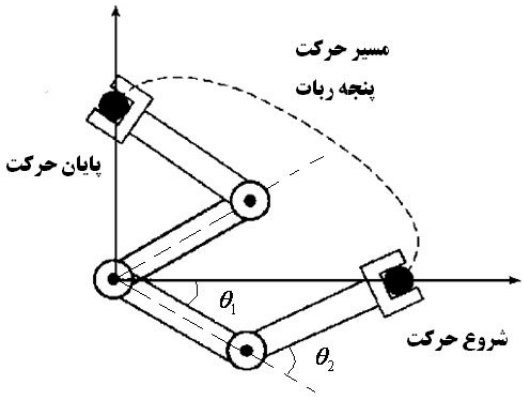
یک فایل تابعی ایجاد کرده و دستورات زیر را وارد کنید:

```
function y=fx1x2(x)
y(1)=x(1).^2+4*x(2).^2-5;
y(2)=2*x(1).^2-2*x(1)-3*x(2)-2.5;
```

دستورات زیر را درپنجره فرامین واردکرده، و پاسخ را مشاهده کنید:

```
>> x0=[0.8 0.2];
>> fsolve('fx1x2',x0)
ans =
    2.0000    0.5000
```

مثال: مساله حرکت روبات دو لینکی در فصل اول را در نظر بگیرید. اگر موقعیت نوک پنجه ربات در نقطه $(x, y) = (6.5, 0)$ باشد، با در نظرگرفتن فرض اولیه $\theta_1(0) = -\frac{\pi}{6}, \theta_2(0) = +\frac{\pi}{6}$ برای هر یک از زوایای لینک‌ها، مقادیر واقعی زاویه‌های اولیه بازوهای ربات θ_1, θ_2 را پیدا کنید.



شکل(۵-۷) روبات دو لینکی با مفاصل صلب

معادلات زیر بر ربات دولینکی حاکم است:

F1 = 4*cos(theta1)+3*cos(theta1+theta2)-6.5;
F2 = 4*sin(theta1)+3*sin(theta1+theta2);

یک فایل تابعی ایجاد کرده و دستورات زیر را وارد کنید:

```
function y=robot(theta)
y(1)=-6.5+4*cos(theta(1))+3*cos(theta(1)+theta(2));
y(2)=4*sin(theta(1))+3*sin(theta(1)+theta(2));
```

دستورات زیر را درپنجره فرامین واردکرده و پاسخ را مشاهده کنید:

```
fsolve('robot',[-pi/6 pi/6])*180/pi
ans =
  -18.7170   44.0486
```

۵-۶-۳ انتگرال‌گیری عددی با استفاده از trapz و quad و انتگرال دوگانه و سه گانه عددی

تابع **trapz(x,y)** برای به‌دست آوردن انتگرال تقریبی تابع f(x) با استفاده از روش ذوزنقه‌ای به‌کار می‌رود. مقادیر x و مقادیر y متناظررا، باید به‌صورت آرایه‌ای وارد کرد. سطح زیر منحنی طبق قانون ذوزنقه‌ای به صورت زیر به‌دست می‌آید:

$$A = \frac{\Delta x}{2}(f(x_0) + 2f(x_1) + 2f(x_2) + ... + 2f(x_{n-1}) + f(x_n)) \quad , \Delta x = \frac{b-a}{n}$$

هر چه مقدار Δx کوچک باشد، دقت محاسبات بیشتر می‌شود. به‌طور مثال، سطح زیر منحنی تابع humps، با Δxهای مختلف را به‌دست آورید تا اثر Δx را مشاهده کنید.

```
>>x = linspace(-1,2,18);
>>y = humps(x);
>>plot(x,y)
>>area = trapz(x,y)
area = 25.1406
===========
>>x = linspace(-1,2,401);
>>y = humps(x);
>>area = trapz(x,y)
area = 26.3449
```

تابع پیش ساخته quad به طور عددی انتگرال f(x) را از حد پائین a تا حد بالای b محاسبه می‌کند. فرمت دستور quad به صورت زیر است:

quad('function',a,b) *%Simpson's rule*

سطح زیر منحنی طبق قانون سیمسون به صورت زیر به‌دست می‌آید:

$$A=\frac{\Delta x}{3}(f(x_0)+4f(x_1)+2f(x_2)+4f(x_3)...+2f(x_{2n-2})+4f(x_{2n-1})+f(x_{2n})) \quad , \Delta x=\frac{b-a}{2n}$$

```
>> quad('humps',-1,2)
ans =
   26.3450
```

انتگرال توابع دو متغیره $\int_{y_{\min}}^{y_{\max}}\int_{x_{\min}}^{x_{\max}} f(x,y)dx\,dy$ با استفاده از تابع dblquad محاسبه می‌شود. فرمت عمومی دستور به صورت زیر می‌باشد:

q = dblquad('function',xmin,xmax,ymin,ymax)

مثال: مقدار انتگرال دوگانه $\int_{-\pi}^{\pi}\int_{0}^{\pi}(\sin(x).\cos(y)+1)dx\,dy$ را به‌دست آورید.

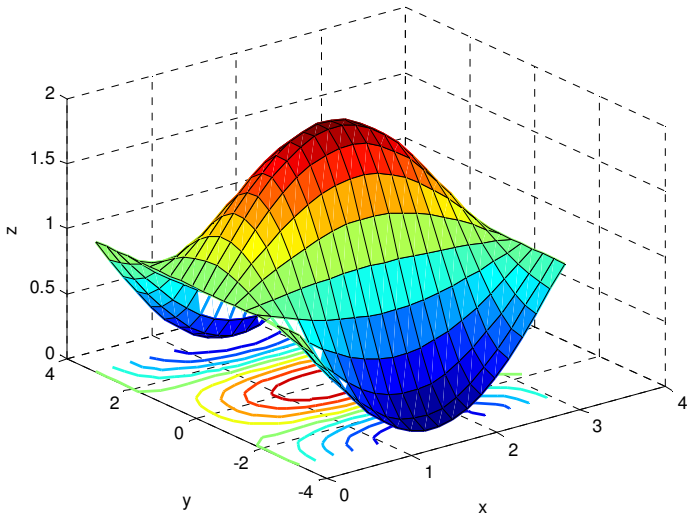
در ابتدا تابع دو متغیره را در یک فایل تابعی تعریف می‌کنیم:

```
function z = func2(x,y)
z = sin(x).*cos(y) + 1;
```

در پنجره فرامین دستورات زیر را وارد کنید:

```
>>x = linspace(0,pi,20);
>>y = linspace(-pi,pi,20);
>>dblquad('func',0,pi,-pi,pi)
ans =
   19.7392
>>[xx,yy] = meshgrid(x,y);
>>zz = func(xx,yy);
>>surf(xx,yy,zz)
>>xlabel('x')
```

```
>>ylabel('y')
```



شکل(۵-۸) منحنی $\sin(x).\cos(y)+1$

برای محاسبه انتگرال‌های سه‌گانه از دستور triplequad استفاده می‌شود. فرمت عمومی دستور به صورت زیر می‌باشد:

q=triplequad(fun,xmin,xmax,ymin,ymax,zmin,zmax)

به طور مثال:

ابتدا تابع زیر را در یک m-file نوشته و ذخیره کنید:

```
function f = integrnd(x,y,z)
f = y*sin(x)+z*cos(x);
```

سپس دستور زیر را در پنجره فرامین تایپ کنید:

```
Q = triplequad(@integrnd,0,pi,0,1,-1,1);
```

۵-۶-۴ پیدا کردن صفرهای توابع با استفاده fzero

تابع fzero یک جواب f(x)=0 را در همسایگی $X0=x_0$ یا در بازه $X0=[x_1,x_2]$ محاسبه می‌کند. فرمت این دستور به صورت زیر می‌باشد:

fzero('F',X0)

F نام فایل تابعی است که در یک علامت نقل قول قرار گرفته، یا تابعی است که با inline ایجاد شده است.

مثال: ریشه $y(x)=\cos(x)\sin^2(x)$ را در بازه [-1 -2.5] به‌دست آورید.

یک فایل تابعی ایجاد کرده و دستورات زیر را وارد کنید:

```
function y=f2(x)
y=cos(x).*sin(x).^2;
```

دستور زیر را درپنجره فرامین واردکنید و پاسخ را مشاهده کنید:

```
fzero('f2',[-2.5 -1])
ans =
   -1.5708
```

۵-۶-۵ حل معادلات دیفرانسیل معمولی با ode45

با استفاده از تابع ode45، می‌توان حل عددی یک سیستم با n معادله دیفرانسیل معمولی مرتبه اول بـه شـکل زیر را، در بازه $t_1 \le t \le t_2$ به دست آورد.

$$\frac{dy_i}{dt} = f(t, y_1, y_2, ...y_n) \ , y_i(t_0) = a_i$$

که در آن a_i عددی ثابت است. فرمت تابع ode45 به‌صورت زیر می‌باشد:

[t,y]=ode45('F',[t0 ,tf],[a1,a2,…an])

F نام فایل تابعی است که در یک علامت نقل قول قرار گرفته، یا تابعی است که با inline ایجاد شده است. خروجی t یک بردار ستونی برحسب زمان است که توسط ode45 تعیین می‌شود و خروجی y ماتریس جواب‌ها است به‌گونه‌ای که سطرها مطابق زمان t و ستون‌ها برابر با جواب‌ها می‌باشند. آرگومان دوم ode45 یک بردار دوعضوی می‌باشد که زمان شروع و پایان عددی به‌دست آمده را ارائه می‌کند.

نکته: حلگر ode45 براساس تابع صریح Rung-Kutta عمل می‌کند و مقدار متغیرحالت در زمان فعلی، توسط مقدار متغیرحالت در گام زمانی قبلی محاسبه می‌شود. این حلگر، حلگر مناسبی برای حل بسیاری از مسائل می‌باشد. چند تابع حلگر معادله دیفرانسیل معمولی دیگر در MATLAB وجود دارد که هر کدام فواید خاص خود را دارند که عبارتند از: ode23، ode113، ode15s، ode23s و ode23tb. به‌طور مثال حلگر ode23 براساس تابع صریح Rung-Kutta می‌باشد ولی نسبت به ode45 سریعتر بوده و دقت کمتری دارد و یا حلگر ode15s براساس مشتق‌گیری عددی و زمان‌هایی که ode45 اجرا نمی‌شود یا دقت قابل قبولی ندارد، مورد استفاده قرار می‌گیرد. فرمت استفاده از این توابع همانند ode45 است.

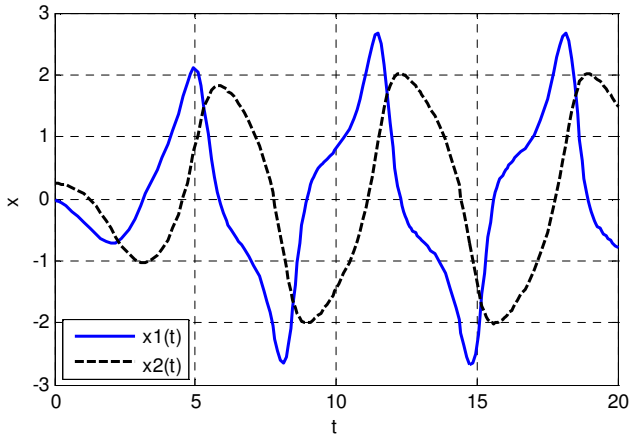
مثال: معادلات $\begin{cases} x_1' = x_1(1-x_2^2) - x_2 \\ x_2' = x_1 \end{cases}$ را با شرایط اولیه $x_0 = [0\ 0.25]$ در بازه زمانی [0,20] با استفاده از ode45 حل کنید.

یک فایل تابعی ایجاد کرده و دستورات زیر را وارد کنید:

```
function xdot=eq2(t,x)
xdot=zeros(2,1)
xdot(1)=x(1).*(1-x(2).^2)-x(2);
xdot(2)=x(1)
```

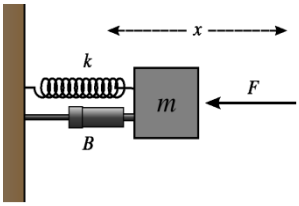
دستورات زیر را درپنجره فرامین واردکنید:

```
>> x0=[0 0.25];
>> [t,x]=ode45('eq2',[0 20],x0)
>> plot(t,x)
```



شکل(۵-۹) منحنی x_1, x_2

مثال: یک سیستم جرم و فنر و دمپر تحت نیروی F را در نظر بگیرید. معادله حرکت جرم به‌صورت زیر است:



$$m\ddot{x} + B\dot{x} + kx = F(t)$$

برای این سیستم شرایط زیر را در نظر می‌گیریم:

$$B = 2m\zeta\omega_n,\ k = 100N/m\ ,m = 2kg,\ w_n = \sqrt{k/m},\ \zeta = 1$$

$$x(0) = 0.1,\ \dot{x}(0) = 0.2$$

معادله دیفرانسیل معمولی مرتبه دوم را به معادلات دیفرانسیل مرتبه اول تبدیل می‌کنیم:

$$\begin{cases} x_1 = x \\ x_2 = \dfrac{dx}{dt} \end{cases} \Rightarrow \begin{cases} x_1' = x_2 \\ x_2' = \dfrac{1}{m}(F(t) - kx_1 - Bx_2) \end{cases}$$

حال به سیستم بالا نیروهای مختلف اعمال می‌کنیم:

$$F(t) = 0,\ F(t) = 1,\ F(t) = 5\sin(t),\ F(t) = 2t\ \ (t \le 2)$$

یک فایل تابعی ایجاد کرده و دستورات زیر را وارد کنید:

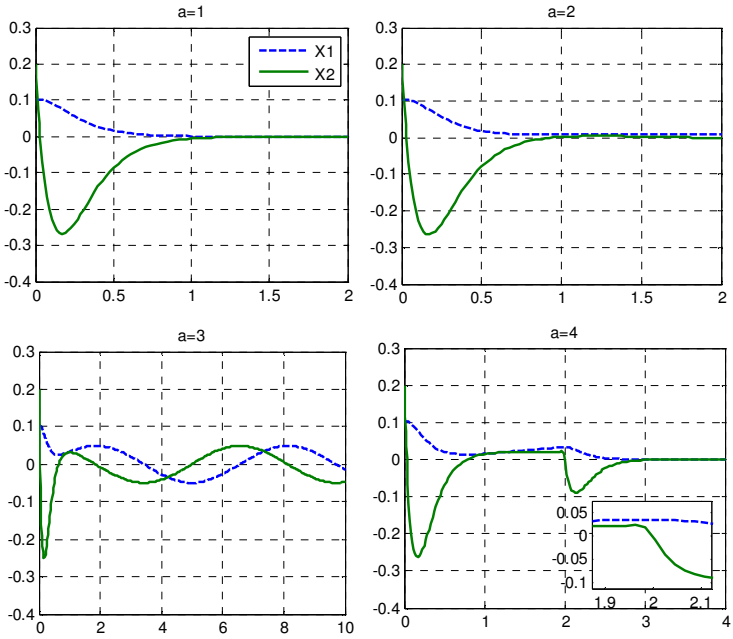
```
function xdot = mass_spring(t,x)
m=2;k=100;zeta=1;wn=sqrt(k/m);B=2*m*zeta*wn;a=1;
switch a
    case 1
```

```
        xdot=[x(2);1/m*(-k*x(1)-B*x(2))];
    case 2
        xdot=[x(2);1/m*(1-k*x(1)-B*x(2))];
    case 3
        xdot=[x(2);1/m*(sin(t)-k*x(1)-B*x(2))];
    otherwise
        f=2*t.*(t<=2);
        xdot=[x(2);1/m*(f-k*x(1)-B*x(2))];
end
```

برای اعمال نیروهای مختلف به جرم m، کافی است که مقدار a را تغییر دهید.

دستورات زیر را درپنجره فرامین واردکنید:

```
[tt,xx]=ode45('mass_spring',[0 2],[0.1;0.2]);
plot(tt,xx)
```



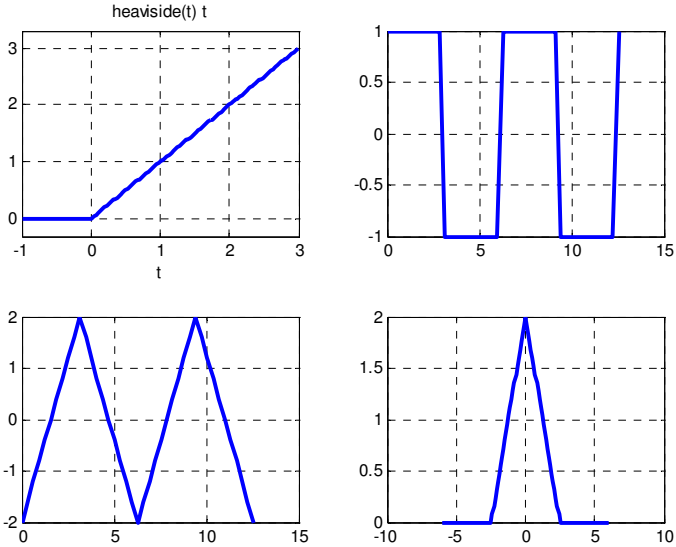
شکل(۵-۱۰) منحنی تغییر موقعیت جرم تحت شرایط مختلف

۵-۷مسائل مهندسی برق

مثال۱- برنامه‌ای بنویسید که موج های زیر را تولید کند:

- تولید موج شیب دار t*u(t)
- تولید موج مربعی
- تولید موج دندانه اره‌ای
- تولید موج مثلثی با پیک ۲ و طول ۵
- تولید موج مستطیلی با حداکثر مقدار ۲ و عرض ۶

```
syms t
ramp=('heaviside(t)'*t)
subplot(2,2,1)
ezplot(ramp,[-1 3])

subplot(2,2,2)
t=0:0.1*pi:4*pi;
f1=square(t,50) %f1(t)
plot(t,f1)

subplot(2,2,3)
f2=2*sawtooth(t,0.5)%  50% period T
plot(t,f2)

t=-6:0.1:6;
subplot(2,2,4)
f3=2*tripuls(t,5);
plot(t,f3)
f4=2*rectpuls(t,6);
```



مثال۲- با استفاده از دستور gensim ، سه دوره کامل موج مربعی با پریود T=3 و سرعت نمونه برداری Ts=0.1 تولید کنید.

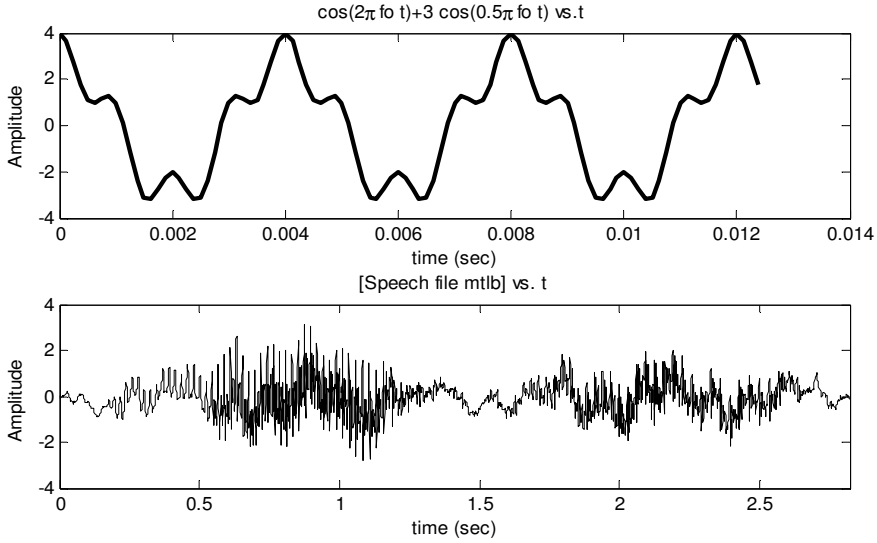
[squar,t]=gensim('squar',3,9,0.1)

تولید قطار پالس مثلثی با دستور pulsetran

```
t=0:0.001:1;
d1=[0:0.33:1]  %  3 cycle
Y1=pulsetran(t,d1,'tripulse',0.25)
```

مثـــال ۳- برنامـــه زیـــر ســـیگنال صـــوتی سینوســـی $y=\cos(2\pi f0t)+3\cos(0.5*\pi f0t)$ را کـــه در آن $f0=1000Hz$ و فرکانس نمونه $Fs=8000HZ$ است، رسم می‌کند. از طرفی یـک فایـل صـوتی را کـه شـامل صحبت یک نفر است رسم می‌کند.

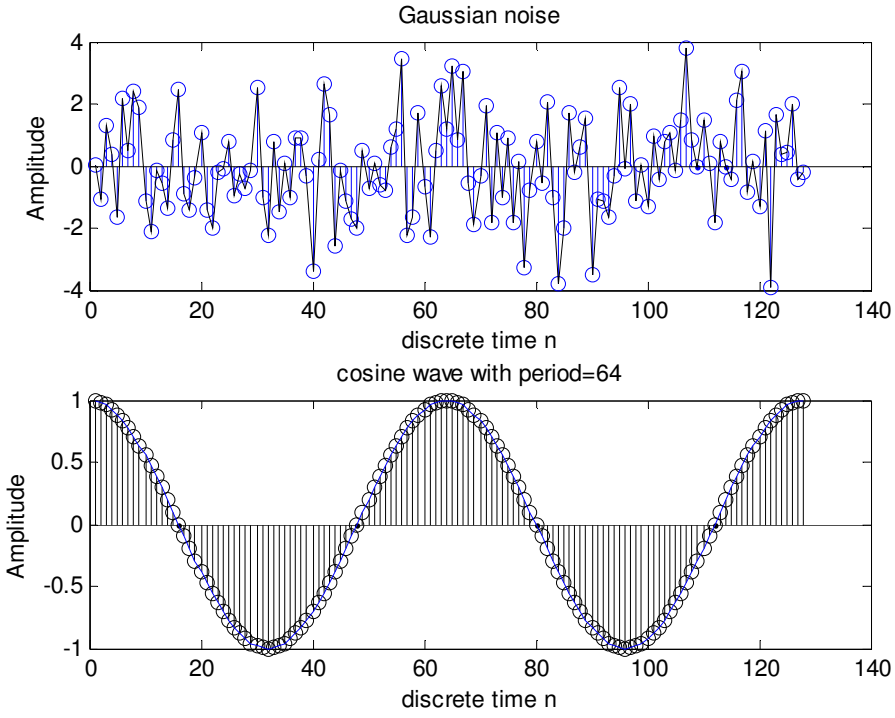
```
fo = 1000;
Fs = 8000;Ts =1/Fs;
t = 0:Ts:1;
y = cos(2*pi*fo.*t)+3*cos(0.5*pi*fo.*t); % audio sequence

subplot(2,1,1)
plot(t(1:100),y(1:100));
ylabel('Amplitude');
xlabel('time (sec)');
title('cos(2\pi fo t)+3 cos(0.5\pi fo t) vs.t ')

subplot(2,1,2)
load mtlb;
Fs=1418; T=1/Fs;
x = 1:4001;xx=x.*T; % speech file
plot(xx,mtlb);
axis([0 4000*T -4 4]);
ylabel('Amplitude ');
title('[Speech file mtlb] vs. t');
xlabel('time (sec)')
```

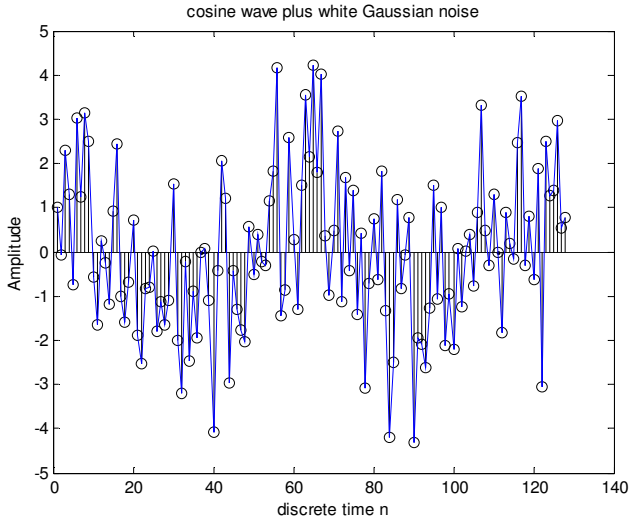


مثال۴- برنامه زیر نمودار نویز سفید گوسین با قـدرت 2.7watt، سـیگنال $y=\cos(2n\pi/64)$ و یـک مـوج خراب شده با نویز سفید با قدرت 2.7watt را رسم می‌کند.

```
n = 1:128; P =2.7;
```

```
figure(1)
subplot(2,1,1)
white_noise = sqrt(P)*randn(1,128);
stem(n,white_noise); hold on;plot(n,white_noise);
xlabel('discrete time n');
ylabel('Amplitude'); title('Gaussian noise');

subplot(2,1,2)
signal = cos(2*pi*n/64);
stem(n,signal);hold on;plot(n,signal);
title('cosine wave with period=64')
xlabel('discrete time n');
ylabel('Amplitude');

figure(2)
signal_noise = white_noise+signal;
stem(n,signal_noise);hold on;plot(n,signal_noise);
title('cosine wave plus white Gaussian noise')
xlabel('discrete time n');
ylabel('Amplitude');
```

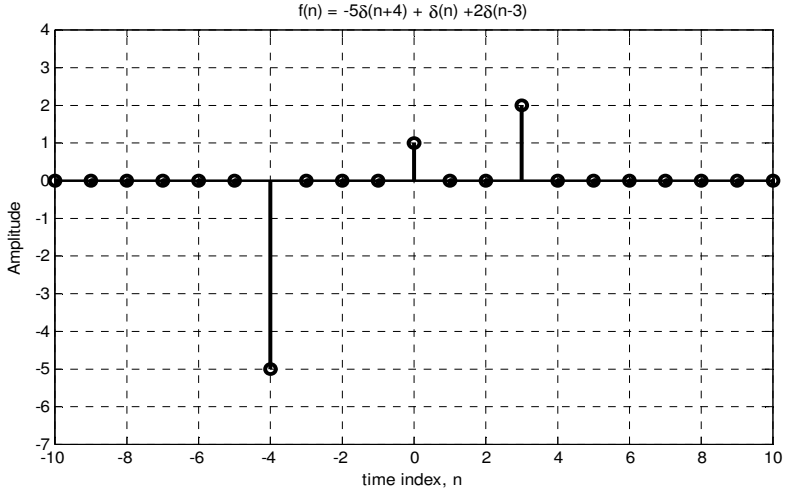




مثال ۵- برنامه‌ای بنویسید که تابع گسسته زیر را در بازه $-10 \le n \le 10$ رسم کند.

$$f(n) = -5\delta(n+4) + \delta(n) + 2\delta(n-3)$$

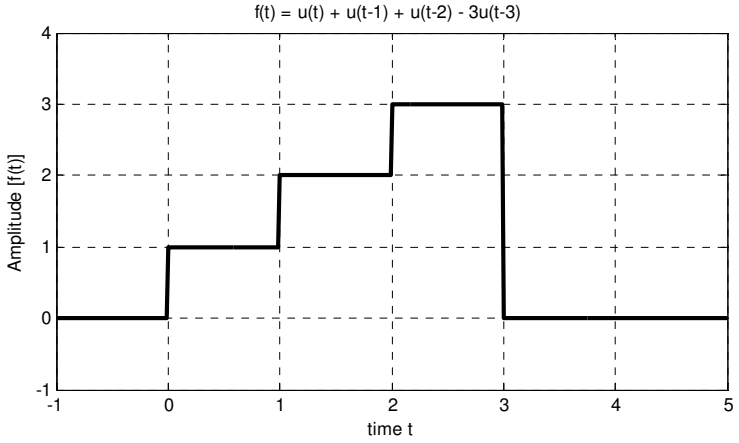
```
n = -10:10;
fn = [zeros(1,6) -5 zeros(1,3) 1 zeros(1,2) 2 zeros(1,7)];
yzero = zeros(1,21);

stem(n,fn) % plot the function f(n)
hold on;plot(n,yzero);
xlabel('time index, n')
ylabel('Amplitude')
axis([-10 10 -7 4])
title('f(n) = -5\delta(n+4) + \delta(n) +2\delta(n-3)')
```



مثال۶- برنامه ای بنویسید که سیگنال آنالوگ زیر را در بازه زمانی [5 -1] رسم کند.

$$f(t)=u(t)+u(t-1)+u(t-2)-3u(t-3)$$

```
t = -1:0.01:5;
t0 = 0;
ut = stepfun(t,t0); % u(t)
t0 =1;
ut1 =stepfun(t,t0); % u(t-1)
t0 = 2;
ut2 = stepfun(t,t0); % u(t-2)
t0 = 3;
ut3 = -3 * stepfun(t,t0); % -3u(t-3)
fn = ut+ut1+ut2+ut3;

plot(t,fn);
axis([-1 5 -1 4])
title('f(t) = u(t) + u(t-1) + u(t-2) - 3u(t-3) ')
xlabel('time t')
ylabel('Amplitude [f(t)]')
```



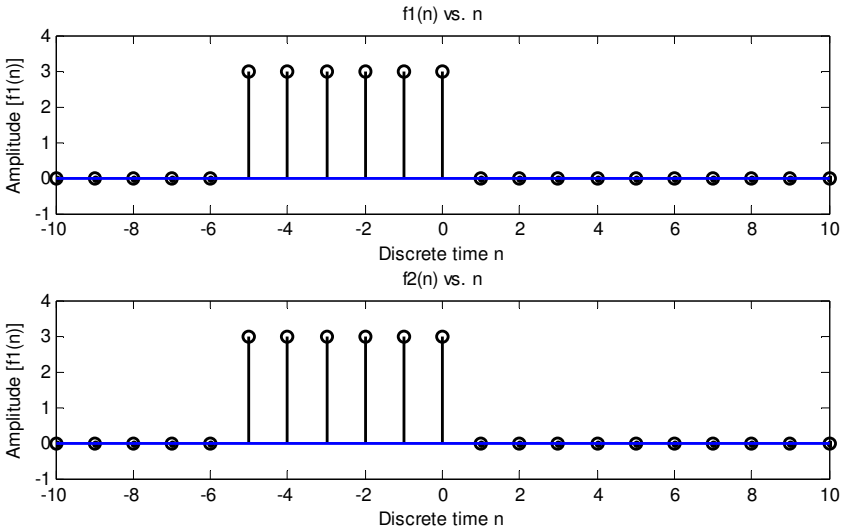
مثال۷- برنامه‌ای بنویسید که نمودارهای دو تابع f1(n),f2(n) را رسم کرده و نشان دهد که این دو تابع در بازه $-10 \le n \le 10$ با هم برابر هستند.

$$f_1(n)=3u(-n)\times u(n+5)$$

$$f_2(n)=3[u(n+5)-u(n+1)]$$

```
clc; clf;
yzero = [zeros(1,21)]; n=-10:10;
n0 = 0;un = stepfun(n, n0);
urev = fliplr(un);n1 = -5;
un_5 = stepfun(n, n1); fn1=3*(urev .* un_5);
```

```
subplot(2,1,1)
stem(n,fn1);hold on; plot(n,yzero)
title('f1(n) vs. n');
ylabel('Amplitude [f1(n)]'); xlabel('Discrete time n')
un1=stepfun(n,1); fn2=3*(un_5 -un1);
axis([-10 10 -1 4])

subplot(2,1,2)
stem(n, fn2);hold on; plot(n,yzero);
title('f2(n) vs. n');
ylabel('Amplitude [f1(n)]'); xlabel(' Discrete time n ');
axis([-10 10 -1 4])
```



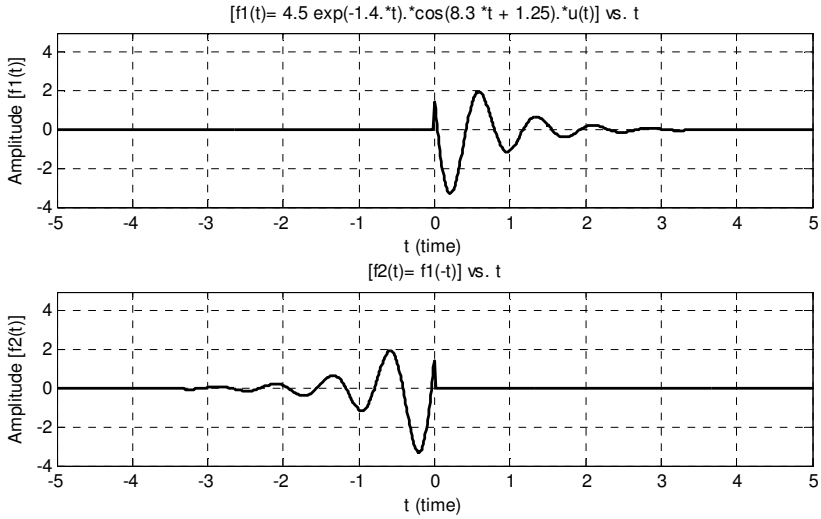
مثال ۸- برنامه‌ای بنویسید که توابع زمانی f1,f2 را در بازه $-5 \le t \le 5$ رسم کند.

$$f_1(t) = 4.5e^{-1.4t}\cos(8.3t + 1.25)u(t)$$

$$f_2(t) = f_1(-t)$$

```
t = -5:.01:5; t0 = 0;
u = stepfun(t,t0);
f1 = 4.5*exp(-1.4.*t).*cos(8.3.*t+1.25).*u;

subplot(2,1,1);
plot(t,f1);axis([-5 5 -4 5]);
title('[f1(t)= 4.5 exp(-1.4.*t).*cos(8.3 *t + 1.25).*u(t)]
vs. t');
ylabel('Amplitude [f1(t)]'); xlabel('t (time)');
f2 = fliplr(f1);
```

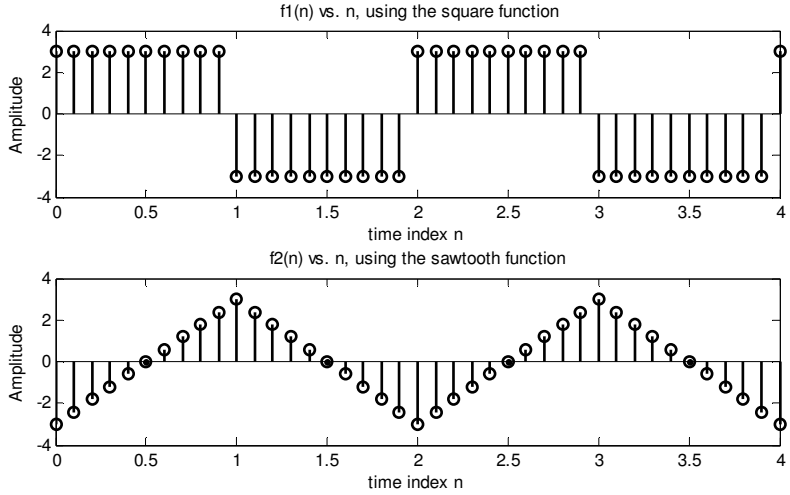
```
subplot(2,1,2);
plot(t,f2);axis([-5 5 -4 5]);
title('[f2(t)= f1(-t)] vs. t');
ylabel('Amplitude [f2(t)]'); xlabel('t (time)');
```



مثال۹: برنامه ای بنویسید که سیگنال‌های گسسته پریودیک زیر را با پریود T=2sec در رنج $0 \le n \le 4$ با توابع مربعی و دندانه اره‌ای رسم کند.

$$f_1(n)=\begin{cases}3 & n>0\\ -3 & <1n<2\end{cases} \qquad f_2(n)=\begin{cases}6n-3 & 0<n<4\\ -6n+9 & 1<n<2\end{cases}$$

```
n = 0:0.1:4; N = 2;
f1 = 3* square((2*pi*n/N),50);
f2 = 3* sawtooth((2*pi*n/N),0.5);

subplot(2,1,1);
stem(n,f1);
title(' f1(n) vs. n, using the square function')
xlabel('time index n');ylabel('Amplitude')

subplot(2,1,2)
stem(n,f2);
title('f2(n) vs. n, using the sawtooth function')
xlabel('time index n');ylabel('Amplitude');
```



مثال۱۰: برنامه‌ای بنویسید که سیگنال های زیر را در بازه $-5 \le t \le 6$ رسم کند.

$$f_1(t) = e^{-t}u(t)$$

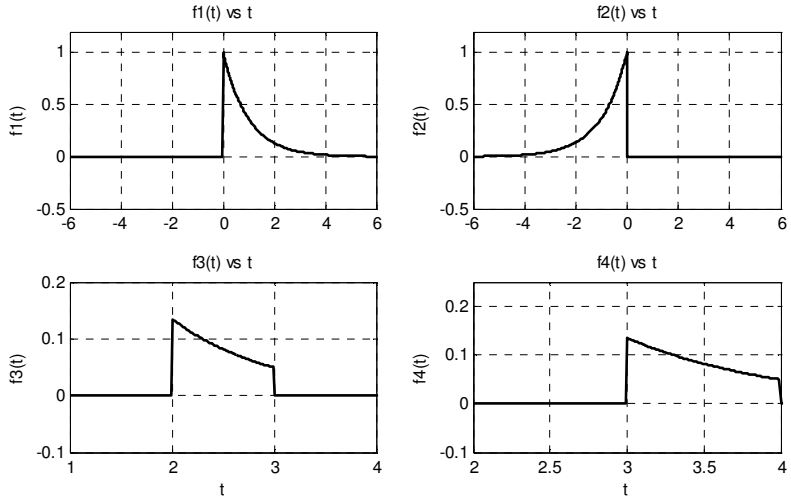
$$f_2(t) = f_1(-t)$$

$$f_3(t) = e^{-t}[u(t-2) - u(t-3)]$$

$$f_4(t) = f_3(t-1)$$

```
% Script file: analog _ plots
t = -6:0.01:6;
t0 = 0;
u0 = stepfun(t, t0);
f1 = exp(-t).*u0;
f2 = fliplr(f1);

subplot(2,2,1);
plot(t, f1);
title('f1(t) vs t');
ylabel('f1(t)')
axis([-6 6 -.5 1.2]);grid on

subplot(2,2,2)
plot(t, f2)
title('f2(t) vs t')
ylabel('f2(t)')
axis([-6 6 -.5 1.2]);grid on

subplot(2,2,3);
ut35 = stepfun(t, 2) - stepfun(t, 3);
f3 = exp(-t).*ut35;
plot(t, f3);grid on
```

```
axis([1 4 -.1 .2])
title('f3(t) vs t')
ylabel('f3(t)')
xlabel('t')

subplot(2,2,4);
t_1 = t+1;
plot(t_1, f3);grid on
title('f4(t) vs t')
ylabel('f4(t)')
axis([2 4 -.1 .25]);grid on
xlabel('t')
```



مثال ۱۱- برنامه‌ای بنویسید که نمودار توابع پریودیک زیر را رسم کند.

$$f_1(n)=\sum_{k=0}^{25}(-1)^k\,\delta(n-k),\ 0\le k\le 25$$

$$f_2(n)=\begin{cases}\sin(n\pi) & 0\le n\le 1\\ 0 & 1\le n\le 2\end{cases}\qquad in\ 0\le n\le 6$$

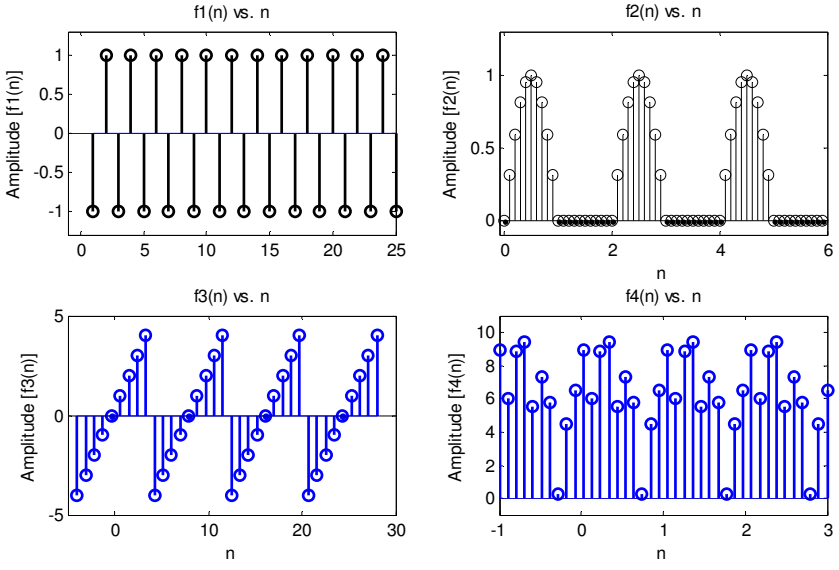
$$f_3(n)=n\qquad -4\le n\le 4$$

تابع $f_3(n)$ در بازه $-5\le n\le 30$ رسم شود.

تابع پریودیک رندوم $f_4(n)$ با دوره T=1 در بازه $-1\le n\le 3$

```
n=1:25;
f1n = [(-1).^n.*ones(1,25)];
yzero = zeros(1,25);
subplot(2,2,1);
stem(n,f1n);hold on; plot(n,yzero);
ylabel('Amplitude [f1(n)]');xlabel(n');
axis([-1 25 -1.3 1.3])
```

```
title ('f1(n) vs. n');

% part (b)
nn = 0:0.1*pi:0.9*pi;
Y = sin(nn);
f2n= [Y zeros(1,10) Y zeros(1,10) Y zeros(1,10)]; m=0:.1:5.9;
subplot(2,2,2);stem(m, f2n);axis([-.1 6 -.1 1.3]);
ylabel('Amplitude [f2(n)]'); xlabel('n');
title('f2(n) vs. n');

% part c
f3 = -4:1:4; f3n = [f3 f3 f3 f3];
nnn = linspace(-4,28,36);
subplot(2,2,3); yzer =zeros(1,length(nnn));
stem(nnn, f3n);hold on; plot(nnn,yzer);
xlabel('n');axis([-5 30 -5 5]);
ylabel('Amplitude [f3(n)]');
title('f3(n) vs. n')

% part d
f4 = 10*rand(1,10) ;f4n=[f4 f4 f4 f4];
n4 =linspace(-1,3,40);
subplot (2,2,4);yze=zeros(1,length(n4));
stem(n4,f4n);hold on; plot(n4,yze);
axis ([-1 3 -1 11]);xlabel('n');
ylabel('Amplitude [f4(n)]');
title('f4(n) vs. n')
```



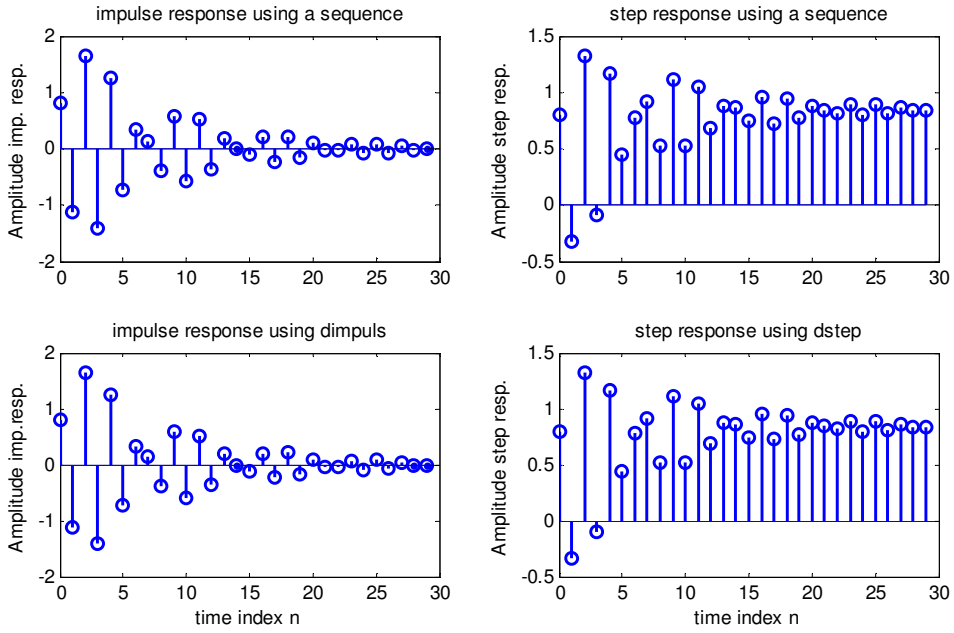
مثال ۱۲- تابع تبدیل گسسته زیر را در نظر بگیرید:

$$H(2) = \frac{0.8 - 0.45z^{-1} + 0.35z^{-2} + 0.01z^{-3}}{1 + 0.85z^{-1} - 0.43z^{-2} - 0.58z^{-3}}$$

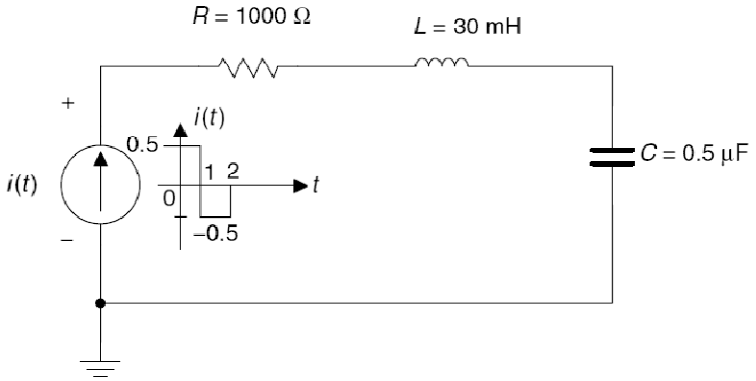
الف- برنامه‌ای بنویسید که پاسخ پله و ضربه گسسته تابع تبدیل را با ورودی‌های دنباله‌ای ضربه و پله رسم کند.

ب- قسمت (الف) را با دستورات dstep و dimpulse انجام دهید.

```
n = 30;
P = [0.8 -0.45 0.35 0.01];Q = [1 0.85 -0.43 -0.58];
I = [1 zeros(1, n-1)]; % impulse sequence
S = [ones(1, n)] ; % step sequence

figure(1)
subplot(2,2,1);
Y1 = filter(P, Q, I); n=0:29;
yzero=zeros(1,30);
stem(n, Y1);hold on;
plot (n,yzero);
title('impulse response using a sequence');
ylabel ('Amplitude imp. resp.');

subplot (2,2,2);
Y2 = filter(P,Q,S);
stem (n, Y2);hold on; plot(n,yzero);
title('step response using a sequence');
ylabel('Amplitude step resp.');

subplot(2,2,3);
Y3 = dimpulse(P, Q, n);
stem(n, Y3);hold on; plot(n,yzero);
title('impulse response using dimpuls');
ylabel ('Amplitude imp.resp.');
xlabel ('time index n');

subplot (2,2,4);
Y4 = dstep(P, Q, n);
stem(n, Y4); hold on; plot(n,yzero);
title('step response using dstep');
ylabel('Amplitude step resp.');
xlabel('time index n')
```



مثـــال ۱۳- مـــدار RLC نشـــان داده شـــده در شـــکل زیـــر را وقتـــی منبـــع جریـــان برابـــر $i(t)=(\frac{2}{\pi})\sum_{n=odd}^{15}(\frac{1}{n})\sin(n\pi t)$ است، تحلیل کنید.



معادله منبع جریان i(t)، نمایش دهنده سری فوریه تقریبی موج مربعی پریودیک می باشد. برنامه‌ای بنویسید که نمودارهای زیر را رسم کند.

۱. جریان i(t) نسبت به زمان t

۲. $\int i(t)$ نسبت به زمان که تقریبی از موج مثلثی پریودیک است.

۳. $v_L(t) = L\frac{di}{dt}$ نسبت به زمان t

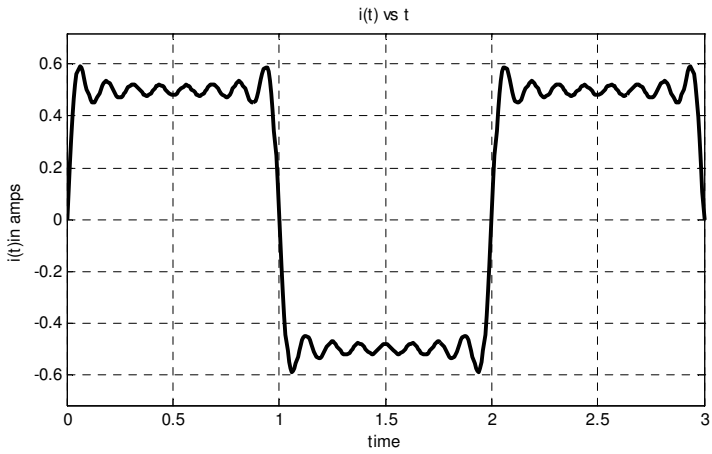
۴. $v_c(t) = \frac{1}{c}\int i(t)dt$ نسبت به زمانt

۵. $P_R(t)$ نسبت به زمان t که $P_R(t) = i^2(t) \times R$

۶. $P_L(t)$ نسبت به زمان t که $P_L(t) = i_L(t) \times v_L(t)$

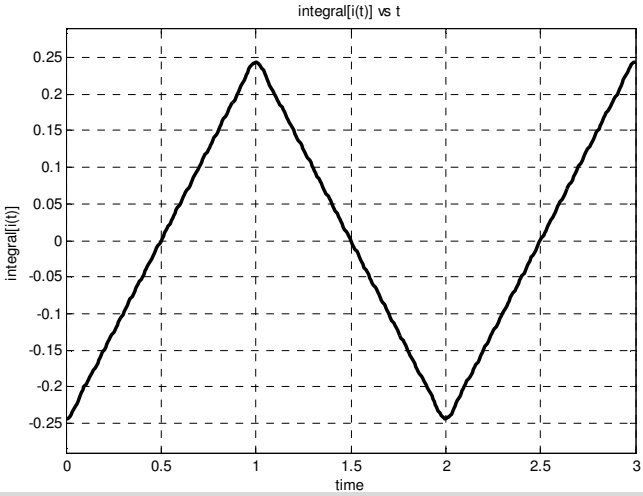
۷. $V_L(t) + V_R(t)$ نسبت به زمان t

```
close  all
clear all
clc
% Analysis of RLC Series circuit
% with non DC source
syms t;
H1 = 2/pi*sin(pi*t);
H3 = 2/(3*pi)*sin(3*pi*t);
H5 = 2/(5*pi)*sin(5*pi*t);
H7 = 2/(7*pi)*sin(7*pi*t);
H9 = 2/(9*pi)*sin(9*pi*t);
H11 = 2/(11*pi)*sin(11*pi*t);
H13 = 2/(13*pi)*sin(13*pi*t);
H15 = 2/(15*pi)*sin(15*pi*t);
iamps = H1+H3+H5+H7+H9+H11+H13+H15;

figure(1)
ezplot(iamps,[0,3])
xlabel(' time ');ylabel('i(t)in amps');
title('i(t) vs t');grid on;
```

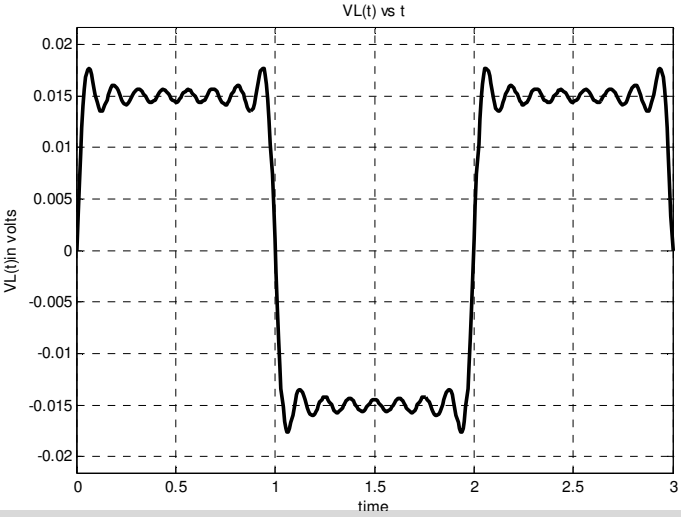



```
figure(2)
inti = int(iamps);
```

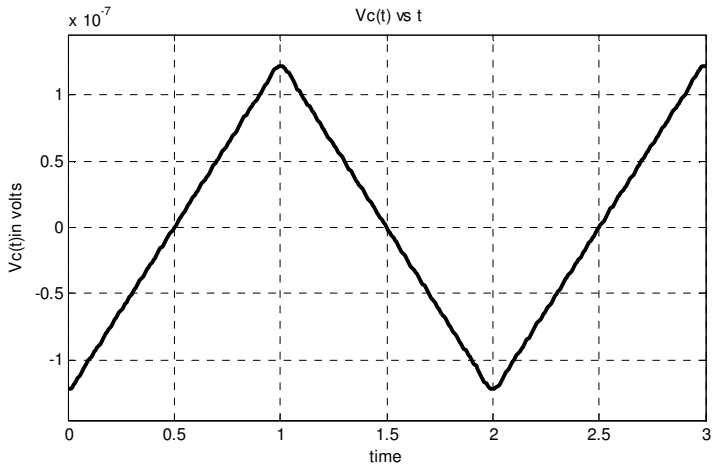
```
ezplot (inti,[0,3])
xlabel(' time ');ylabel('integral[i(t)]');
title('integral[i(t)] vs t');grid on;
```



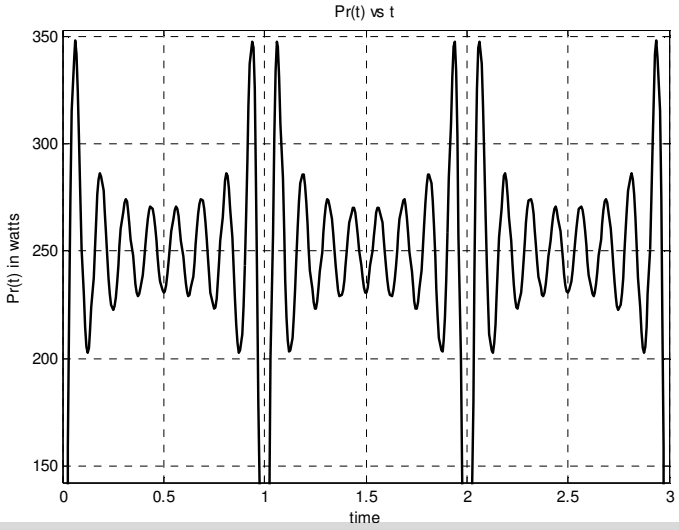

```
figure(3)
Vl = 30e-3*diff(inti);
ezplot(Vl,[0,3])
xlabel(' time ');ylabel('VL(t)in volts');
title('VL(t) vs t');grid on;
```



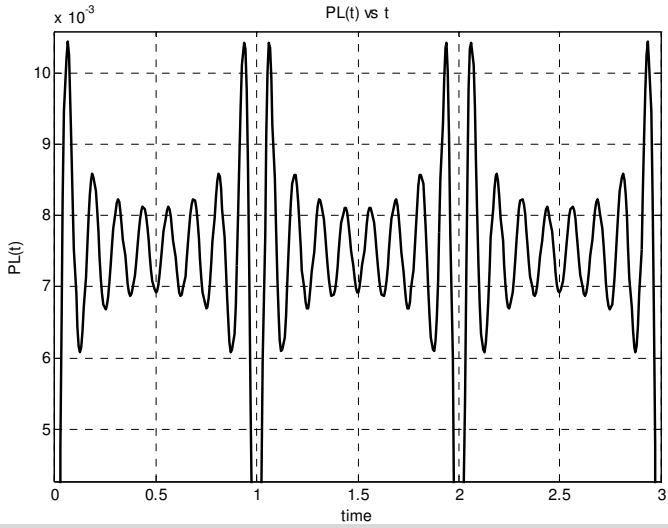

```
figure(4)
Vc=.5e-6*inti;
ezplot(Vc,[0,3])
title('Vc(t) vs t')
xlabel(' time ');ylabel('Vc(t)in volts')
title(' Vc(t) vs t');grid on;
```



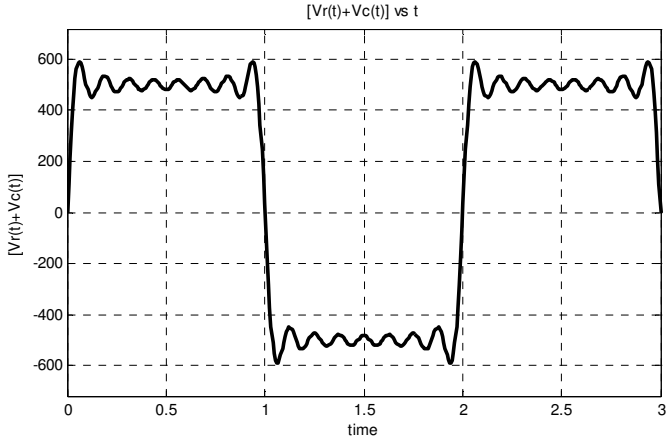

```
figure(5)
Vr=1000*iamps;
Pr = iamps*(Vr);
ezplot(Pr,[0,3])
title('Pr(t) vs t')
xlabel(' time ')
ylabel('Pr(t) in watts');grid on;
```



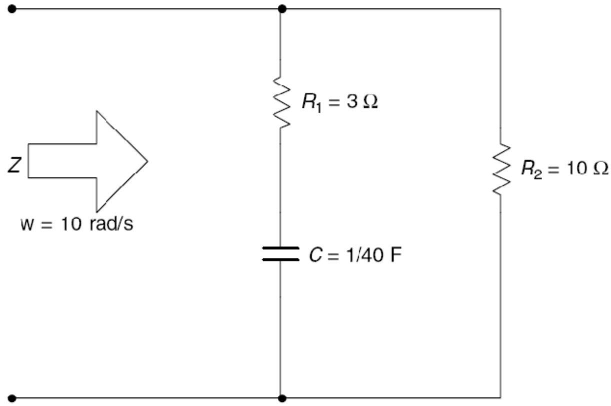

```
figure(6)
Pl = iamps*Vl;
ezplot(Pl,[0,3])
title('PL(t) vs t'); xlabel(' time ');
ylabel('PL(t)');grid on
```




```
figure(7)
Vcr=Vc+Vr;
ezplot(Vcr,[0,3])
title('[Vr(t)+Vc(t)] vs t')
xlabel(' time ');ylabel('[Vr(t)+Vc(t)]');
title('[Vr(t)+Vc(t)] vs t');grid on
```



مثال ۱۴- برنامه ای بنویسید که مقدار امپدانس Z مدار نشان داده شده در شکل را برای w=10rad/s محاسبه کند.




```
W = 10;
R1 = 3;
R2 = 10;
C = 1/40;
XC = -j/(W*C);
X = R1+XC;
Z = X*R2/(X+R2);
Zmag = abs(Z);
Phase = angle(Z);
Phase_deg = Phase*360/(2*pi);
disp('^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^')
disp('Magnitude of Z (in Ohms) and phase (in degrees) is
given by :');
Zmag
Phase_deg
disp('^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^')
```

مثال۱۵- با استفاده از دستورات MATLAB تبدیل فوریه و عکس تبدیل فوریه توابع زیر را به‌دست آورید.

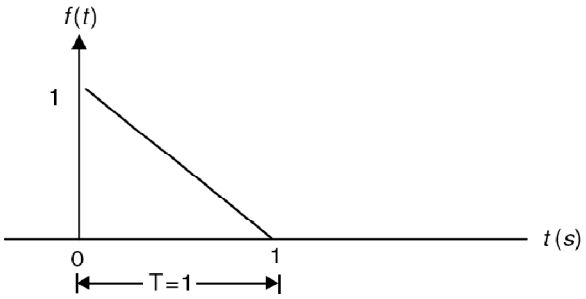
$$1-f_1(t)=\cos(3t+\frac{\pi}{4})\rightarrow \text{fourier}(\cos(3*t+pi/4))$$

$$2-f_2=te^{-t}u(t)\rightarrow \text{fourier}(t*\exp(-t)*\text{sym}('\text{heaviside}(t)'),t,w)$$

$$3-F_1=\frac{2}{1+w^2}\rightarrow \text{ifourier}(2/1+w^2),w,t)$$

$$4-F_2=\pi\times \text{Dirac}(w)-\frac{i}{w}\rightarrow \text{simplify}(\text{ifourier}(pi*\text{dirac}(w)-\frac{i}{w}),w,t)$$

مثال۱۶- تابع مثلثی پریودیک $f(t)=\frac{1}{\pi}\sum_{n=1}^{\infty}\frac{1}{n}\sin(n\omega_0 t)$ در شکل زیر در پریود ($\omega_0=2\pi$) T=1 نشان داده شده است.

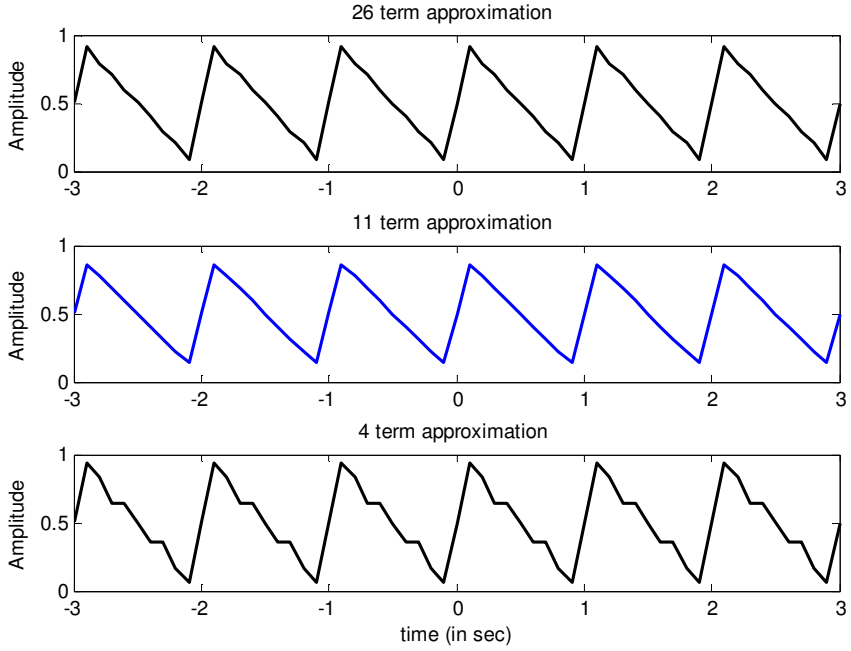


۱) برنامه ای بنویسید که تابع f(t) تقریبی را با استفاده از ۴، ۱۱ و ۲۶ ترم اول سری فوریه تابع در محدوده $-3 \le t \le 3$ رسم کند.

۲) برای هریک تقریب های قسمت اول مقادیر f(t) را در نقاط t=0,t=0.1 بدست آورید.

```
close all
clear all
clc
%Script file : Fourier _ approx
t =-3:0.1:3;
x25=0;
for n=1:1:25;
    xn25=1/(pi*n)*sin(2*pi*n*t);
    x25=x25+xn25; % sum of 25 terms
end;
ft26=0.5+x25; % add the DC

figure(1)
subplot(3,1,1);plot(t,ft26)
title('26 term approximation ')
ylabel('Amplitude')
x10=0;
for n=1:1:10;
    xn10=1/(pi*n)*sin(2*pi*n*t);
    x10=x10+xn10; % sum of 10 terms
end;
x3=0;
for n=1:1:3;
    xn3 =1/(pi*n)*sin(2*pi*n*t);
    x3 = x3+xn3; % sum of 3 terms
end;
ft4 = 0.5+x3; % add DC
ft11 = 0.5+x10;

subplot(3,1,2);plot(t,ft11);
ylabel('Amplitude');
title('11 term approximation ')
```

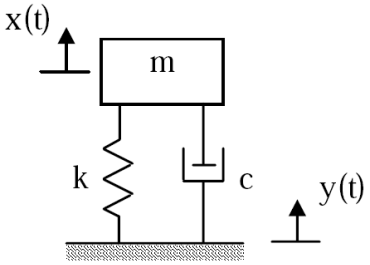
```
subplot(3,1,3);
plot(t,ft4);ylabel('Amplitude')
xlabel('time (in sec)')
title('4 term approximation ')
minft26 = min(ft26); maxft26 = max(ft26);
minft11 = min(ft11); maxft11 = max(ft11);
minft4 = min(ft4);maxft4 = max(ft4)
A=[minft26 maxft26];

disp('******************************************')
disp(' *********R E S U L T S ***************')
disp('******************************************');
disp ('The magnitude of f(t) using 26 terms at t = 0, and t =
0.1 are:');
disp (A);
B=[minft11 maxft11];C=[minft4 maxft4];
disp ('The magnitude of f(t) using 11 terms at t = 0, and t =
0.1 are:');
disp (B)
disp ('The magnitude of f(t) using 4 terms at t = 0 , and t =
0.1 are:');
disp (C);
disp('******************************************');
```



۵–۸ مسائل مهندسی مکانیک

مثال۱: آنالیز ارتعاشی- ارتعاشات تحت تحریک هارمونیک پایه

سیستم با یک درجه آزادی زیر را در نظر بگیرید. در این سیستم، پایه به صورت هارمونیک حرکت می کند.



برنامه‌ای بنویسید که مقادیر $\frac{|X(iw)|}{A}$ و زاویه فاز $\varphi(w)$ را بر حسب نسبت فرکانسی $r=\frac{w}{w_n}$ رسم کند.

$$\frac{|X(i\omega)|}{A}=\left[\frac{1+(\frac{2\zeta\omega}{\omega_n})^2}{1-(\frac{\omega}{\omega_n})^2+(\frac{2\zeta\omega}{\omega_n})^2}\right]^{\frac{1}{2}}$$

$$r=\frac{\omega}{\omega_n}$$

$$\varphi(\omega)=\tan^{-1}\left[\frac{2\zeta(\frac{\omega}{\omega_n})^3}{1-(\frac{\omega}{\omega_n})^2+(\frac{2\zeta\omega}{\omega_n})^2}\right]$$

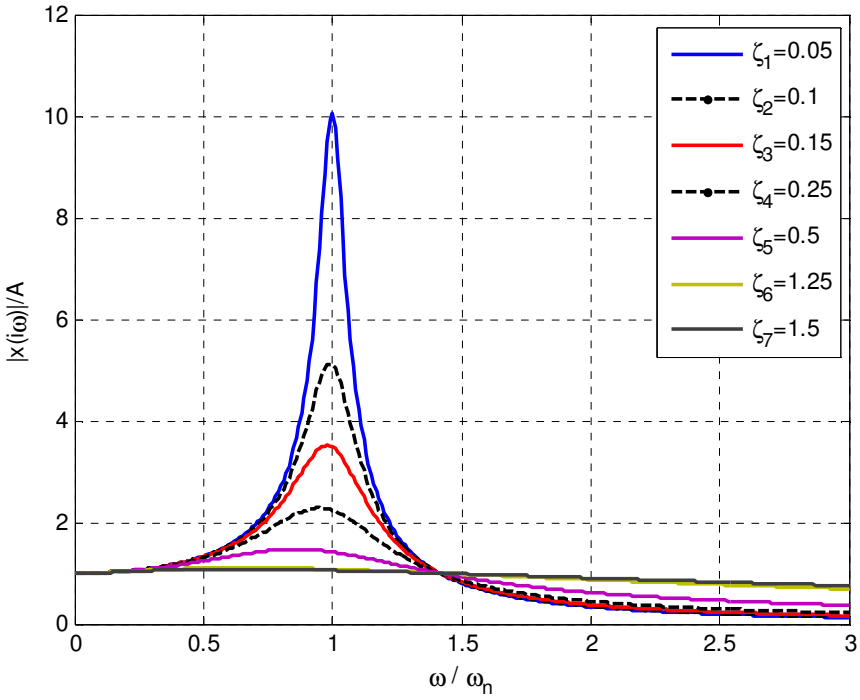
$$x(t)=X(i\omega)e^{i\omega t}$$

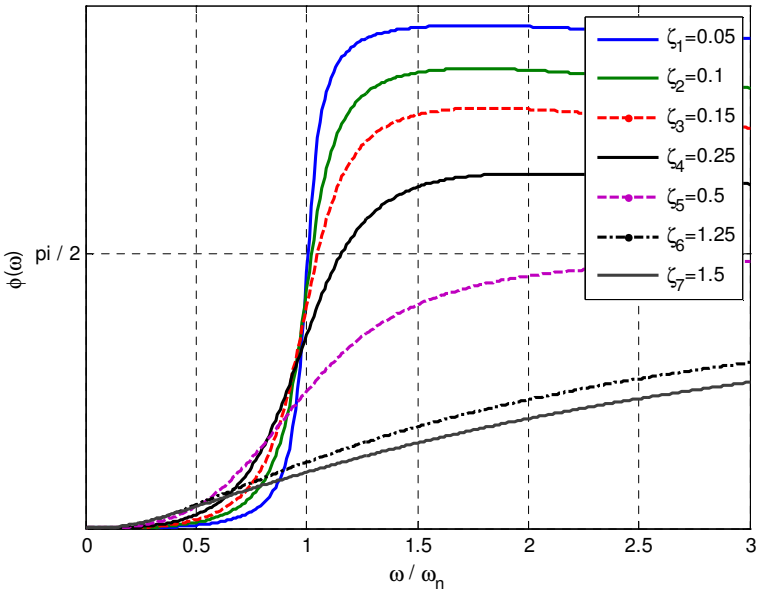
برنامه زیر را در یک m-file وارد کرده و اجرا کنید.

```
close all
clear all
clc
zeta = [0.05;0.1;0.15;0.25;0.5;1.25;1.5];% damping factors
r= 0:0.01:3;% frequency ratio
for k = 1:length(zeta)
    G(k,:)          =          sqrt((1+(2*zeta(k)*r).^2)./((1-
r.^2).^2+(2*zeta(k)*r).^2));
    phi(k,:) =atan2(2*zeta(k)*r.^3,1-r.^2+(2*zeta(k)*r).^2);
end
```

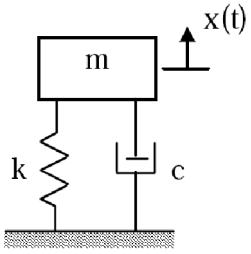
```
figure(1)
plot(r,G,'linewidth',2.2)
xlabel('\omega / \omega_n')
ylabel('|x(i\omega)|/A')
grid
legend('\zeta_1=0.05','\zeta_2=0.1','\zeta_3=0.15','\zeta_4=0
.25','\zeta_5=0.5','\zeta_6=1.25','zeta_7=1.5')

figure(2)
plot(r,phi,'linewidth',2.2)
xlabel('\omega / \omega_n')
ylabel('\phi(\omega)')
grid
legend('\zeta_1=0.05','\zeta_2=0.1','\zeta_3=0.15','\zeta_4=0
.25','\zeta_5=0.5','\zeta_6=1.25','\zeta_7=1.5')

ha=gca;
set(ha,'ytick',[0:pi/2:pi])
set(ha,'yticklabel',{[];'pi / 2';'p'})
```

خروجی برنامه به صورت زیر خواهد بود:





مثال۲: پاسخ سیستم یک درجه آزادی میرا با شرایط جابه‌جایی اولیه و سرعت اولیه به صورت زیر است:



$$x(t) = ce^{-\xi\omega_n t}\cos(\omega_d t - \varphi)$$

که در این رابطه، c دامنه و φ زاویه فاز پاسخ می‌باشد. مقادیر c و φ برابر است با:

$$c = \sqrt{x_0^{\,2} + \left(\frac{\xi\omega_n x_0 + v_0}{\omega_d}\right)^2} \qquad \varphi = \tan^{-1}\left(\frac{\xi\omega_n x_0 + v_0}{\omega_d x_0}\right)$$

$$\omega_d = \sqrt{1-\xi^2}\,\omega_n$$

برنامه‌ای در MATLAB بنویسید که پاسخ سیستم را با شرایط زیر رسم کند.

$$\omega_n = 5\frac{rad}{\sec},\ x(0) = 0,\ v_0 = 60\frac{cm}{s}$$

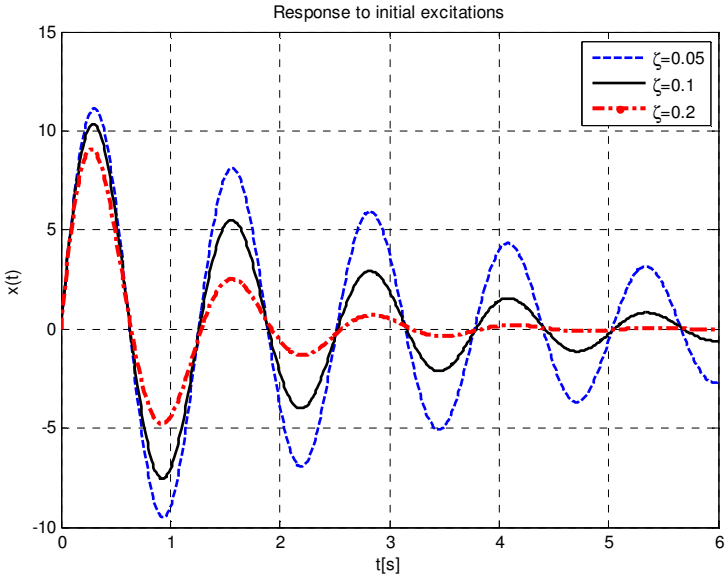
$$\zeta_1 = 0.05, 0.1, 0.2$$

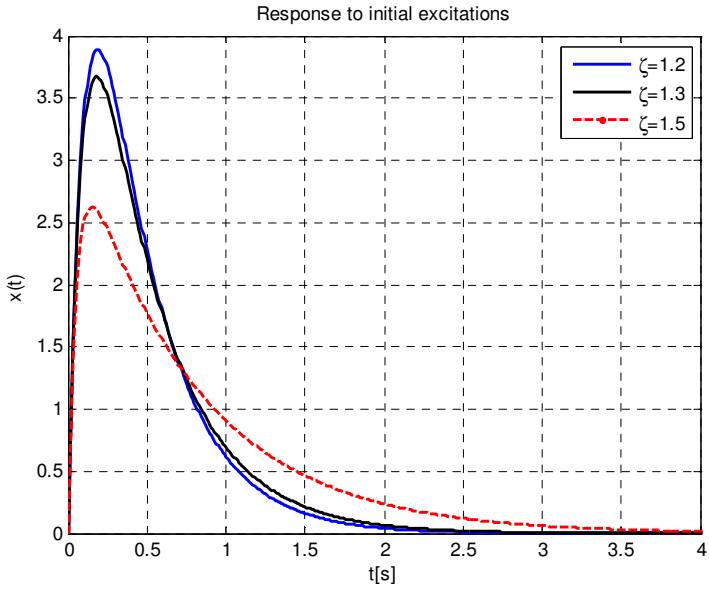
$$\zeta_2 = 1.3, 1.5, 2$$

برنامه زیر را در یک m-file وارد کرده و اجرا کنید.

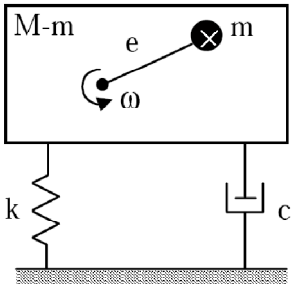
```
clear all
close all
clc

wn=5;% Natural frequency
zeta=[0.05;0.1;0.2];% Damping ratio
x0=0; % Initial displacement
v0=60; % Initial velocity
t0=0; % Initial time

deltat=0.01; % Time step
tf=6; % Final time
t=[t0:deltat:tf];
for i=1:length(zeta)
    wd=sqrt(1-zeta(i)^2)*wn; % Damped frequency
    x=exp(-
zeta(i)*wn*t).*(((zeta(i)*wn*x0+v0)/wd)*sin(wd*t)+x0*cos(wd*t
));
    plot(t,x,'linewidth',2.2)
    hold on
end
title('Response to initial excitations')
xlabel('t[s]')
ylabel('x(t)')
```

خروجی برنامه به صورت زیر خواهد بود:





مثال ۳: سیستم زیر را با یک جرم نامتعادل کننده دوار در نظر بگیرید. برنامه‌ای بنویسید که بزرگی پاسخ فرکانسی سیستم زیر را رسم نماید.

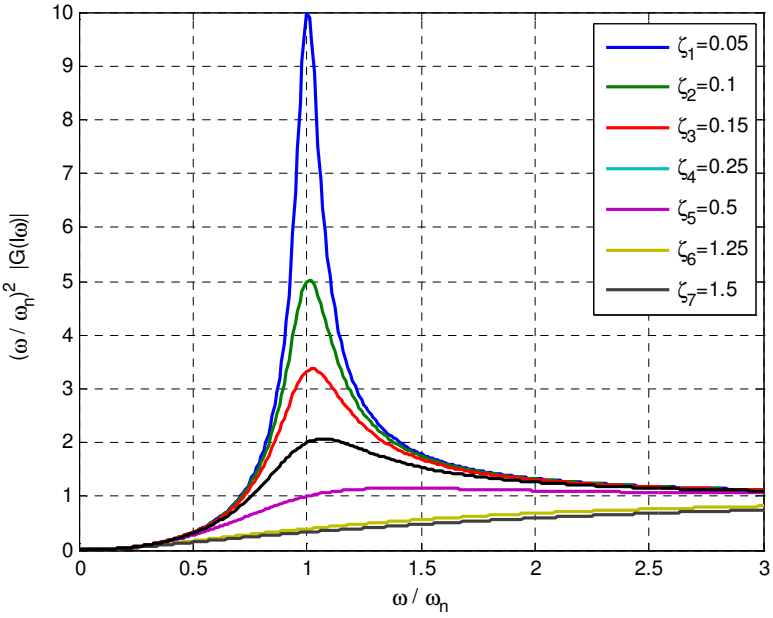


ξ=[0.05 0.01 0.15 0.2 0.25 0.3 1.0 1.5]

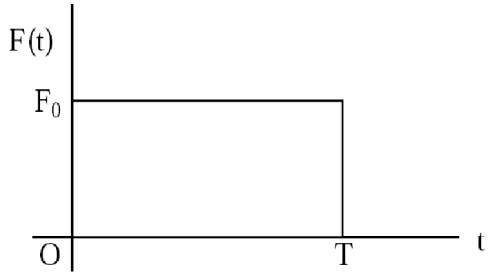
$r = \frac{\omega}{\omega_n}$=0:0.01:3;

```
zeta=[0.05;0.1;0.15;0.25;0.5;1.25;1.5]; % Damping factors
r=0:0.01:3; % Frequency ratios
for k = 1:length(zeta),
    G(k,:) = (r.^2)./ sqrt((1-r.^2).^2+(2*zeta(k)*r).^2);
end
plot(r,G,'linewidth',2.2)
xlabel('\omega / \omega_n')
ylabel('({\omega / \omega_n})^2   |G(I\omega)| ')
```

```
legend('\zeta_1=0.05','\zeta_2=0.1','\zeta_3=0.15','\zeta_4=0
.25','\zeta_5=0.5','\zeta_6=1.25','\zeta_7=1.5')
grid
```



مثال۴: برنامه ای بنویسید که پاسخ سیستم یک درجه آزادی با میرایی ویسکوز تحت نیروی $F(t) = F_0 e^{-\alpha t} u(t)$ را، به‌وسیله انتگرال کانولوشن محاسبه کند. پالس به صورت مستطیلی بوده و T=0.1sec است.



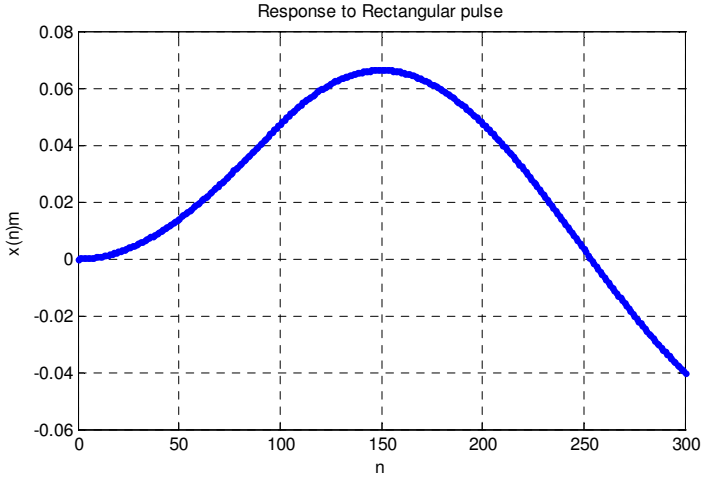
پریود نمونه برداری را برابر T=0.oo1sec و تعداد نمونه ها را n=300 در نظر بگیرید. پارامترهای سیستم برابرند با:

$$M = 25kg \quad ,c = 30\frac{Ns}{m}, k = 6000\frac{N}{m}, F_0 = 300N \ , \alpha = 1$$

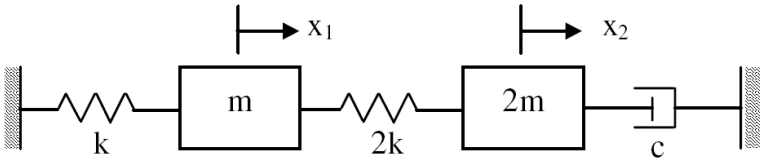
پاسخ ضربه سیستم جرم و فنر و دمپر برابر است با:

$$g(t)=\frac{1}{m\omega_d}e^{-\zeta\omega_n t}\sin\omega_d t\times u(t)$$

```
clear all
clc
close all
m = 25; % mass
c = 30;% damping
k=6000; % stiffness
F0=300; % Force amplitude
T =0.1;

wn = sqrt(k/m);   % Natural frequency
zeta = c/(2*sqrt(m*k));  % damping factor
Ts = 0.001;  % sampling period
N = 301;  % sampling times
wd = wn*sqrt(1-zeta^2);  % damped frequency

for n = 1:N,
    if n<=T / Ts+1
        F(n)=F0
    else
        F(n)=0
    end
end

n = [1:N];
g=Ts*exp(-(n-1)*zeta*wn*Ts).*sin((n-1)*wd*Ts)/(m*wd);

% discrete-time impulse response
c0 = conv(F,g);%convolution sum
c=c0(1:N); % plot to N samples
n = [0:N-1];
axes('position',[0.1 0.2 0.8 0.7])
plot(n,c,'.')
title('Response to Rectangular pulse')
xlabel('n')
ylabel('x(n)m');
grid
```



مثال۵: پاسخ ارتعاشات آزاد سیستم دو درجه آزادی نشان داده شده در شکل زیر را با شرایط اولیه زیر به‌دست آورید.



پارامترهای سیستم برابرند با:

$$x_1(0)=0 \quad x_2(0)=0.005m$$

$$\dot{x}_1(0)=0 \quad \dot{x}_2(0)=0$$

$$m=30kg, k=20000\frac{N}{m}, c=150\frac{N.s}{m}$$

معادله دیفرانسیل حاکم بر سیستم برابر است با:

$$\begin{bmatrix} m & 0 \\ 0 & 2m \end{bmatrix}\begin{bmatrix} \ddot{x}_1 \\ \ddot{x}_2 \end{bmatrix}+\begin{bmatrix} 0 & 0 \\ 0 & c \end{bmatrix}\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix}+\begin{bmatrix} 3k & -2k \\ -2k & 2k \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}=\begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$M\dot{y}+ky=0$$

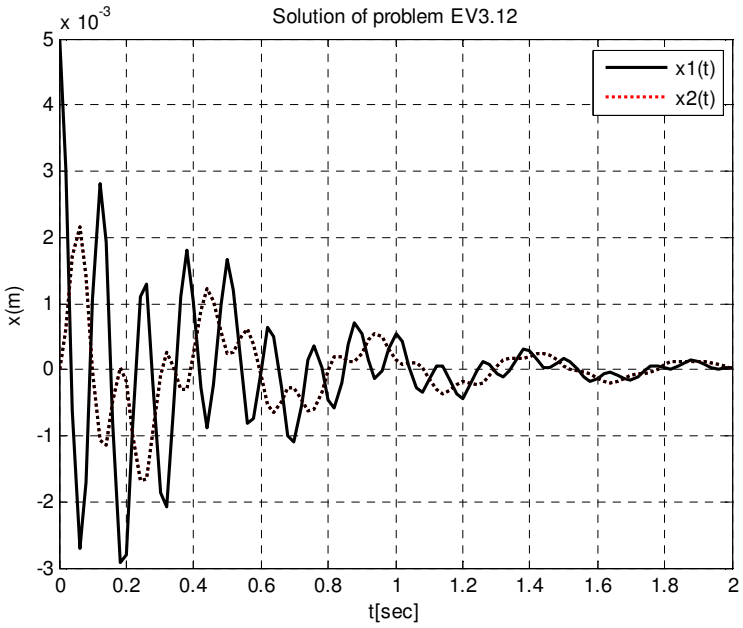
$$M=\begin{bmatrix} 0 & M \\ M & c \end{bmatrix} \qquad k=\begin{bmatrix} -M & 0 \\ 0 & K \end{bmatrix} \; ; \; Y=\begin{bmatrix} \dot{x} \\ x \end{bmatrix}$$

$$y=\varphi e^{-\gamma t}$$

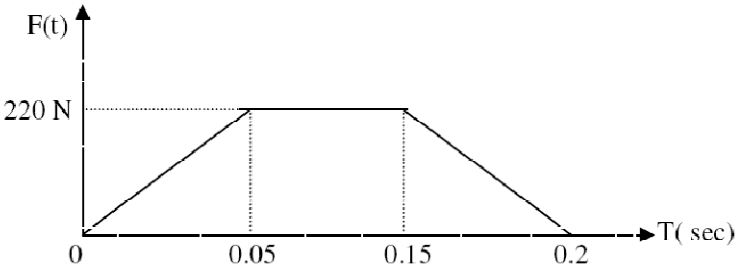
که γ مقادیر ویژه $M^{-1}K$ و φ بردارهای ویژه می باشند. حل عمومی برابر ترکیب خطی تمام جواب‌ها می باشد، یعنی:

$$y = \sum_{j=1}^{4} c_j \varphi_j e^{-\gamma_j t}$$

```
m = 30; % Mass
k=20000; % Stiffness
c = 150; % Damping

disp('4 x 4 Mass matrix');
mt = [0,0,m,0;0,0,0,2*m;m,0,0,0;0,2*m,0,c];
disp('4 x 4 stiffness matrix');
kt=[-m,0,0,0;0,-2*m,0,0;0,0,3*k,-2*k;0,0,-2*k,2*k];
Z=inv(mt)*kt;
[V,D]=eig(Z);
disp('Eigenvalues');
V
disp('Initial conditions');
x0=[0;0;0.005;0]
disp('Integration constants');
S=inv(V)*x0
tk=linspace(0,2,101);

for k=1:101
    t=tk(k);
    for i=3:4
        x(k,i-2)=0;
        for j=1:4
            x(k,i-2)=x(k,i-2)+(real(S(j))*real(V(i,j))-
imag(S(j))*imag(V(i,j)))*cos(imag(D(j,j))*t);
            x(k,i-2)=x(k,i-2)+(imag(S(j))*real(V(i,j))-
real(S(j))*imag(V(i,j)))*sin(imag(V(i,j))*t);
            x(k,i-2)=x(k,i-2)*exp(-real(D(j,j))*t);
        end
    end
end
plot(tk,x(:,1),'k-',tk,x(:,2),'r:')
title('Solution of problem EV3.12')
xlabel('t[sec]')
ylabel('x(m)')
legend('x1(t)','x2(t)')
```



مثال۶: پاسخ سیستم یک درجه آزادی با میرایی ویسکوز به پالس ذوزنقـه ای نمـایش داده در شـکل زیـر را بـا پارامترهای سیستم m=15kg و c=25Ns/m و k= 5000N/m رسم کنید.



معادله دیفرانسیل حاکم بر سیستم برابر است با:

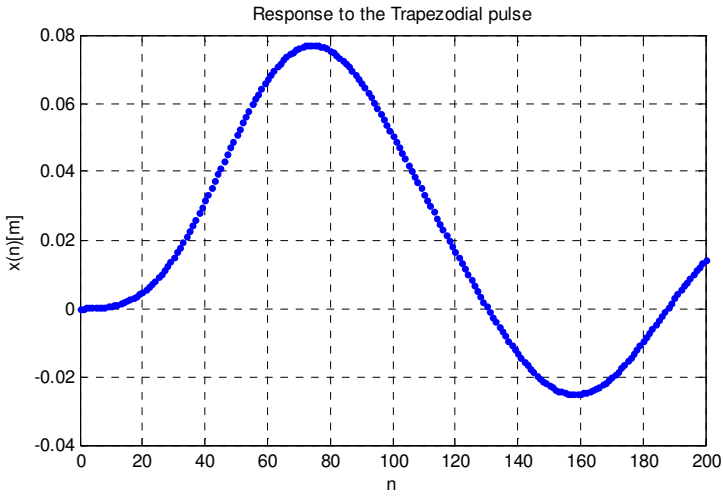
$$\ddot{x} + 2\zeta\omega_n\dot{x} + \omega_n{}^2 x = \frac{F(t)}{m}$$

$$F(t) = \begin{cases} \frac{2F_0}{T}t & 0 < t < \frac{T}{2} \\ F_0 & \frac{T}{2} < t < \frac{3T}{2} \\ 2F_0(2 - \frac{t}{T}) & (\frac{3T}{2}) < t < 2T \\ 0 & t > 2T \end{cases}, \quad T = 0.2\,\text{sec}$$

پاسخ زمان گسسته با جمع کانولوشن برابر است با:

$$X(n)=\sum_{k=0}^{n}F(k)g(n-k)$$

```
m=15; % mass
c = 25; % damping
k = 5000; % stiffness
F0 = 220;
T = 0.2;
wn = sqrt(k / m); % Natural frequency
zeta = c / (2*sqrt(m*k));
Ts=0.003; % Sampling period
N = 201; % sampling times
wd = wn*sqrt(1-zeta^2); % frequency

for n = 1:N,
    if n<= (T / 2) / Ts+1
        F(n)=2*F0*(n-1)*Ts / T
    else
        F(n)=F0
    end
    if n>(3*T / 2) / Ts+1
        F(n)=2*F0*(2-(n-1)*Ts / T)
    end
    if n>2*T / Ts+1
        F(n)=0
    end
end
n=[1:N];

g=Ts*exp(-(n-1)*zeta*wn*Ts).*sin((n-1)*wd*Ts) / (m*wd);
% discrete-time impulse response
c0 = conv(F,g); % Convolution sum
c = c0(1:N); % plot to N samples
n = [0:N-1];

axes('position',[0.1 0.2 0.8 0.7])
plot(n,c,'.');
title('Response to the Trapezodial pulse');
xlabel('n')
ylabel('x(n)[m]')
grid
```


Response to the Trapezodial pulse
x(n)[m]
n
0.08
0.06
0.04
0.02
0
-0.02
-0.04
0
20
40
60
80
100
120
140
160
180
200

❖ تمرینات تکمیلی

۱- در صورتی‌که $v=\frac{\pi R^3}{3\tan\theta}-\frac{\pi}{3}(R-y\tan\theta)^2(\frac{R}{\tan\theta}-y)$، برنامه‌ای بنویسید کـه مقـادیر y,θ را از کاربر بگیرد و مقدار حجم v را برای سه شعاع R1=1,2,3 محاسبه نمائید.

۲- مسافت افقی و عمودی سرعت حرکت پرتابه که با سرعت اولیه v و زاویه A نسبت به افق پرتاب شده است به صورت زیر می‌باشد:

$$h(t)=vt\sin A-\frac{1}{2}gt^2$$
$$x(t)=vt\cos A$$
$$v(t)=\sqrt{v^2-2vgt\sin A+g^2t^2}$$

زمانی‌که پرتابه به زمین برخورد می‌کند $h(t)=0$ ، بنابراین داریم

$$t_{hit}=\frac{2v}{g}\sin A$$

فرض کنید: $g=9.81\frac{m}{s^2}, v=40\frac{m}{s}, A=30\deg$

با استفاده از عملگرهای منطقی:

- مدت زمانی را که ارتفاع بیشتر از 15m است، به‌دست آورید.
- مدت زمانی را که ارتفاع بیشتر از 15m و سرعت کمتر از 30m/s است، به‌دست آورید.
- مسیر حرکت پرتابه را رسم نمائید.

۳- برنامه‌ای بنویسید که مقدار π را از فرمول زیر محاسبه کند. از طرفی مدت زمان انجـام برنامـه را بـه‌دسـت آورید.

$$Arc\tan(X)=X-\frac{X^3}{3}+\frac{X^5}{5}-\frac{X^7}{7}+\ldots$$

۴- برنامه‌ای بنویسید که مقدار تابع f(x) را در بازه [1 -1] با گام‌های متفاوت 0.01, 0.1, 0.2 محاسبه کند.

$$F(X)=X\sin[\frac{\pi(1+20X)}{2}]$$

مدت زمان انجام برنامه را در هر یک از حالت‌ها محاسبه کنید.

۵- برنامه‌ای بنویسید که سری فوریه برای تابع f(t) را که با رابطه زیر داده شده است، در بازه [1.1 -1.1] با گام‌های n=0.01 رسم کند. فرض کنید T=1 می‌باشد. این برنامه را برای مقادیر مختلف n به‌طـور مثـال 0.1 و 0.001 اجرا کنید.

$$f(t)=\frac{4}{\pi}\sum_{k=0}^{\infty}\frac{1}{2k+1}\sin[\frac{(2k+1)\pi t}{T}]$$

۶- برنامه‌ای بنویسید که تابع نمایی e^x را مستقیماً از سری تیلور زیر محاسبه کند.

$$e^x = 1 + X + \frac{X^2}{2!} + \frac{X^3}{3!} + \ldots$$

محاسبات سری، زمانی باید متوقف شود که آخرین ترم کوچکتر از 10^{-6} باشد.

۷- در صورتی‌که $\varphi(x) = 0.5 - r(at + bt^2 + ct^3)$، $a = 0.43, b = -0.12, c = 0.93$ و $r = \frac{e^{-0.5x^2}}{\sqrt{2\pi}}$ و $t = \frac{1}{(1+0.33x)}$، در آن صورت:

- تابعی بنویسید که $\varphi(x)$ را محاسبه کند.
- مقدار $\varphi(1)$ را به‌دست آورید.
- در برنامه‌ای دیگر با استفاده از تابع تعریف شده، نمودار $\varphi(x)$ را در بازه $0 \le x \le 4$ با گام 0.1 رسم کنید.

۸- اعداد فیبوناچی با استفاده از دنبالهٔ زیر حاصل می‌شوند:

$1,1,2,3,5,8,13,\ldots$

با استفاده از توابع بازگشتی F(n) مقادیر اعداد فیبوناچی F_0 تا F_{20} را با استفاده از رابطه $F_n = F_{n-1} + F_{n-2}$ به‌دست آورید.

۹- سه جمله اول چندجمله‌ای لژاندر برابر $P_0(x) = 1, P_1(x) = x, P_2(x) = \frac{3x^2 - 1}{2}$ است. فرمول بازگشتی عمومی چندجمله‌ای لژاندر بصورت زیر می‌باشد:

$$(n+1)P_{n+1}(x) - (2n+1)xP_n(x) + nP_{n-1}(x) = 0$$

برنامه‌ای بنویسید که تابع بازگشتی $P(n,x)$ را محاسبه کند.

۱۰- معادله دیفرانسیل غیرخطی زیر را به صورت عددی محاسبه نمائید.

$$\frac{dx_1(t)}{dt} = -15x_1 + 10|x_2| + 10x_1x_2x_3 \quad , x_1(t_0) = x_{10}$$

$$\frac{dx_2(t)}{dt} = -5x_1x_2 - \sin x_1 + x_2 - x_3 \quad , x_2(t_0) = x_{20}$$

$$\frac{dx_3(t)}{dt} = -5x_1x_2 + 10x_2\cos x_1 - 15x_3 \quad , x_3(t_0) = x_{30}$$

اگر $x_0 = \begin{bmatrix} x_{10} \\ x_{20} \\ x_{30} \end{bmatrix} = \begin{bmatrix} 15 \\ -15 \\ 10 \end{bmatrix}$، $x_1(t), x_2(t), x_3(t)$ را در بازه زمانی $[0,3]$ رسم کنید، سپس تابع $x(t) = (x_1(t), x_2(t), x_3(t))$ را با plot3 رسم نمائید.

۱۱- در مدار الکتریکی RLC سری پاسخ گذرای سیستم نسبت به ورودی پله واحد را با شرایط اولیه زیر ترسیم کنید.

$$\frac{d^2i}{dt^2}+\frac{R}{L}\frac{di}{dt}+\frac{1}{LC}i=\frac{1}{L}\frac{du_a}{dt}$$

$$R=0.4\ Ohm, C=2F, L=0.5H, a=2,\ b=-2$$

۱۲- در یک مدار RLC سری با معادلات زیر حل تحلیلی را انجام داده، و $u_e(t)$ را بیابید:

$$c\frac{du_c}{dt}=i, L\frac{di}{dt}=-u_c-R_i+u_a(t)$$

که در آن $u_a(t)$ ولتاژ منبع تغذیه می‌باشد.

۱۳- اگر $f(t)=3u(t)-3u(t-1)+3u(t-2)$ باشد، نمودار تابع f را در بازه $-5\le t\le 5$ رسم کنید.

۱۴- انتگرال‌های زیر را محاسبه نمائید.

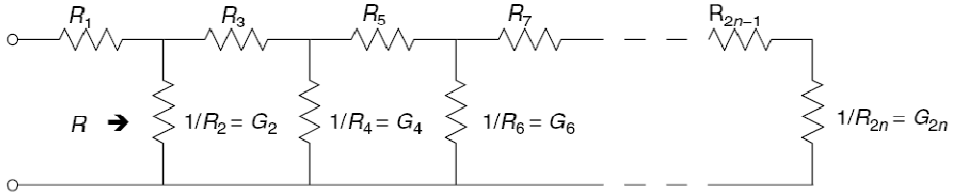
$$\int_{-\infty}^{\infty}(2t+3)\delta(t-1)dt$$

$$\int_{-\infty}^{\infty}\sin(\omega t)\delta((\frac{3t}{2})-\frac{2}{3})dt$$

۱۵- تابع f1 را در بازه $0\le n\le 128$ رسم کنید.

$$f_1(n)=\left\{\frac{\sin[0.3\pi(n-64)]}{0.3\pi(n-64)}\right\}^2$$

۱۶- برای مدار نشان داده شده در شکل زیر:



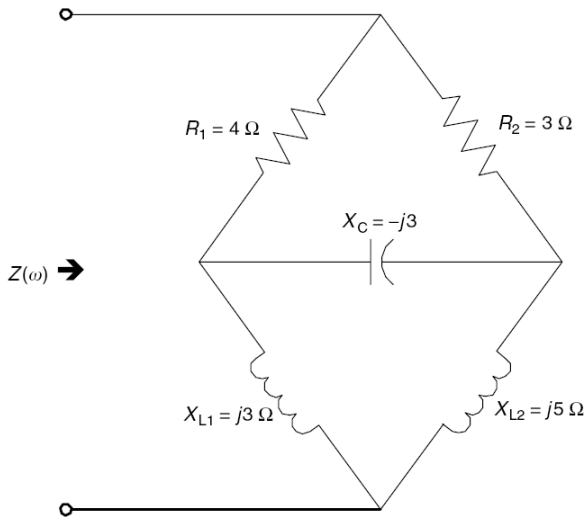
الف- مقدار R را با نوشتن برنامه‌ای در MATLAB،که با استفاده از تکنیک‌های تحلیلی مدار به‌دست می‌آید، محاسبه کنید.

ب- با نوشتن برنامه‌ای در MATLAB، مقدار R را با استفاده از رابطه زیر محاسبه کنید.

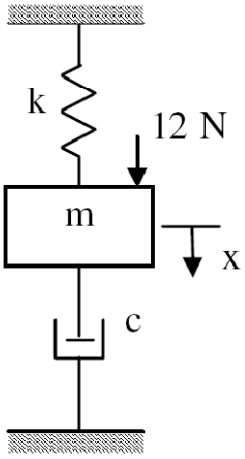
$$R=R_1+\sum_{I=1}^{n}\frac{1}{G_{2i}}+\sum_{i=1}^{n}\frac{1}{R_{2i-1}}$$

$$for\ \ n=3, n=4$$

۱۷- برنامه‌ای بنویسید که نمودارهای $|z(w)|, z(w)$ را نسبت به w برای مدار زیر رسم کند.



۱۸- پاسخ سیستم یک درجه آزادی ارتعاشی زیر را درصورتی‌که نیروی پله f=12N به جرم m وارد شود، رسم کنید. جابه‌جایی x از نقطه تعادل محاسبه می شود.

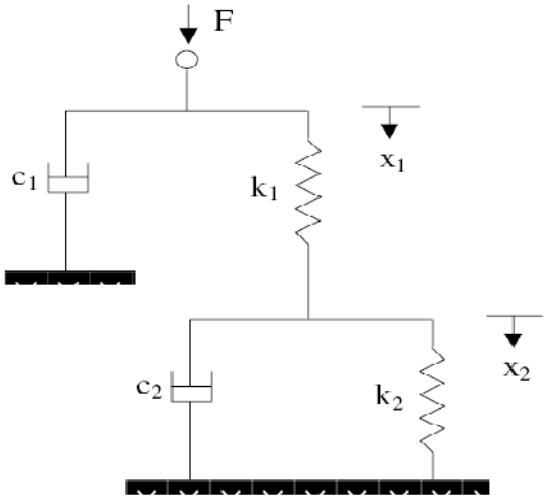


فرض کنید:

m=2kg, c=10Ns/m, k=80N/m

۱۹- برای سیستم ارتعاشی مکانیکی نشان داده شده در شکل زیر، پارامترهای سیستم برابر است با :

$k_1 = 10N/m \ , k_2 = 30N/m \ , c_1 = 3N.s/m \ , c_2 = 25N.s/m$



پاسخ x2(t) را در صورتی‌که که نیروی پله f برابر 4N باشد، رسم کنید.

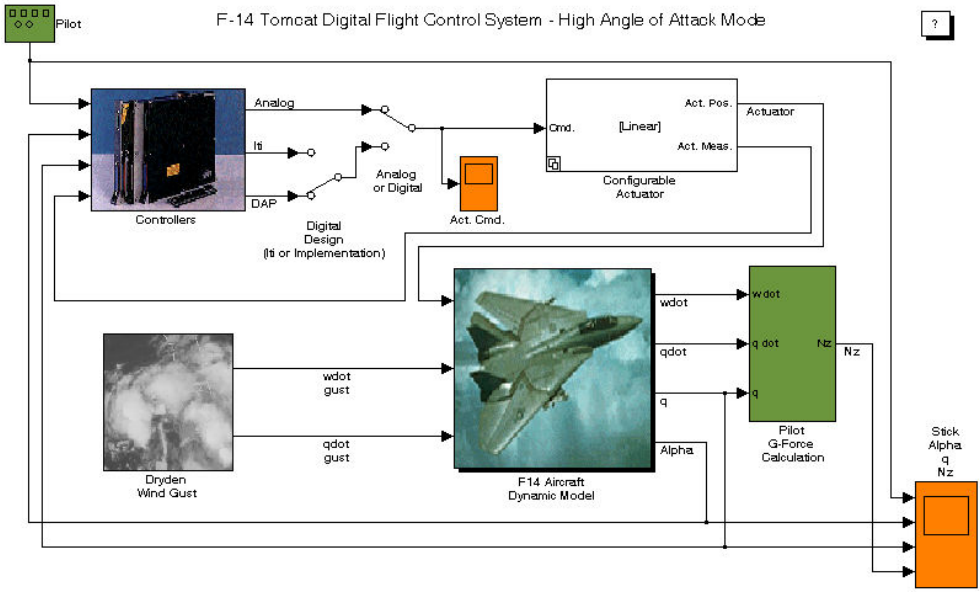
فصل ششم

سیمولینک

۶–۱ ایجاد مدل در سیمولینک

در این بخش با محیط سیمولینک و ایجاد مدل در آن آشنا می‌شوید. سیمولینک از آیکون‌ها برای ساخت بلوک نمودار یک پروسه، حل، وشبیه سازی آن استفاده می‌کند. در سیمولینک به‌جای کدنویسی، آیکون‌های مورد نیاز را به هم متصل می‌کنیم تا بلوک نمودار مدل شکل بگیرد. سیمولینک به کاربر این امکان را می‌دهد که به‌راحتی معادلات دیفرانسیل خطی و غیرخطی معمولی را حل کند. این بسته نرم افزاری مکرراً در طراحی پروسه‌های شیمیایی در صنعت و طراحی سیستم‌های کنترلی استفاده می‌شود. در این بسته می‌توان سیستم‌های خطی و غیرخطی را به صورت زمان پیوسته یا زمان گسسته و یا ترکیبی از این دو شبیه سازی کرد. برای مشاهده چند نمونه از demoهای سیمولینک، دستورات زیر را در پنجره دستورات تایپ کنید:

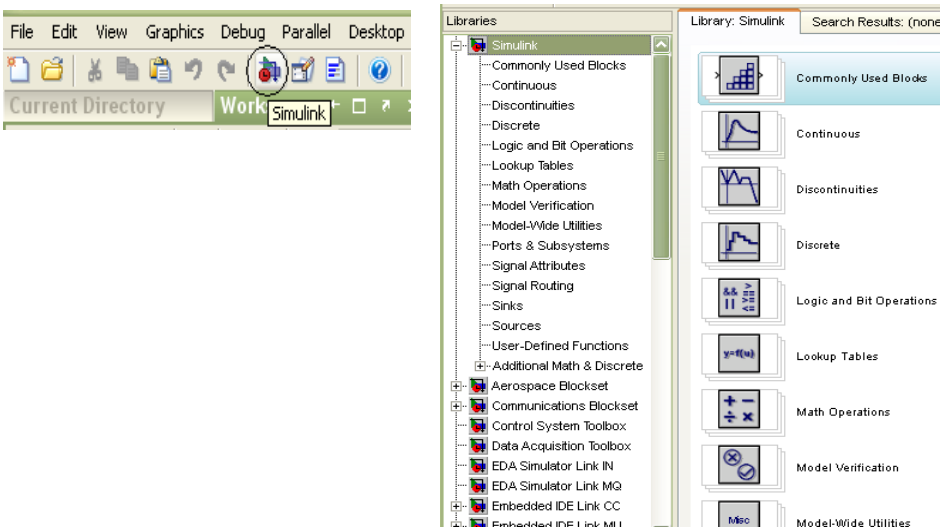
```
>> open('f14_digital')
>> open('aero_dap3dof')
>> open('sldemo_tank.mdl')
```



شکل (۶–۱) بلوک نمودار هواپیمای F-14

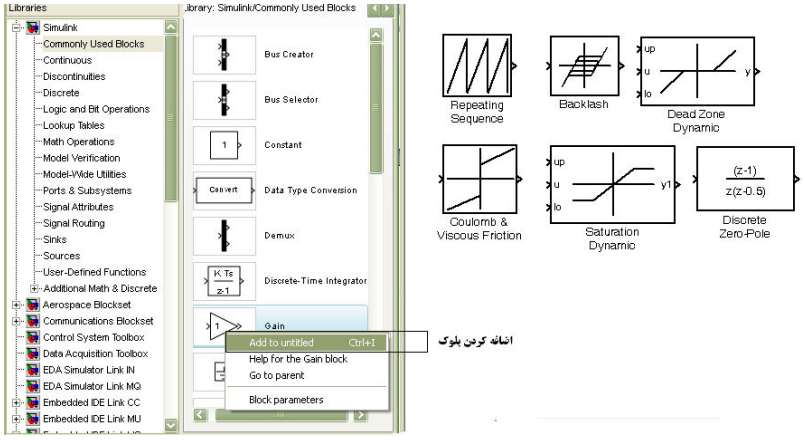
سپس روی هر یک از بلوک‌ها دوبارکلیک کرده تا با طریقه شکل‌گیری این مدل آشنا شوید. در نوار ابزار، روی start simulation کلیک نمائید تا شبیه‌سازی آغاز شود.

برای مشاهده بلوک‌های سیمولینک روی آیکون Simulink در نوار ابزار کلیک کنید یا در پنجره دستورات، Simulink را تایپ کنید تا پنجره کتابخانه سیمولینک باز شود. همانطور که در شکل(۶-۲) مشاهده می‌کنید کتابخانه سیمولینک دارای بلوک‌های مختلفی می‌باشد که در ادامهٔ این بخش توضیح داده خواهند شد.



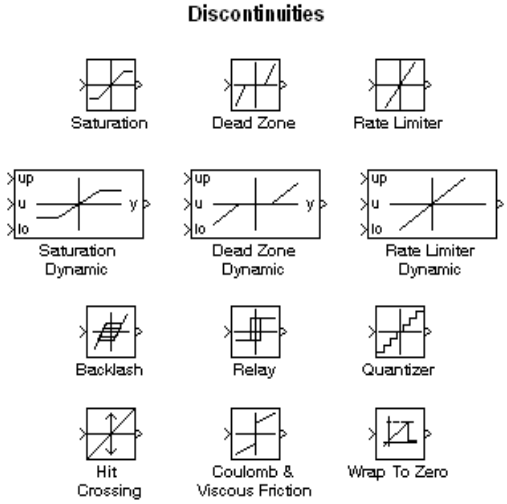
شکل (۶-۲) نمایش بلوک‌های سیمولینک

برای باز کردن یک پنجره جدید مسیر File>>New>>Model را دنبال کنید. برای اضافه کردن بلوک به پنجره مدل‌سازی می‌توان از روش Drag and Drop استفاده کرد. البته می‌توان روی بلوک مورد نظر کلیک راست کرده و گزینه Add to untitled را انتخاب کرد تا بلوک در پنجره مدل ایجاد شود.



شکل (۶-۳) نمایش بلوک‌ها، نحوه انتقال بلوک‌ها به صفحه مدل‌سازی

در شبیه سازی سیستم‌ها می‌توان بسیاری از عوامل غیرخطی و ناپیوستگی‌ها مثل اشباع (Saturation)، لقی در چرخ دنده‌ها (Back Lash)، هیسترزیس، Dead Zone، و... را در نظر گرفت که در شکل (۶-۴) بعضی از این بلوک‌ها را مشاهده می‌کنید.



شکل (۶-۴) بلوک‌های غیرخطی سیمولینک

۶-۲ آشنایی با بلوک‌های اشباع و انتگرال‌گیر و منبع موج سینوسی

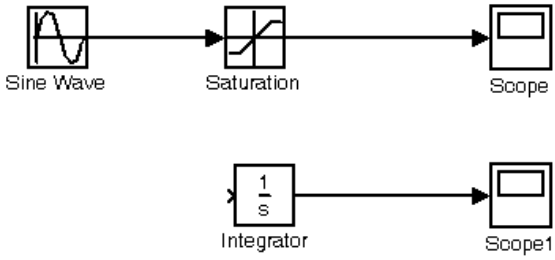
مثال: موج سینوسی با دامنه ۱ را به صفحه مدل سازی انتقال داده، سپس برای انتگرال‌گیری و محدود کردن خروجی در بازه [-0.5,+0.5] به ترتیب از بلوکهای انتگرال‌گیر و اشباع استفاده کنید.

ابتدا از کتابخانه سیمولینک مسیرهای زیر را دنبال کنید و بلوک‌های مورد نظر را وارد پنجره مدل کنید:

جدول(۶-۱) مسیر بلوک‌های مورد استفاده در این مثال

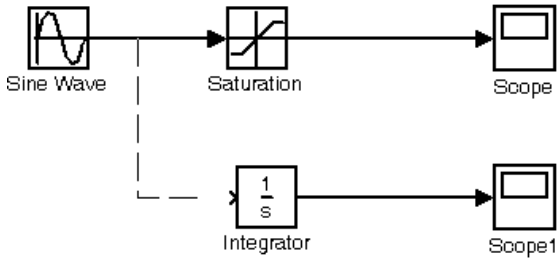
تعداد	مسیر بلوک
1	Simulink >>source>>Sine wave
2	Simulink >>Sinks>>Scope
1	Simulink >>Continuous>>Integrator
1	Simulink >>Discontinuities>>Saturation

بلوک‌ها را مانند شکل زیر با کلیک چپ و جابه‌جا کردن هر یک، مرتب کرده و آنها را به‌هم اتصال دهید. برای اتصال بلوک Sine Wave به بلوک Saturation ، اشاره گر ماوس را به سمت موج خروجی موج سینوسی برده تا به شکل (+) شود. سپس با نگه داشتن دکمه سمت چپ ماوس به سمت ورودی میانی بلوک Saturation رفته و سپس دکمه ماوس را رها کنید. بقیه بلوک‌ها را به همین طریق به‌هم وصل کنید.



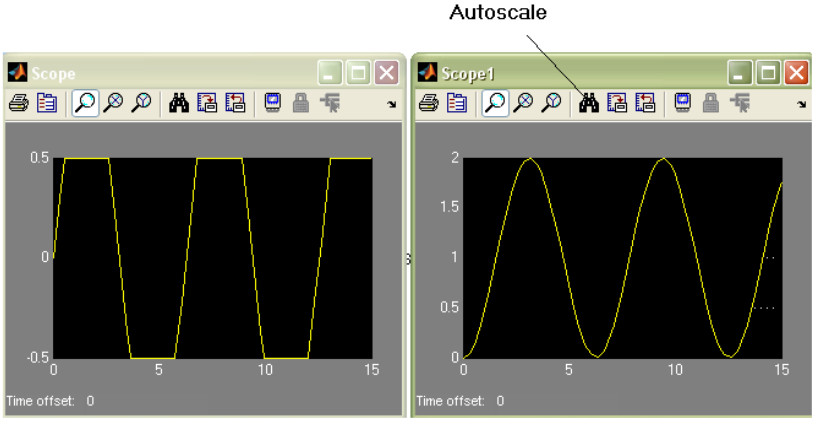
شکل (۶-۵) اتصال بلوک‌ها به‌یکدیگر

برای اتصال موج سینوسی به Integrator، اشاره گر ماوس را روی خط اتصال بین موج سینوسی و saturation برده و کلیک راست کنید. سپس با نگه داشتن دکمه سمت راست ماوس آن را به سمت ورودی بلوک Integrator برده و دکمه ماوس را رها کنید.



شکل (۶-۶) اتصال بلوک‌ها به‌یکدیگر

برای تنظیم مدت زمان حل مسأله، در قسمت بالای مدل زمان توقف حل سیمولینک را ۱۵ وارد کنید. سپس روی دکمه Start Simulation کلیک کرده تا شبیه سازی آغاز شود. بعد از اتمام شبیه سازی روی هر یک از بلوک‌های scope و scope1 دوبار کلیک کرده، سپس در پنجره ظاهر شده در نوار ابزار روی Autoscale کلیک نمایید تا نتایج را مشاهده کنید.



شکل (۶-۷) خروجی شبیه‌سازی

۶-۳ آشنایی با بلوک‌های مثلثاتی و توابع ریاضی و انتقال داده‌های خروجی به فضای کاری MATLAB

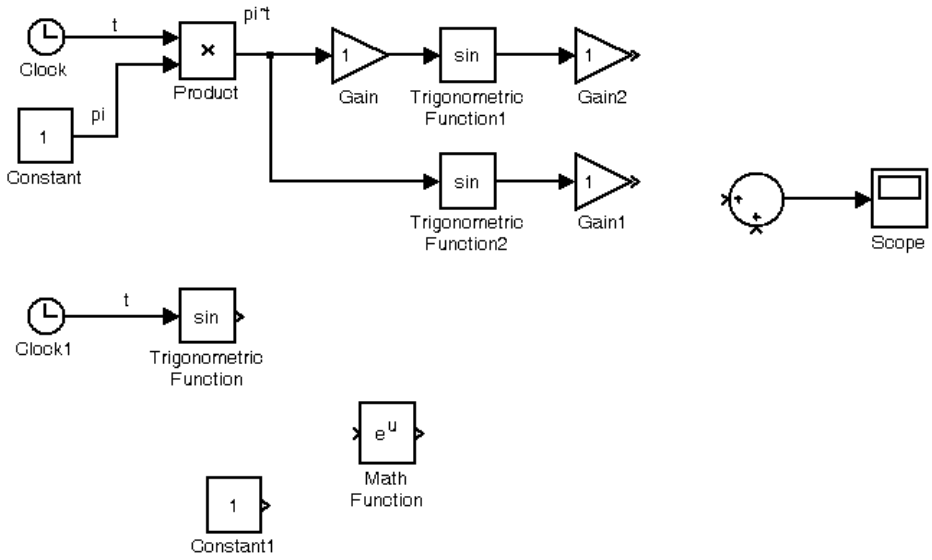
مثال: موج $y(t)=2\sin(\pi t)+6\tan(2\pi t)-\cos(t)-0.5t^{-2}$ را در محیط سیمولینک ایجاد کرده، سپس مقادیر ورودی و خروجی را به Workspace انتقال دهید.

ابتدا از کتابخانه سیمولینک مسیرهای زیر را دنبال کرده و بلوک‌های مورد نظر را وارد پنجره مدل کنید:

جدول(۶-۲) مسیر بلوک‌های مورد استفاده در این مثال

تعداد	مسیر بلوک
2	Simulink >>Sources>>Clock
2	Simulink >>Sources>>constant
1	Simulink >>Sinks>>Scope
4	Simulink >> Math operation>> Gain
1	Simulink >> Math operations>> sum
3	Simulink >> Math operations>> Trigonometric Function
1	Simulink >> Math operations>> product
1	Simulink >> Math operations>> Math Function
1	Simulink >>Sinks>>to workspace

بلوک‌ها را مانند شکل (۶-۸) با کلیک چپ کردن و جابه‌جا کردن هر یک، مرتب کرده و آنها را به‌هم اتصال دهید.



شکل (۶-۸) اتصال بلوک‌ها به‌یکدیگر

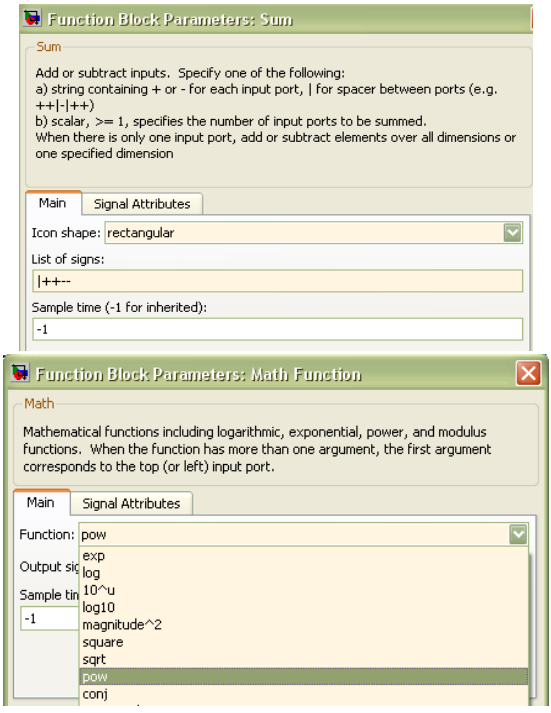
تنظیمات بلوک‌ها

- روی بلوک sum دوبار کلیک کرده و در پنجره باز شده در مقابل Icon shape گزینه rectangular را انتخاب کنید. در قسمت List of Signs علامت‌ها را با توجه به تعداد + و - های موجود در معادله موج تنظیم کرده و ok کنید. سپس بلوک Sum را از اطراف بکشید تا به اندازه موجود درشکل زیر برسید.
- خروجی clock همان زمان اجرای سیمولینک است.

خروجی Constant یک مقدار ثابت نسبت به زمان است. روی بلوکهای constant دوبار کلیک کرده تا پنجره تنظیمات ظاهر شود و در قسمت Constant value مقادیر زیر را برای هر یک از بلوک‌های constant وارد کنید:

Constant =1 , Constant1= -2

- از بلوک Math Function برای ایجاد توابع ریاضی استفاده می‌شود. حال روی بلوک Math Function دوبار کلیک کرده و در قسمت function گزینه pow را انتخاب کنید. ورودی اول مقدار u و ورودی دوم مقدار v است.

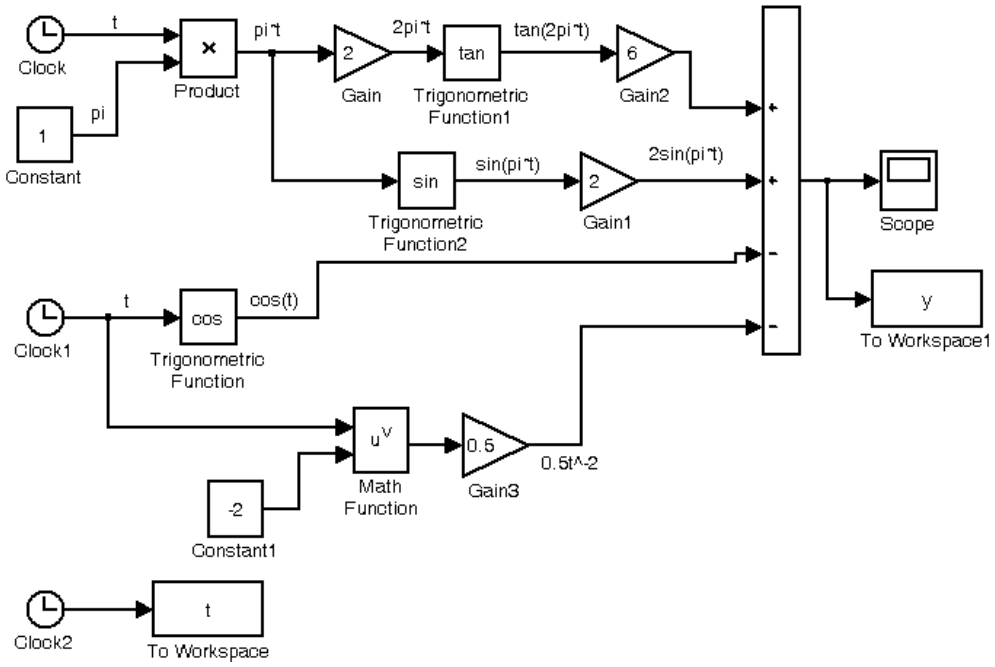


شکل (۶-۹) تنظیمات بلوک‌ها

- بلوک Gain ورودی را در یک عدد ثابت ضرب می‌کند. روی بلوکهای Gain به ترتیب دوبارکلیک کرده تا پنجره تنظیمات ظاهر شود و در قسمت Gain مقادیر زیر را برای هریک از بلوک‌ها وارد کنید:

Gain=2 , Gain1=2 , Gain2= 6, Gain3=0.5

- روی بلوک‌های Trigonometric Function دوبار کلیک کنید تا پنجره تنظیمات ظاهر شود. سپس در قسمت Function نوع تابع را با توجه به شکل (۶-۱۰)تنظیم کنید.

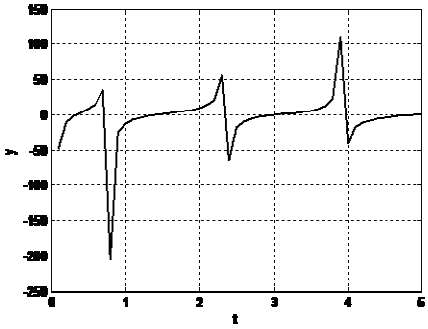


شکل (۶-۱۰) اتصال بلوک‌ها به‌یکدیگر

- روی بلوک to workspace دوبار کلیک کنید تا پنجره تنظیمات ظاهر شود. سپس در قسمت variable name نام t را وارد کنید. از طرفی برای انتخاب نوع فرمت ذخیره شدن، در قسمت save format گزینه Array را انتخاب کنید. همین کار را برای بلوک to workspace1 انجام دهید و نام متغیر را y قرار دهید.

توجه: برای نوشتن یک متن در داخل مدل، کافی است در محل مورد نظر دوبار کلیک نموده و سپس در کادر باز شده متن دلخواه را بنویسید.

برای اجرای مدل، روی Start Simulation کلیک کرده و سپس روی scope دوبار کلیک کنید. روش دیگر برای مشاهده خروجی، تایپ دستور plot(t,y) در پنجره فرامین می‌باشد.



شکل (۶-۱۱) منحنی $y(t)=2\sin(\pi t)+6\tan(2\pi t)-\cos(t)-0.5t^{-2}$

۶-۴ آشنایی با زیرسیستم(Subsystem) و ایجاد ماسک

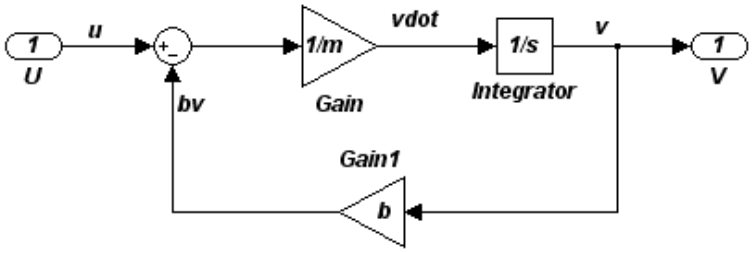
مثال: سیگنال میرای $y(t)=5e^{-o.7t}\sin(2.4t-\frac{\pi}{6})$ را تولید کرده، سپس از این سیگنال به‌عنوان Source برای شبیه سازی سیستم $u-bv=m\frac{dv}{dt}$ استفاده کنید.

در ابتدا مدل را در محیط سیمولینک شبیه سازی می‌نمائیم. برای مدل سازی سیستم در محیط سیمولینک بلوک‌های مورد نیاز را مطابق جدول زیر به پنجره مدل سازی انتقال دهید.

جدول(۶-۳) مسیر بلوک‌های مورد استفاده در این مثال

تعداد	مسیر بلوک
1	Simulink>>Source>>In1
1	Simulink>>Sink>>Out1
1	Simulink>>Sink>>Scope
1	Simulink>>Continuous>>Integrator
2	Simulink>>Math operation>>Gain
2	Simulink>> Math operation >> Sum

بعد از انتقال بلوک‌ها مطابق شکل (۶-۱۲) بلوک‌ها را به هم متصل کنید تا مدل سیستم ساخته شود.

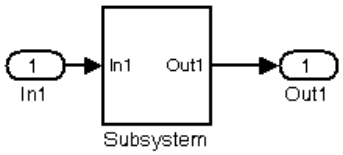


شکل (۶-۱۲) اتصال بلوک‌ها به‌یکدیگر

تنظیمات بلوک‌ها

- روی Gain دوبار کلیک کرده و مقدار 1/m را وارد کنید.
- روی Gain1 دوبار کلیک کرده و مقدار b را وارد کنید. برای چرخاندن بلوک‌ها از Ctrl+R استفاده کنید.

سپس تمامی بلوک‌ها را انتخاب کرده و روی صفحه کلیک راست کرده و گزینه Create Subsystem را انتخاب کنید. تمام مدل در یک زیر سیستم[1] قرار داده می‌شود. از این به بعد این زیر سیستم به‌عنوان مدل شناخته می شود.



شکل (۶-۱۳) بلوک subsystem

زمان شبیه سازی را برابر ۱۰ ثانیه قرار داده و مقادیر زیر را در پنجره دستورات واردکنید. سپس دکمه start simulation را بزنید تا شبیه سازی شروع شود.

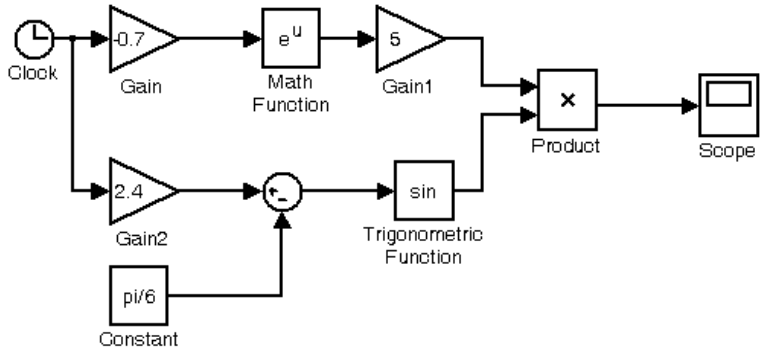
```
>>m=1;
>>b=20;
```

در مرحله بعد باید سیگنال میرای $y(t)=5e^{-o.7t}\sin(2.4t-\frac{\pi}{6})$ را به عنوان ورودی به سیستم اعمال کنیم تا خروجی سیستم مشاهد شود. برای انجام این کار از کتابخانه سیمولینک مسیرهای زیر را دنبال کرده و بلوک های مورد نظر را وارد پنجره مدل کنید:

جدول(۶-۴) مسیر بلوک‌های مورد استفاده در این مثال

تعداد	مسیر بلوک
2	Simulink >>Sources>>Clock
1	Simulink >>Sources>>constant
1	Simulink >>Sinks>>Scope
3	Simulink >> Math operation>> Gain
1	Simulink >> Math operations>> sum
1	Simulink >> Math operations>> Trigonometric Function
1	Simulink >> Math operations>> product
1	Simulink >> Math operations>> Math Function
1	Simulink >>Sinks>>to workspace

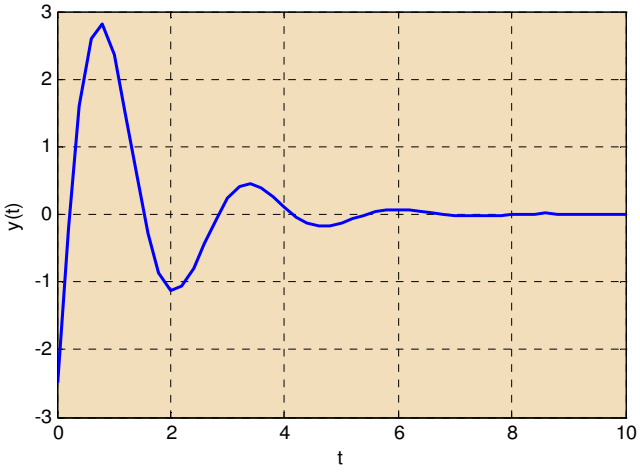
[1] Subsystem



شکل (۶-۱۴) اتصال بلوک‌ها به‌یکدیگر

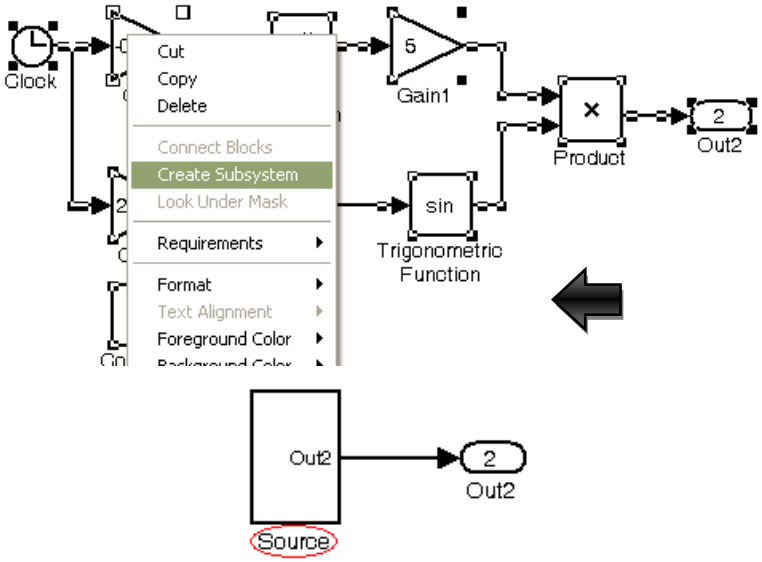
توجه: مقادیر بلوک‌های Gain در شکل مشخص شده اند.

سپس روی start simulation کلیک کرده و با دوبار کلیک کردن روی scope سیگنال میراشده را مشاهده کنید.



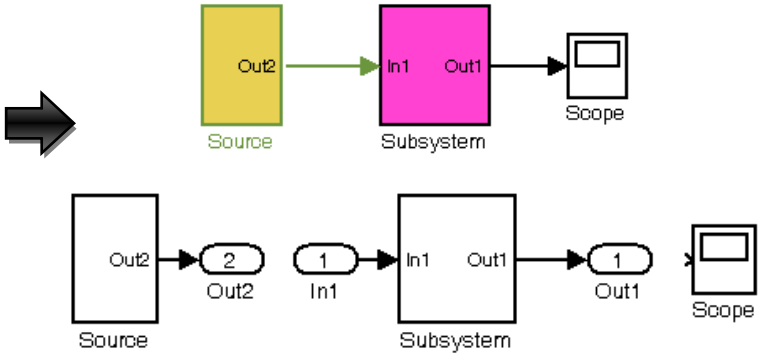
شکل (۶-۱۵) رسم منحنی میرا شده

خط اتصالی بین scope و product را پاک کرده و خروجی product را به out2 وصل کنید. همه بلوک‌های مربوط به شکل‌گیری سیگنال میرا را انتخاب کرده و روی صفحه بر روی مدل، کلیک راست نمایید. سپس create subsystem را انتخاب کنید تا زیرسیستم ایجاد شود. سپس نام subsystem را به Source تغییر دهید.



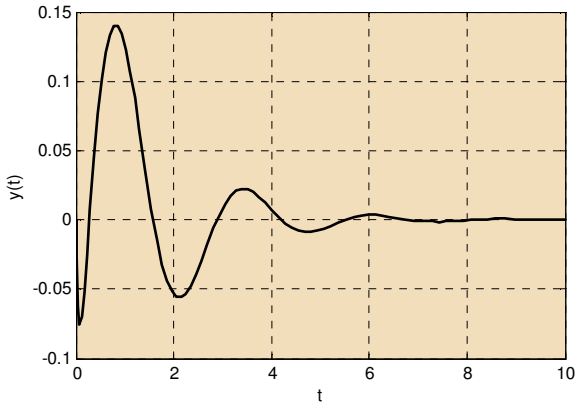
شکل (۶-۱۶) نحوه اتصال بلوک‌ها به‌یکدیگر

حال بلوک‌های in1 ,out2,out1 را پاک کرده و خروجی زیرسیستم source را مطابق شکل، به بلوک subsystem وصل کنید.



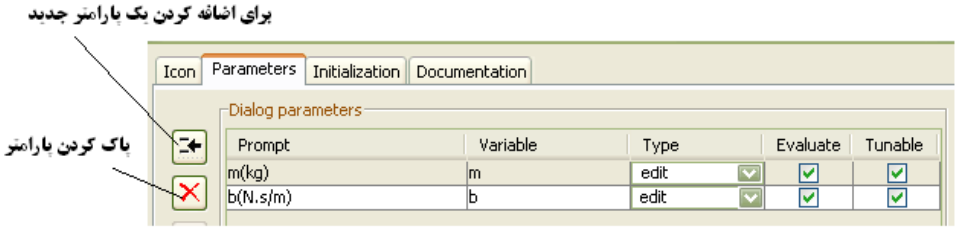
شکل (۶-۱۷) نحوه اتصال بلوک subsystem به بلوک‌های source و scope

سپس روی start simulation کلیک کرده و با دوبار کلیک کردن روی scope خروجی سیستم را به سیگنال میراشده مشاهده کنید.



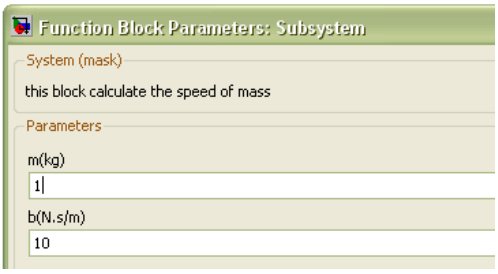
شکل (۶-۱۸) خروجی سیستم تحت تاثیر سیگنال ورودی میرا شده

ایجاد ماسک برای subsystem: روی subsystem کلیک راست کرده و گزینه Mask Subsystem را انتخاب کنید. با انتخاب این گزینه پنجره Mask Editor ظاهر می‌شود. در این پنجره به قسمت Parameters مراجعه نموده و تنظیمات زیر را اعمال کنید:



شکل (۶-۱۹) تنظیم پارامترها

حال اگر روی subsystem دوبار کلیک کنید پنجره زیر ظاهر می‌شود که می‌توانید مقادیر b و m را وارد کنید.



شکل (۶-۲۰) پارامترهای بلوک تابع

۶-۵ حل چند معادله- چند مجهول‌های خطی و غیرخطی

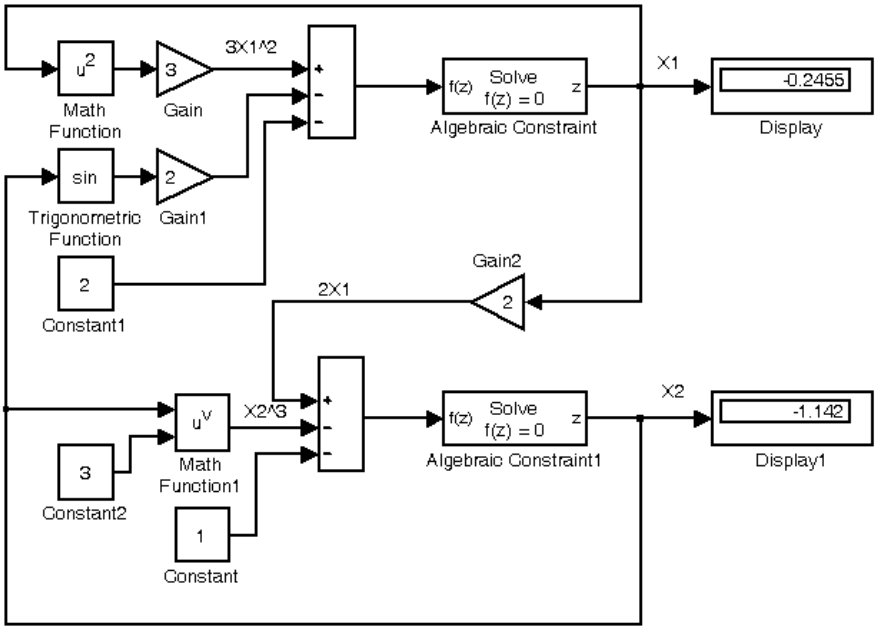
مثال: دومعادله- دومجهول غیرخطی $\begin{cases} 3x_1^{\,2} - 2\sin(x_2) = 2 \\ 2x_1 - x_2^{\,3} = 1 \end{cases}$ را به کمک سیمولینک حل کنید.

ابتدا از کتابخانه سیمولینک مسیرهای زیر را دنبال کنید و بلوک‌های مورد نظر را وارد پنجره مدل کنید:

جدول(۶-۵) مسیر بلوک‌های مورد استفاده در این مثال

تعداد	مسیر بلوک
3	Simulink >>Sources>>constant
2	Simulink >>Sinks>>Display
3	Simulink >> Math operation>> Gain
2	Simulink >> Math operations>> sum
1	Simulink >> Math operations>> Trigonometric Function
2	Simulink >> Math operations>> Math Function
2	Simulink >> Math operations>> Algebraic Constraint

بلوک‌ها را مانند شکل (۶-۲۱)با کلیک چپ کردن و جابه‌جا کردن هر یک، مرتب کنید و آنها را بهم اتصال دهید.



شکل (۶-۲۱) نحوه اتصال بلوک‌ها به‌یکدیگر

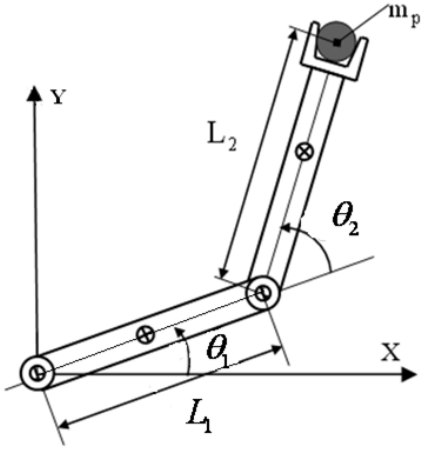
با فشردن دکمه start Simulation می‌توان مقادیر x1 وx2 را در بلوک‌های Display1 و Display2 مشاهده کرد.

در این مثال برای حل معادله از روش حلقه بسته استفاده شده است. در ابتدا به وسیله خروجی Algebraic Constraint که همان جواب مسأله می‌باشد جملات موجود در معادله f(z)=0 ساخته می‌شوند. سپس در بلوک sum، این جملات با هم جمع و تفریق می‌شوند تا معادله f(z) ساخته شود. سپس f(z) وارد بلوک Algebraic Constraint می‌شود تا پاسخ z به‌دست آید.

۶-۶ آشنایی با بلوک Embedded MATLAB Function

مثال: مدل‌سازی دینامیکی و شبیه‌سازی منیپولاتورهای دو لینکی با مفاصل صلب

با توجه به یک ربات دو درجه آزادی صفحه‌ای که در شکل (۶-۲۲)نشان داده شده است، معادلات سینماتیکی که موقعیت پنجه را نسبت به تغییر مکانهای مفصل بیان می‌کند، به صورت زیر قابل بیان است:



شکل (۶-۲۲) روبات دولینکی با مفاصل صلب

$$X_{tip} = L_1 \cos(\theta_1) + L_2 \cos(\theta_1 + \theta_2)$$
$$Y_{tip} = L_1 \sin(\theta_1) + L_2 \sin(\theta_1 + \theta_2)$$

معادلات دینامیکی منیپولاتور از معادلات لاگرانژ- اویلر به صورت زیر قابل بیان است:

$$\tau(t) = M(q(t))\ddot{\theta}(t) + C(\theta(t), \dot{\theta}(t)) + G(\theta(t))$$

که در آن $\tau \in R^n$ گشتاور مفاصل، $M(\theta) \in R^{n \times n}$ ماتریس اینرسی، $C(\theta, \dot{\theta}) \in R^n$ نیروهای کریولیس و جانب مرکز و $G(\theta) \in R^n$ اثرات جاذبه را توصیف می‌کنند.

$$M(q(t)) = \begin{bmatrix} M11 & M12 \\ M21 & M22 \end{bmatrix}$$

$$C(\theta, \dot{\theta}) = \begin{bmatrix} C_1 \\ C_2 \end{bmatrix} = \begin{bmatrix} -l_1(m_2 l_{c2}^2 + m_p l_2) \times Sin(\theta_2) \times (2\dot{\theta}_1\dot{\theta}_2 + \dot{\theta}_2^2) \\ l_1(m_2 l_{c2}^2 + m_p l_2) \times Sin(\theta_2) \times \dot{\theta}_2^2 \end{bmatrix}$$

$$G(\theta)=\begin{bmatrix}G_1\\G_2\end{bmatrix}=\begin{bmatrix}g(a_1\times Cos(\theta_1)+a_2\times Cos(\theta_1+\theta_2)\\g\times a_2\times Cos(\theta_1+\theta_2)\end{bmatrix}$$

که θ_1 , θ_2 زوایای لینک‌ها می‌باشند. پارامترهای استفاده شده در معادلات بالا به شرح زیر می‌باشند:

$$a_1=m_1l_{c1}+(m_2+m_p)l_1$$

$$a_2=m_2l_{c2}+m_pl_2$$

$$M_{11}=(I_1+m_1l_{c1}^2+(m_2+m_p)l_1^2)+(I_2+m_2l_{c2}^2+m_pl_2^2)+2l_1(m_2l_{c2}^2+m_pl_2)\times Cos(\theta_2)$$

$$M_{12}=M_{21}=(I_2+m_2l_{c2}^2+m_pl_2^2)+l_1(m_2l_{c2}^2+m_pl_2)\times Cos(\theta_2)$$

$$M_{22}=I_2+m_2l_{c2}^2+m_pl_2^2$$

با در نظر گرفتن متغیرهای حالت X_1,X_2 و کنترل U به صورت زیر:

$$X_1=\begin{bmatrix}\theta_1(t)\\\theta_2(t)\end{bmatrix}=\begin{bmatrix}x_1(t)\\x_2(t)\end{bmatrix},\quad X_2=\begin{bmatrix}\dot\theta_1(t)\\\dot\theta_2(t)\end{bmatrix}=\begin{bmatrix}x_3(t)\\x_4(t)\end{bmatrix},\quad U=\begin{bmatrix}\tau_1(t)\\\tau_2(t)\end{bmatrix}$$

معادلات دینامیکی سیستم به فرم فضای حالت زیر استخراج می‌گردند:

$$\begin{cases}\dot X_1=X_2\\\dot X_2=f(X_1,X_2)+g(X_1)U\end{cases}$$

که در آن مقادیر بردار f و ماتریس g عبارتند از:

$$g=\frac{1}{M_{11}M_{22}-M_{12}^2}\begin{bmatrix}M_{22}&-M_{12}\\-M_{12}&M_{11}\end{bmatrix}$$

$$f=\frac{-1}{M_{11}M_{22}-M_{12}^2}\begin{bmatrix}M_{22}(C_1+G_1)-M_{12}(C_2+G_2)\\-M_{12}(C_1+G_1)+M_{11}(C_2+G_2)\end{bmatrix}$$

متغیرهای حالت عبارتند از:

$$\dot x_1=x_3$$

$$\dot x_2=x_4$$

$$\dot x_3=\frac{M_{22}(U_1\text{-}C_1\text{-}G_1)}{M_{11}M_{22}\text{-}M_{12}^2}-\frac{M_{12}(U_2\text{-}C_2\text{-}G_2)}{M_{11}M_{22}\text{-}M_{12}^2}$$

$$\dot x_4=\frac{-M_{12}(U_1\text{-}C_1\text{-}G_1)}{M_{11}M_{22}\text{-}M_{12}^2}+\frac{M_{11}(U_2\text{-}C_2\text{-}G_2)}{M_{11}M_{22}\text{-}M_{12}^2}$$

برای شبیه سازی روبات دولینکی، پارامترها را مطابق جدول(۶-۶) تنظیم می‌کنیم:

جدول (۶-۶) پارامترهای ربات دولینکی

پارامتر	مقدار	واحد
طول لینک‌ها	$L_1 = L_2 = 1$	M
جرم لینک‌ها	$m_1 = m_2 = 1$	Kg
ممان اینرسی لینک‌ها	$I_1 = I_2 = 1$	$Kg.m^2$
جرم بار	$m_p = 2$	Kg

شبیه‌سازی

ابتدا از کتابخانه سیمولینک مسیرهای زیر را دنبال کنید و بلوک‌های مورد نظر را وارد پنجره مدل کنید:

جدول (۶-۷) مسیر بلوک‌های مورد استفاده در این مثال

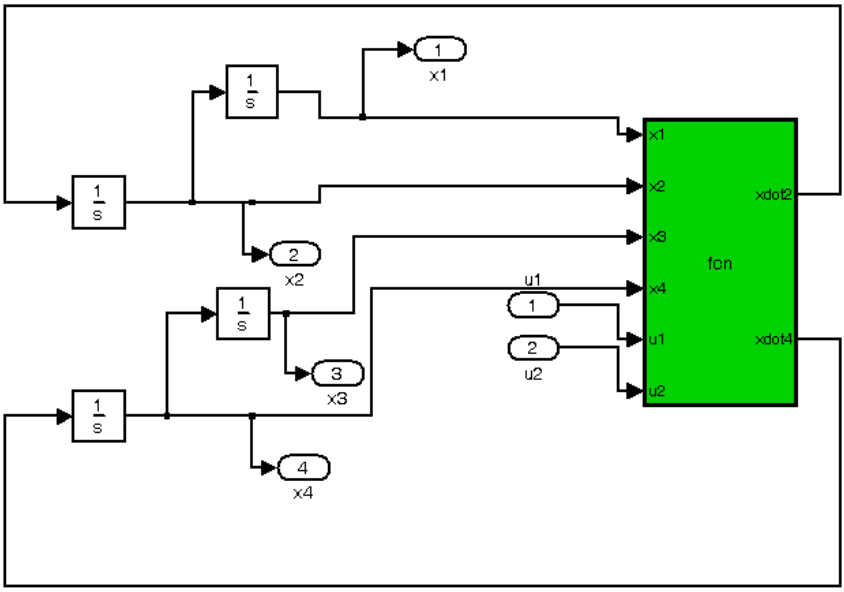
تعداد	مسیر بلوک
2	Simulink >>Sources>>Sine wave
1	Simulink >>Sinks>>Scope
1	Simulink >> Signal Routing>> Mux
4	Simulink >> Continuous>> Integrator
2	Simulink >>Sources>>In1
4	Simulink >>Sinks>>Out
1	Simulink >> User defined Function>> Embedded MATLAB Function

روی Embedded MATLAB Function دوبار کلیک کرده و در پنجره ظاهر شده دستورات زیر را تایپ کنید:

```
function [xdot2,xdot4] = fcn(x1,x2,x3,x4,u1,u2)

l1=1;  l2=1;
m1=1;  m2=1;
I1=1;  I2=1;
Lc1=0.5;   Lc2=0.5;
g=9.8;
Mp=2;

D11=(m1/3+m2)*l1^2+m2*l1*l2*cos(x3)+m2*l2^2/3;
D12=m2*l1*l2*cos(x3)/2+m2*l2^2/3;
D22=m2*l2^2/3;
delta=D11*D22-D12^2;
C=m2*l1*l2*sin(x3);
H=m2*l2*cos(x1+x3)/2;
A=C*(x2*x4+x4^2/2);
B=g*((m1/2+m2)*l1*cos(x1)+H);
E=C*x2^2/2+g*H;
```

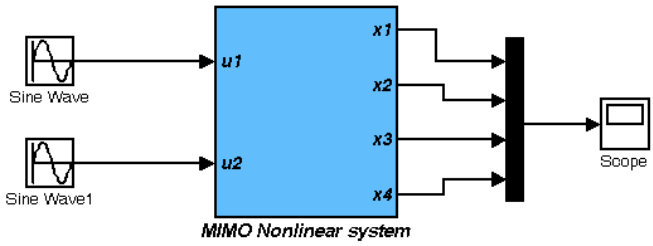
```
xdot2=(D22*u1+D22*A-B*D22-D12*u2+D12*E)/delta;
xdot4=(-D12*u1-D12*A+B*D12+D11*u2-D11*E)/delta;
```

بلوک‌ها را مانند شکل (۶-۲۳)با کلیک چپ کردن و جابه‌جا کردن هر یک، مرتب کنید و آنها را بهم اتصال دهید.



شکل(۶-۲۳) اتصال بلوک Fcn به بلوک‌های دیگر

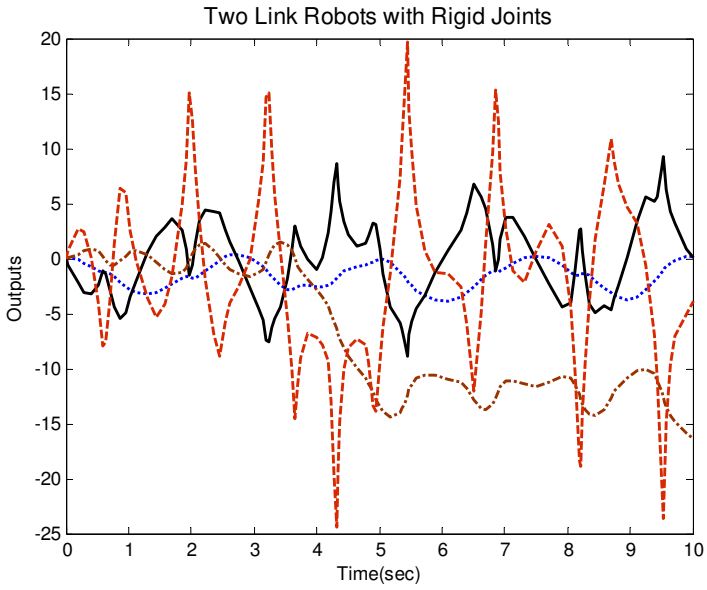
تمامی بلوک‌ها را انتخاب، سپس روی مجموعه آنها راست کلیک کرده و گزینه Create Subsystem را انتخاب کنید تا تمامی بلوک‌ها در یک بلوک قرار بگیرند. سپس بلوک‌های موج سینوسی را به عنوان ورودی سیستم قرار دهید.



شکل(۶-۲۴) نحوه اتصال بلوک سیستم غیرخطی به ورودی‌های سینوسی و Mux

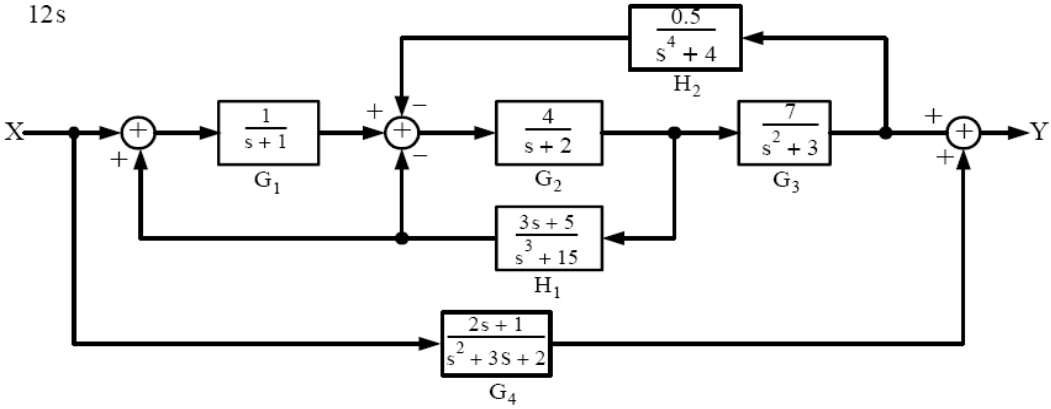
روی هر یک از بلوک‌های موج سینوسی دوبار کلیک کرده و در پنجره ظاهر شده مقدار Amplitude را برابر 0.2 قرار دهید. روی بلوک Mux دوبار کلیک کرده و در پنجره ظاهر شده مقدار Number of Inputs را برابر ۴ قرار دهید. سپس خروجی Subsystem را به ورودی‌های Mux متصل کنید و خروجی بلوک Mux را نیز به ورودی Scope وصل کنید.

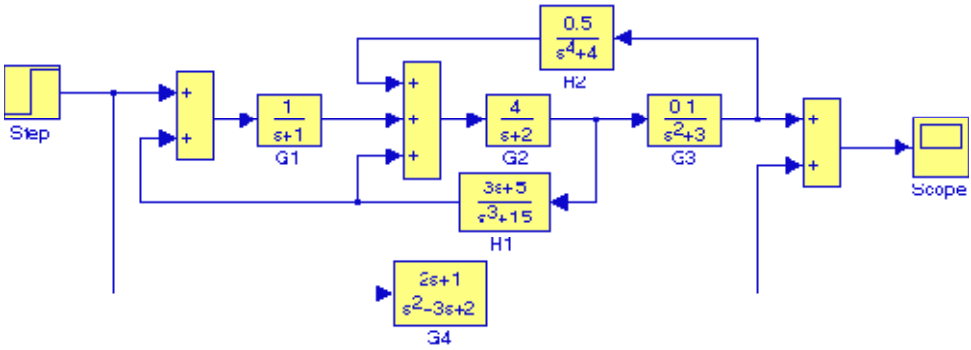
مدت زمان اجرای برنامه را روی ۱۰ ثانیه قرار داده و برنامه را اجرا کنید. سپس روی Scope دوبار کلیک کرده تا خروجی زیر را مشاهده کنید.



شکل(۶-۲۵) متغیرهای حالت خروجی

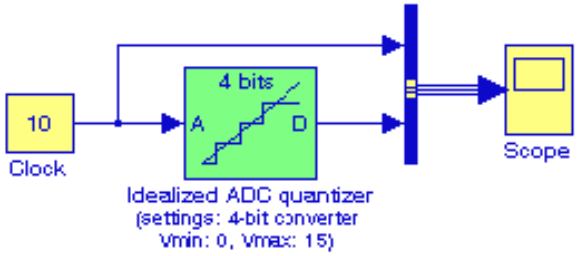
مثال: پاسخ سیستم کنترلی به همراه فیدبک زیر را به ورودی پله شبیه‌سازی کنید.



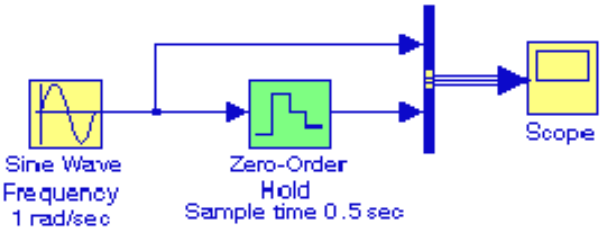


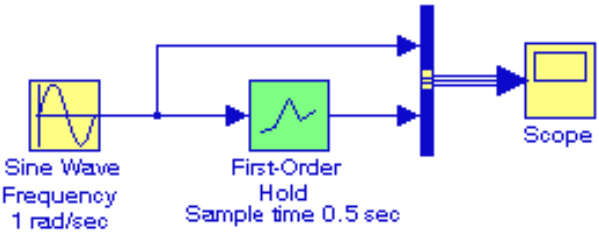
چند نمونه از کاربردهای سیمولینک در حوزه دیجیتال

- تبدیل آنالوگ به دیجیتال

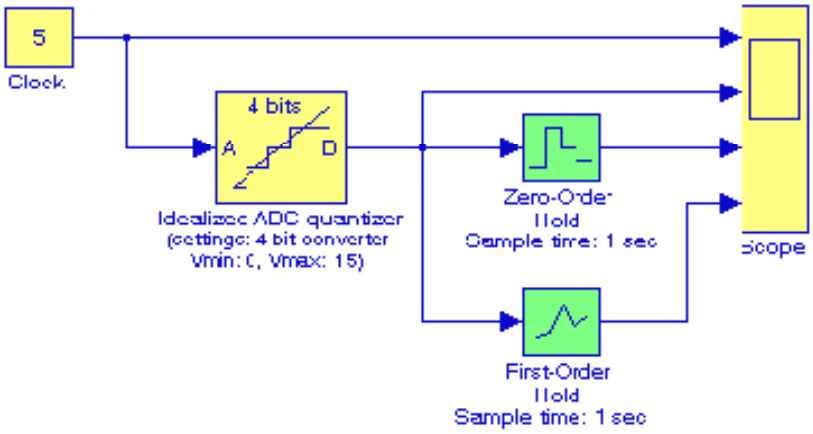


- نگهدارنده مرتبه صفر (Zero Order Hold) و نگهدارنده مرتبه یک (First Order Hold)





- مقایسه خروجی نگهدارنده مرتبه صفرو نگهدارنده مرتبه یک با ورودی یکسان



❖ تمرینات تکمیلی

۱- معادله دیفرانسیل Vander Pol زیر را در سیمولینک شبیه‌سازی نمائید.

$$\frac{d_2x}{dt^2} - k(1-x^2)\frac{dx}{dt} + x = d(t)$$

$$k = 10, d(t) = 10rect(2t), x_0 = \begin{bmatrix} -2 \\ 2 \end{bmatrix}$$

۲- یک پاندول ساده با معادله زیر را شبیه سازی کنید.

$$\frac{dw}{dt} = \frac{1}{J}(-mgl\sin\theta + T_a - B_m w), \quad \frac{d\theta}{dt} = w$$

$$T_a = 2,\ m = 1,\ l = 0.5,\ g = 9.8,\ B_m = 0.3$$

۳- سیستمی با معادلات دیفرانسیل خطی زیر در اختیار داریم. این سیستم را شبیه سازی کنید.

$$\frac{dx_1(t)}{dt} = -k_1x_1(t) + u(t)\ \ , x_1(t_0) = x_{10} = 20$$

$$\frac{dx_2(t)}{dt} = k_2x_1 - x_2\ \ , x_2(t_0) = x_{20} = 0$$

$$k_1 = 5, k_2 = 10$$

U(t) سیگنال سینوسی با بزرگی 50 و فرکانس 2Hz است .

۴- سیستم دینامیکی زیر را به صورت عددی شبیه سازی کنید.

$$\frac{dx_1(t)}{dt} = x_2\ , x_1(t_0) = x_{10}$$

$$\frac{dx_2(t)}{dt} = -x_1 + k_1x_2 - k_1k_2{x_1}^2x_2\ ,\ \ x_2(t_0) = x_{20}$$

$$I:\ \ k_1 = 1, k_2 = 5$$

$$II:\ k_1 = 1, k_2 = 100$$

$$III: k_1 = 0, k_2 = 100$$

شرایط اولیه برابر است با:

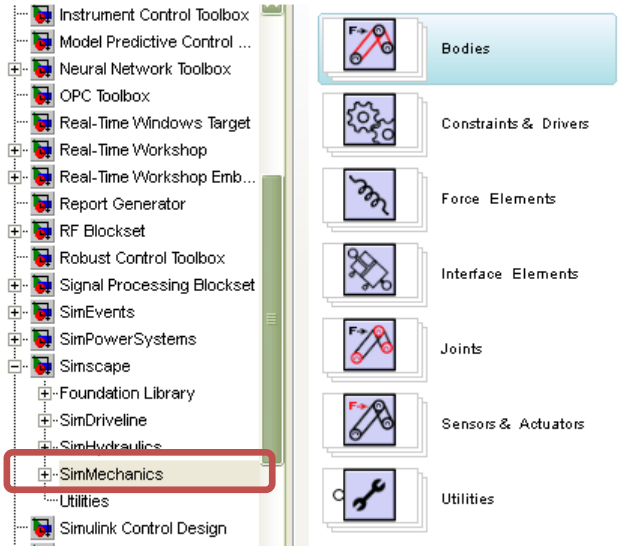
$$x_0 = \begin{bmatrix} x_{10} \\ x_{20} \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

فصل هفتم

مدل‌سازی در SimMechanics

۷-۱ کتابخانه SimMechanics

از این بسته نرم افزاری برای شبیه سازی حرکت انتقالی یا دورانی سیستم‌های مکانیکی استفاده می‌شود. مدل ساخته شده در این بسته، روابط هندسی و سینماتیکی اجسام سازنده یا به زبانی فیزیک مدل را دربر می‌گیرد. در شکل (۷-۱) بلوک‌های موجود در کتابخانه SimMechanics نشان داده شده است.



شکل (۷-۱) بلوک‌های SimMechanics

۷-۱-۱ بلوک‌های Body

در SimMechanics هر جسم صلب به‌وسیله یک بلوک Body و حداقل یک سیستم مختصات (CS=Coordinate System) چسبیده به آن بیان می‌شود. مبدا CS جسم در مرکز ثقل آن قرار دارد. بقیه CSها نیز می‌توانند به جسم متصل شوند. هر Joint یا محرک یا سنسور که به جسم متصل می‌شود می‌بایست به مبدا Body CS متصل گردد. بلوک‌های موجود در این قسمت عبارتند از:

Body, Machine Environment, Ground, Shared Environment

۷-۱-۲ بلوک‌های Constraint & Drivers

این بلوک‌ها برای اعمال محدودیت از پیش معلوم شده روی درجات آزادی بین Bodyها استفاده می‌شوند. این بلوک‌ها می‌توانند درجات آزادی را از یک ماشین بگیرند. یک Constraint یا Driver می‌بایست کاملاً دو جسم

را به هم متصل کند. بلوک‌های موجود در این قسمت عبارتند از: درایورهای Angle, Distance, Velocity, Linear و قیدهای Parallel, Gear.

۷-۱-۳ بلوک‌های Force Elements

این بلوک‌ها برای ایجاد نیروها و گشتاورها بین Bodyها استفاده می‌شوند. بلوک‌های موجود در این قسمت عبارتند از:

- Body Spring &Damper
- Joint Spring &Damper

۷-۱-۴ بلوک‌های Interface Elements

برای برقراری ارتباط بین یک مفصل prismatic با یک المان مکانیکی که حرکت انتقالی انجام می‌دهد و یا بین یک مفصل Revolute با یک المان مکانیکی که حرکت دورانی انجام میدهد، از این بلوک‌ها استفاده می‌شود. بلوک‌های موجود در این قسمت عبارتند از:

- Prismatic-Translational Interface
- Revolute-Rotational Interface

۷-۱-۵ بلوک‌های Joints

در SimMechanics یک Joint بیانگر درجات آزادی است که یک جسم نسبت به جسم دیگر دارد. بنابراین یک Joint الزاماً یک اتصال فیزیکی مابین دو جسم اعمال نمی‌کند. این بسته صرفاً درجات آزادی به مدل می‌افزاید، زیرا بلوک‌های Body حامل هیچ درجه آزادی نیستند. بلوک‌های موجود در این قسمت عبارتند از:

- **Assembled Joints** (Prismatic, Revolute, Six-DoF,…)
- **Disassembled Joints** (Disassembled Revolute, Disassembled Prismatic,…)
- **Mass less Connectors** (Revolute-Revolute, Revolute-Spherical, …)

۷-۱-۶ بلوک‌های Sensors & Actuators

با استفاده از بلوک‌های Actuators می‌توان:

۱. یک نیروی متغیر با زمان به یک Body یا Joint اعمال کرد.
۲. موقعیت، سرعت، و شتاب یک Joint یا Driver را به صورت تابعی از زمان مشخص کرد.
۳. موقعیت اولیه و سرعت اولیه یک نقطه ابتدایی را مشخص کرد.

با استفاده از بلوک‌های Sensors می‌توان:

۱. حرکت جسم را اندازه‌گیری کرد.
۲. حرکات Joint، نیرو، وگشتاورهای روی آن را اندازه‌گیری کرد.
۳. عکس العمل‌های Constraint، نیرو، و گشتاورها را اندازه‌گیری کرد.

بلوک‌های موجود در این قسمت عبارتند از:

- Body Actuator, Joint Actuator, Driver Actuator, Variable Mass & Inertia Actuator,…
- Body Sensor, Joint Sensor, Constraint & Driver Sensor, …

با توجه به نوع مفصل(Joint) در هر یک از حرکت‌های انتقالی، دورانی و کروی، سنسور می‌تواند موارد زیر را اندازه‌گیری کند:

جدول(۷-۱) انواع مفاصل در هر یک از حرکت‌های انتقالی، دورانی و کروی

Motion Type	Joint Primitive Type	Measurements
Translational	Prismatic	Linear position Linear velocity Linear acceleration
Rotational	Revolute	Angle Angular velocity Acceleration
Spherical	Spherical	Quaternion Quaternion first derivative Quaternion second derivative

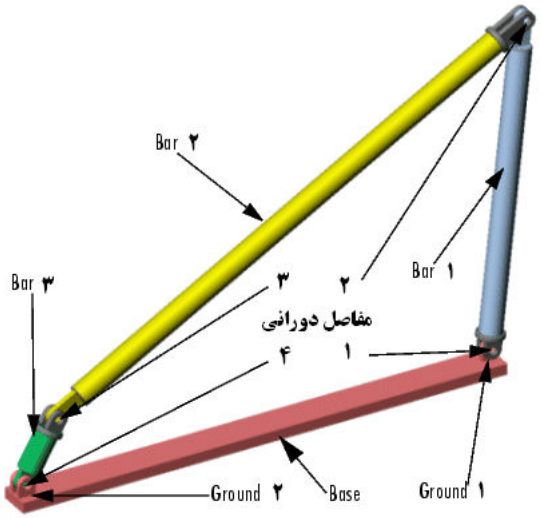
۷-۱-۷ بلوک‌های Utilities

بلوک‌های مفید مختلف در ایجاد مدل را فراهم می‌سازد. بلوک‌های موجود در این قسمت عبارتند از:

Connection Port, Continuous Angle, Mechanical Branching Bar

۷-۲ مدل سازی سیستم مکانیکی چهار میله‌ای حلقه بسته

مثال: مدل سازی سیستم مکانیکی چهار میله‌ای حلقه بسته زیر را در SimMechanics انجام دهید.



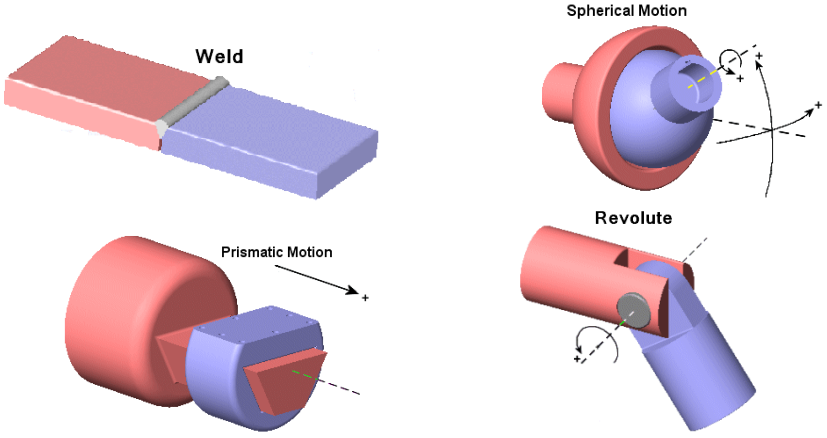
شکل (۷-۲) سیستم مکانیکی چهارمیله‌ای

سه میله متحرک بالا را در نظر بگیرید که تنها می‌توانند در یک صفحه حرکت کنند. بنابراین هر میله، ۲ درجه آزادی انتقالی و یک درجه آزادی دورانی دارد. ولی تنها یک درجه آزادی مستقل برای سیستم وجود دارد و بقیه درجات آزادی وابسته به آن می‌باشند.

تعداد درجات آزادی مستقل را می‌توان از رابطه زیر به‌دست آورد:

تعداد درجات آزادی مستقل = تعداد درجات آزادی جسم + تعداد درجات آزادی اصلی – تعداد محدودیت‌های حرکتی

- تعداد درجات آزادی جسم، شش برابر حاصل تفریق (تعداد Bodyها - تعداد مفاصل) است. اگر سیستم در دو بعد حرکت کند باید به جای شش عدد سه و اگر حرکت سیستم تنها در یک بعد باشد عدد شش با عدد یک جایگزین می‌شود. درمثال بالا حرکت در دوبعد است؛ بنابراین درجات آزادی جسم برابر $3(4-3)=3$ است.
- برای به‌دست آوردن تعداد درجات آزادی اصلی به ازای مفاصل prismatic(P) و Revolute(R) عدد یک، به ازای مفصل کروی Spherical(S) عدد سه و به ازای مفصل جوشکاری Weld(W) عدد صفر قرار دهید و مجموع درجات آزادی را حساب کنید.



شکل (۷-۳) درجات آزادی انواع مفاصل

- تعداد **محدودیت‌های حرکتی** با استفاده از تعداد Constraintها و Driverها به‌دست می‌آید.

در این مثال تعداد درجات آزادی برابر $3(3-4)+4=1$ است.

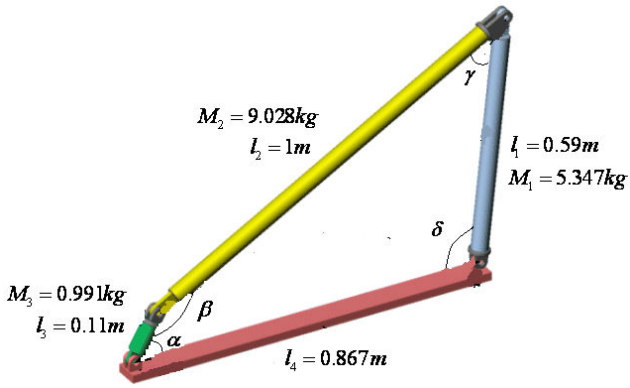
معادلات حاکم بر مدل به‌صورت زیر می‌باشد:

$$\alpha+\beta+\gamma+\delta=2\pi\,.$$

$$l_1^2+l_2^2-2l_1l_2Cos\gamma=l_3^2+l_4^2-2l_3l_4Cos\alpha\,.$$

$$l_2^2+l_3^2-2l_2l_3Cos\beta=l_1^2+l_4^2-2l_1l_4Cos\delta\,.$$

اگر α را به عنوان درجه آزادی مستقل در نظر بگیریم، می‌توان β, γ, δ را از α به‌دست آورد.

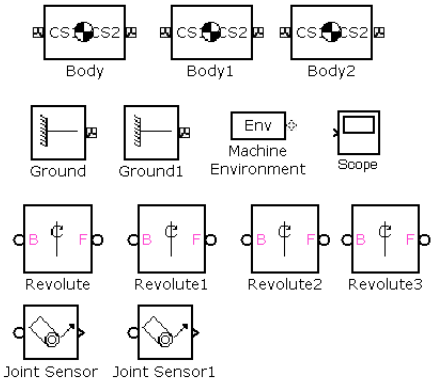


شکل (۷-۴) سیستم مکانیکی چهارمیله‌ای

جهت ایجاد مدل یک پنجره جدید باز کرده و بلوک‌های زیر را از کتابخانه سیمولینک وارد پنجره مدل کنید. سپس مطابق شکل زیرآنها را مرتب و به‌یکدیگر متصل کنید.

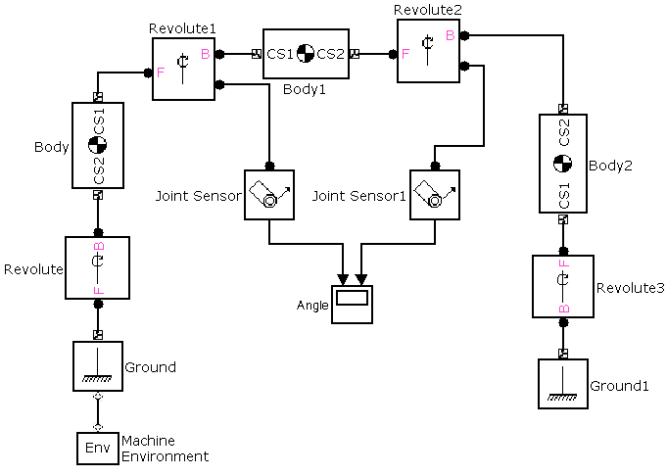
جدول(۷-۲) مسیر بلوک‌های مورد استفاده در این مثال

تعداد	مسیر بلوک
1	Simulink >>Sinks>>Scope
2	Simscape>>SimMechanics>>Bodies>>Ground
1	Simscape>>SimMechanics>>Bodies>>Machine Environment
3	Simscape>>SimMechanics>>Bodies>>Body
4	Simscape>>SimMechanics>>Joints>>Revolute
2	Simscape>>SimMechanics>>Sensors & Actuators>>Joint Sensor



شکل (۷-۵) بلوک‌های SimMechanics مورد استفاده

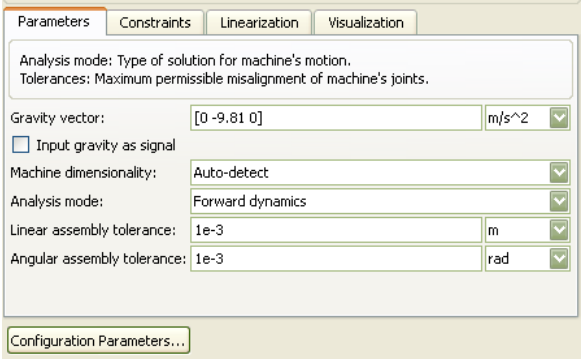
نکته: به جهت اتصال F,B در Revoluteها توجه کنید. همیشه از یک F به یک B می‌رسیم.



شکل (۷-۶) نحوه اتصال بلوک‌ها به‌یکدیگر

تنظیمات زیر را برای بلوک‌های مورد نظر انجام دهید:

- روی بلوک Machine Environment دوبار کلیک کرده و در قسمت Parameters مقدار Gravity Vector را برابر [0 -9.81 0] وارد کنید. این بردار نشان می‌دهد که جاذبه درخلاف جهت Y به سیستم وارد می‌شود. نوع مود آنالیزی را Forward Dynamic انتخاب کنید. این مود از حلگرهای معادلات دیفرانسیل معمولی (ODE) سیمولینک استفاده می‌کند تا معادلات نیوتنی را حل کند. در واقع در این مود با معلوم بودن نیروها وگشتاورهای وارد شده روی سیستم می‌توان حرکت حاصل شده را به‌دست آورد. ولی در مود Inverse حل معکوس مسأله انجام می‌شود؛ یعنی نیروها و گشتاورهایی که برای ایجاد یک مجموعه حرکات خاص مورد نیاز است را پیدا می‌کند. این مود تنها برای حل نمودارهای باز بدون حلقه‌های بسته استفاده می‌شود. در غیر این صورت باید مود Kinematics را انتخاب کرد.



شکل (۷-۷) تنظیمات بلوک Machine Environment

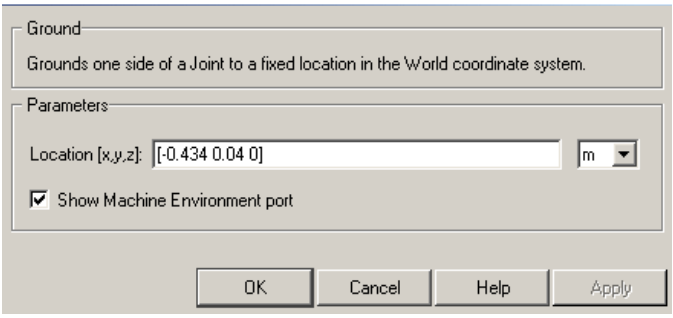
روی configuration parameters کلیک کرده سپس از پنجره درختی سمت راست SimMechanics را انتخاب نمائید. گزینه Show Animation during simulation را فعال کنید تا هنگام اجرا انیمیشن مدل به صورت خودکار ظاهر شود.

- روی بلوک‌های Ground تنظیمات زیر را انجام دهید:

جدول(۷-۳) تنظیمات Groundها

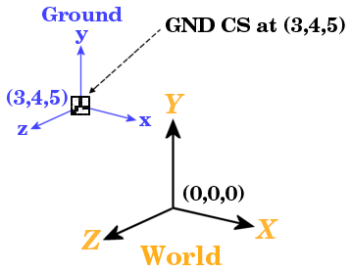
بلوک	Location	Show machine environment
Ground	[-0.434 0.04 0]	فعال شود
Ground1	[0.433 0.04 0]	غیرفعال شود

بلوک Ground یک نقطه روی جسم ایده‌آل با جرم بی‌نهایت و ابعاد بی‌نهایت را مدل می‌کند، به‌گونه‌ای که مانند یک محیط ثابت شده برای سیستم‌ها به‌کار می‌رود. این بلوک به کاربر این امکان را می‌دهد که درجات آزادی سیستم نسبت به محیطش را تعیین کند. برای معتبر بودن مدل در SimMechanics باید حداقل یک بلوک Ground متصل شده به یک بلوک جسم یا Joint وجود داشته باشد.



شکل (۷-۸) تنظیم پارامترهای بلوک Ground1

این نرم افزار شامل یک سیستم مختصات اصلی داخلی و چارچوب مرجع به نام World است که همه Groundها در این سیستم مختصات جهانی قرار دارند. محورهای این سیستم مختصات موازی با سیستم مختصات World بوده و به‌طور پیش فرض بر مبدا سیستم World منطبق هستند. البته به کاربر این امکان را می‌دهد که مبدا GND را به نقاط دیگری در سیستم مختصات جهانی انتقال دهد.

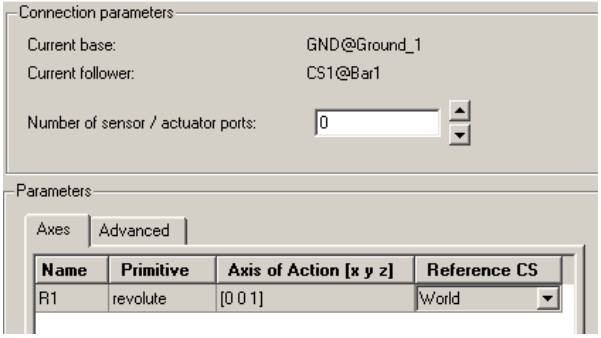


شکل (۷-۹) مبدا مختصات جهانی و GND

- روی هر یک از بلوک‌های Revolute دوبارکلیک کرده و تنظیمات زیر را انجام دهید:

جدول(۷-۴) تنظیمات مفاصل

Connection parameters>>Number of sensors/actuators	**Parameters>>Axis of rotation**	**بلوک**
0	[0 0 1]	Revolute
1	[0 0 1]	Revolute1
1	[0 0 1]	Revolute2
0	[0 0 1]	Revolute3

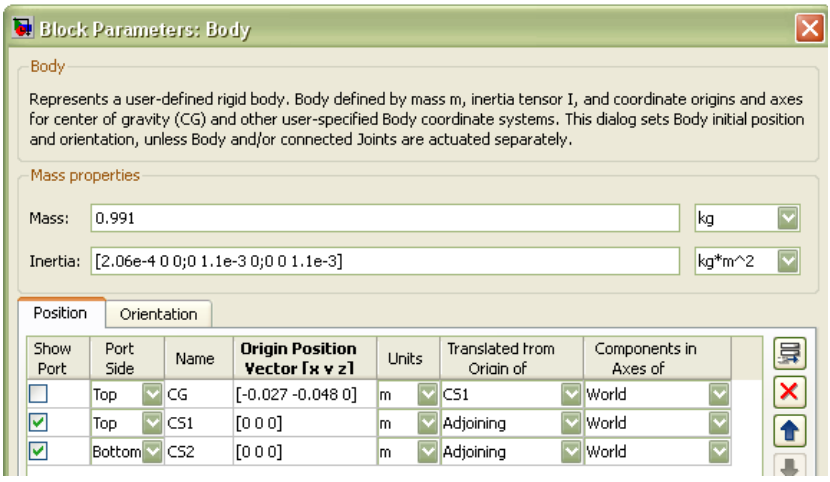


شکل (۷-۱۰) تنظیمات بلوک Revolute1

- روی بلوک‌های Body دوبارکلیک کرده و در پنجره‌های ظاهر شده تنظیمات زیر را انجام دهید:

تنظیمات Body:

واژه Body به هر نقطه یا به‌ویژه هر موضوع قابل توسعهٔ صلبی اطلاق می‌شود که دارای جرم متناهی باشد.

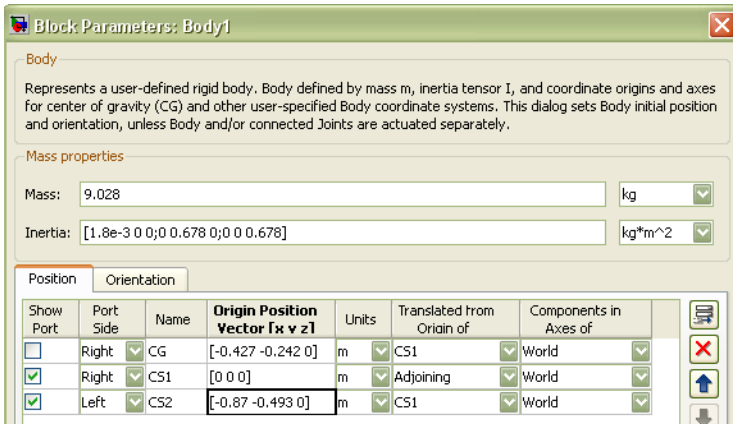


شکل (۷-۱۱) تنظیمات Body

اجسام در SimMechanics درجه آزادی ندارند. پنجره باز شده بلوک Body به کاربر این امکان را می‌دهد که مبدا یک سیستم مختصات Body و جهت گیری[1] جسم را تغییر دهد. در این بسته نرم افزاری برای تعیین مبدا و جهت گیری سیستم‌های مختصات جسم می‌توان اطلاعات را نسبت به یکی از موارد زیر وارد کرد:

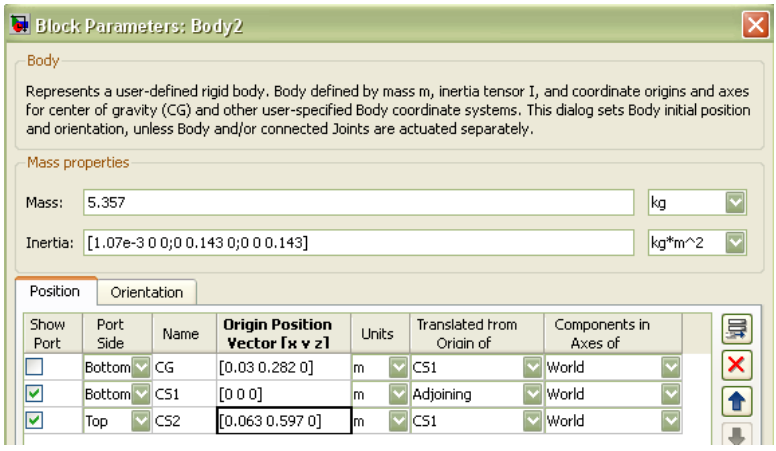
- World CS
- هر CS دیگری روی همین جسم
- Adjoining CS (سیستم مختصات روی جسم نسبت به سیستم مختصات جسم مجاور که توسط یک Joint و قید به این جسم متصل شده است)

تنظیمات Body1:



شکل (۷-۱۲) تنظیمات Body1

تنظیمات Body2:



شکل (۷-۱۳) تنظیمات Body2

[1] Orientation

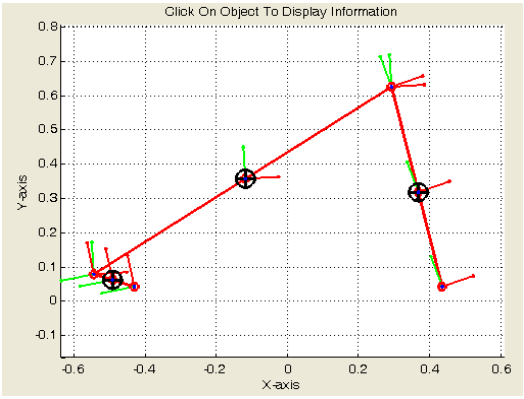
- روی بلوک‌های Joint sensor دوبار کلیک کرده و در پنجره‌های ظاهر شده تنظیمات زیر را انجام دهید:

جدول(۷–۵) تنظیمات سنسورها

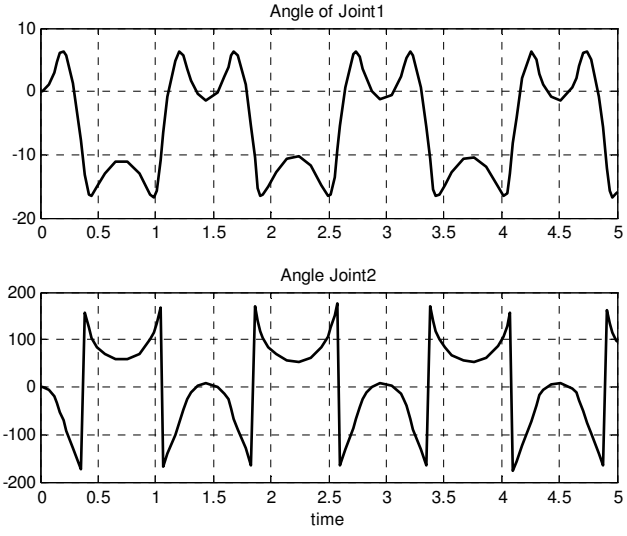
بلوک	Measurement
Joint sensor	Angle
Joint sensor1	Angle

- روی بلوک Scope دوبار کلیک کرده، در پنجره ظاهر شده در نوار ابزار بر روی Parameters کلیک نمائید. سپس مقدار Number of axis را برابر ۲ قرار دهید.

در مرحله آخر زمان شبیه‌سازی را ۵ قرار داده، روی Start Simulation کلیک کنید تا شبیه سازی آغاز شود. سپس روی Scope دوبارکلیک کرده تا نتایج را مشاهده کنید. در انتها مدل را با نام Fourbar ذخیره کنید.



شکل (۷–۱۴) شبیه‌سازی مکانیزم چهارمیله‌ای



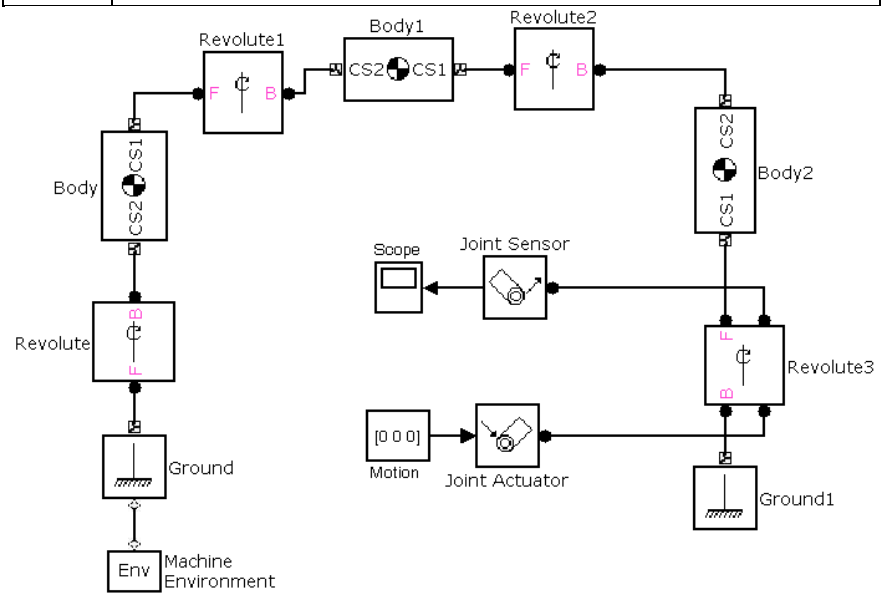
شکل (۷–۱۵) تغییرات زاویه‌های مفصل اول و دوم برحسب زمان

حال می‌خواهیم مقدارگشتاوری را که لازم است به Revolute3 وارد شود تا سیستم چهار میله ای قسمت اول ساکن شود، به‌دست آوریم.

در ابتدا یک کپی از فایل Fourbar بگیرید و آن را با نام Fourbar1 ذخیره کنید. بلوک Joint Sensor1 و Scope را حذف کرده و بلوک‌های زیر را وارد پنجره مدل سازی کنید.

جدول(۷-۶) مسیر بلوک‌های مورد استفاده در این مثال

تعداد	مسیر بلوک
1	Simulink >>Sinks>>Scope
1	Simulink >>sources>>Constant
1	Simscape>>SimMechanics>>Sensors & Actuators>>Joint Actuators



شکل (۷-۱۶) نحوه اتصال بلوک‌ها به‌یکدیگر در حالت جدید

تنظیمات بلوک‌ها

- روی بلوک‌های Revolute1 و Revolute2 دوبار کلیک کرده و مقدار Number of sensors/actuators را برای هر یک از بلوک‌ها به ۲ افزایش دهید.
- روی Constant دوبار کلیک کرده و مقدار [0 0 0] را وارد کنید.

- روی بلوک Joint Actuator دوبار کلیک کرده و گزینه Motion را انتخاب کنید. سپس خروجی Constant را به ورودی Joint Actuator وصل کنید تا تغییرات زاویه‌ای، سرعت زاویه‌ای، و شتاب زاویه‌ای صفر به مفصل ۳ اعمال شود.
- روی بلوک Joint Sensor دوبارکلیک کرده و تنها گزینه Computed Torque را انتخاب کنید تا این مقدار محاسبه شود.
- روی Machine Environment کلیک کرده و نوع مود تحلیلی را Kinematics انتخاب کنید.

در انتها روی Start Simulation کلیک کنید تا شبیه سازی آغاز شود. سپس روی Scopeکلیک کرده تا نتیجه را که همان مقدار 27.2915N.m است مشاهده کنید.

۷–۳ شبیه سازی سیستم جرم و فنر ساده

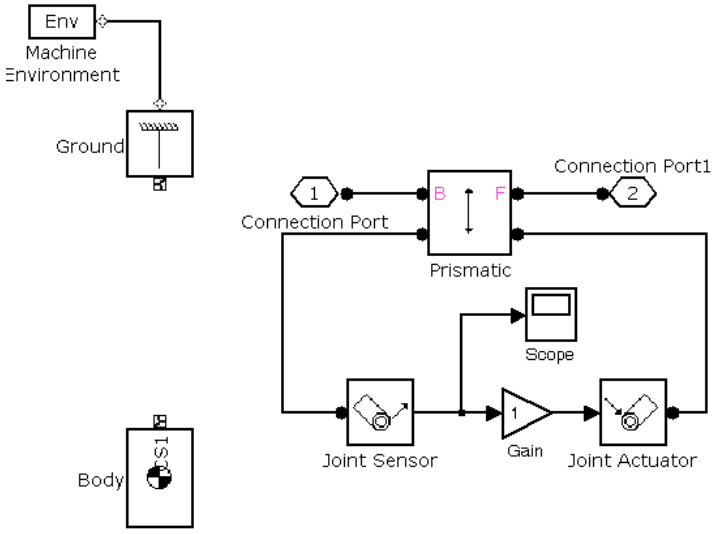
مثال: شبیه سازی سیستم جرم و فنر ساده را در محیط SimMechanics انجام دهید.

در این سیستم جرم وزنه 5kg و سختی فنر 600N/m می‌باشد. وزنه از موقعیت 10cm رها می‌شود و تحت جاذبه زمین شروع به حرکت نوسانی می‌کند. نمودار تغییر مکان وزنه نسبت به زمان را رسم کنید.

جهت ایجاد مدل یک پنجره جدید باز کنید و بلوک‌های زیر را از کتابخانه سیمولینک وارد پنجره مدل کنید و مطابق شکل زیر آنها را مرتب و به‌هم متصل کنید.

جدول(۷–۷) مسیر بلوک‌های مورد استفاده در این مثال

مسیر بلوک	تعداد
Simulink >>Sinks>>Scope	1
Simscape>>SimMechanics>>Bodies>>Ground	1
Simscape>>SimMechanics>>Bodies>>Machine Environment	1
Simscape>>SimMechanics>>Bodies>>Body	1
Simscape>>SimMechanics>>Sensors & Actuators>>Joint Actuator	1
Simscape>>SimMechanics>>Sensors & Actuators>>Joint Sensor	1
Simscape>>SimMechanics>>Utilities>>Connection port	2
Simscape>> SimMechanics >>Joints>>Prismatic	1
Simulink>>Math operation>>Gain	1



شکل (۷-۱۷) نحوه اتصال بلوک‌ها برای ایجاد سیستم جرم و فنر

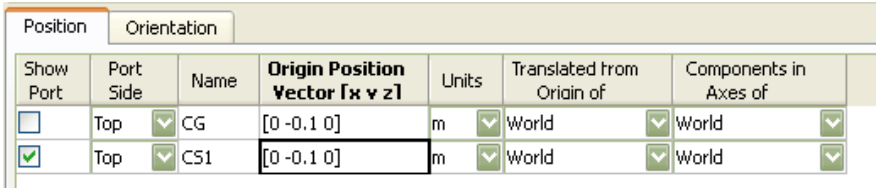
تنظیمات بلوک‌ها

- روی بلوک Machine Environment دوبار کلیک کرده و در قسمت Parameters مقدار Gravity Vector را [0 -9.81 0] وارد کنید. این بردار نشان می‌دهد که جاذبه درخلاف جهت Y به سیستم وارد می‌شود. روی configuration parameters کلیک کرده سپس از پنجره درختی سمت راست SimMechanics را انتخاب نمائید. سپس گزینه Show Animation during simulation را فعال کنید تا هنگام اجرا انیمیشن مدل به صورت خودکار ظاهر شود.
- روی بلوک Ground تنظیمات زیر را انجام دهید:

جدول(۷-۸) تنظیمات بلوک Ground

بلوک	Location	Show machine environment
Ground	[0 0 0]	فعال شود

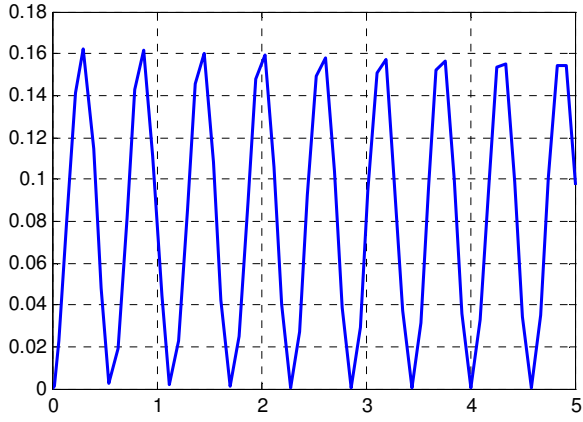
- روی بلوک Body دوبار کلیک کرده و در پنجره باز شده مقدار جرم را برابر 5kg قرار داده و تنظیمات زیر را انجام دهید:

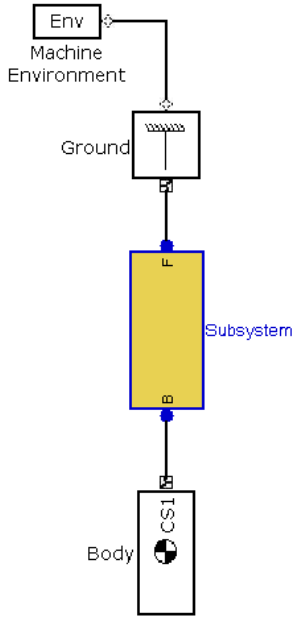


شکل (۷-۱۸) تنظیمات بلوک Body

- روی بلوک Prismatic دوبار کلیک کرده و در پنجره باز شده در قسمت Connection Parameters مقدار Number of sensor/actuator ports را به ۲ افزایش دهید از طرفی Axis of translation را برابر [0 1 0] قرار دهید.
- روی بلوک Gain دوبار کلیک کرده و مقدار 600- را وارد کنید.
- روی بلوک Joint sensor دوبار کلیک کرده و در قسمت Measurement گزینه position را انتخاب کنید و گزینه output selected parameters as one signal را غیرفعال کنید.
- بلوک‌های متصل به‌هم سمت راست را انتخاب کرده و روی آنها کلیک راست کرده و گزینه Create Subsystem را انتخاب کنید. سپس بلوک subsystem را مطابق شکل زیر به Ground و Body وصل کنید.

زمان شبیه سازی را ۵ ثانیه قرار داده و روی start simulation کلیک کنید.





شکل (۷-۱۹) سیستم جرم و فنر و منحنی تغییر موقعیت جرم m

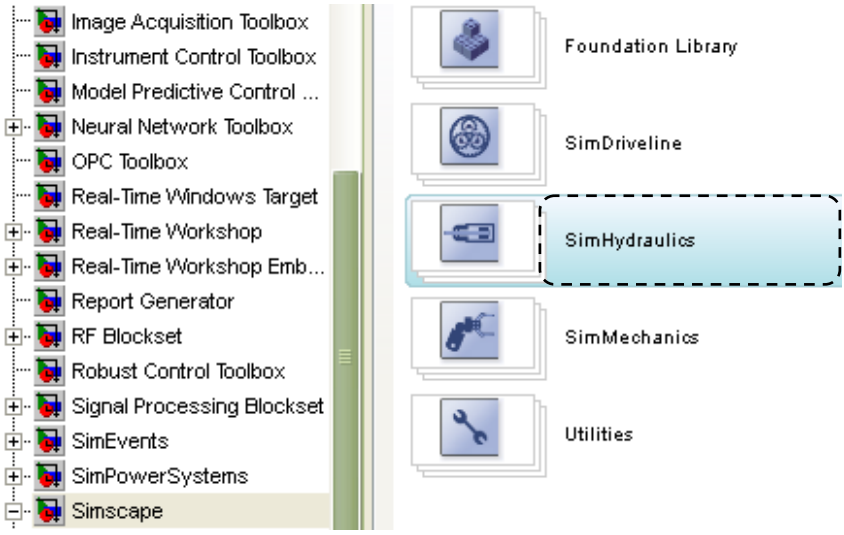
فصل هشتم

مدل‌سازی و شبیه سازی سیستم‌های هیدرولیکی

۸-۱ کتابخانه SimHydraulic

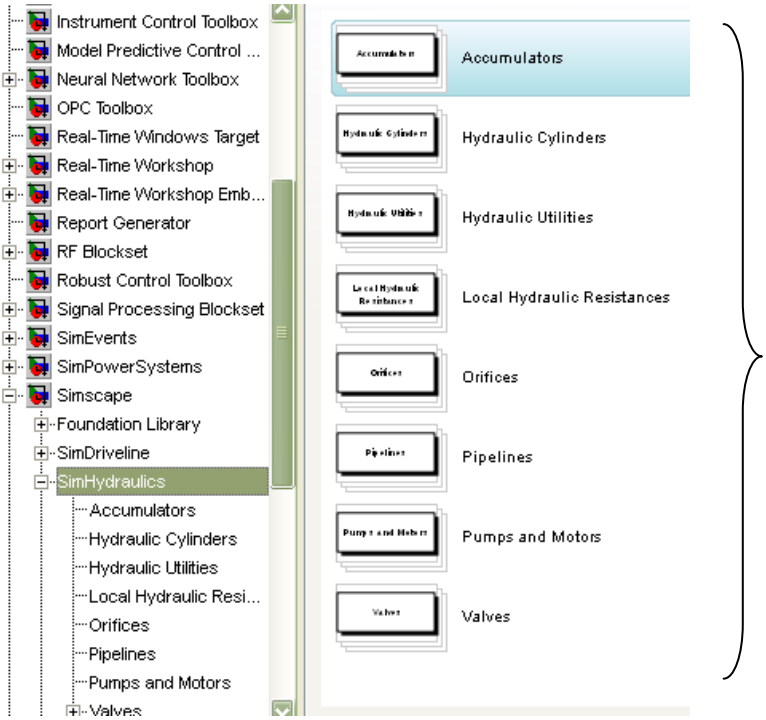
نرم افزار SimHydrolics مخصوص شبیه‌سازی سیستم‌های هیدرولیکی و سیستم‌های کنترلی می‌باشد. نرم افزار هیدرولیکی یک آنالیز گذرا از سیستم‌های مکانیکی و هیدرولیکی انجام می‌دهد. البته توجه داشته باشید که این نرم‌افزار توانایی مدل‌کردن سیستم‌های زیر را ندارد:

- انتقال سیالات
- سیستم‌های فاضلاب و تامین آب
- سیستم پارامترهای توزیع شده



شکل(۸-۱) SimHydraulic

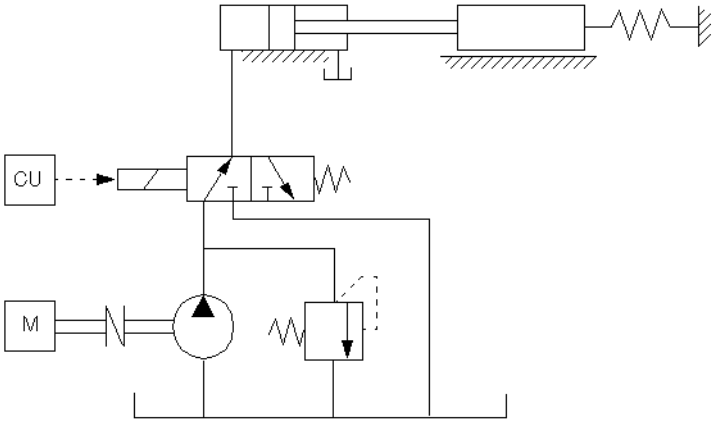
بسته نرم افزاری SimHydrolics بر پایه ثابت بودن دمای سیالات در طول زمان شبیه سازی است. از این نرم‌افزار می‌توان برای تحلیل‌های هیدرولیکی، الکتریکی، انتقال و دوران یک بعدی عناصر مکانیکی نیز استفاده نمود.



شکل(۸-۲) بلوک‌های SimHydraulic

۸-۲ مدل سازی سیستم‌های هیدرولیکی

یک سیستم ساده هیدرولیکی در شکل زیر نشان داده شده است. می‌خواهیم سیستم را در محیط SimHydraulic مدل‌سازی کرده، سپس پارامترها را تنظیم و آن را شبیه سازی نمائیم.



شکل(۸-۳) سیستم ساده هیدرولیکی

برای مدل کردن سیستم هیدرولیکی بالا مراحل زیر را انجام دهید:

۱. در پنجره Simulink بر روی New کلیک کرده و یک مدل جدید ایجاد کنید.

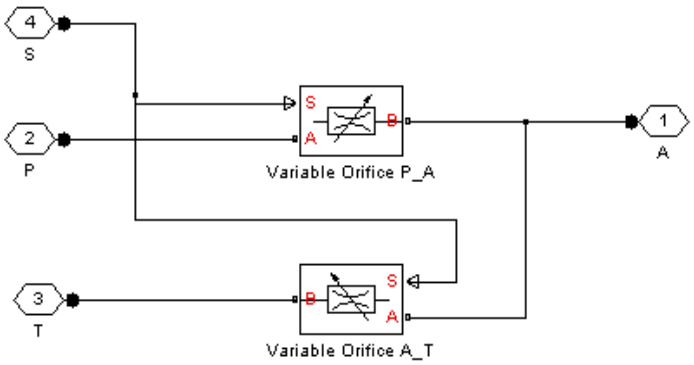
۲. از مسیر Simscape >> Foundation Library>>Hydraulic>>Hydraulic sensors and source بلوک Ideal Hydraulic pressure source را به داخل پنجره مدل سازی درگ کنید.

۳. از مسیر Simscape >>SimHydraulic >>Hydraulic cylinder بلوک Single acting hydraulic cylinder را به داخل پنجره مدل سازی درگ کنید.

فرض: از نشتی سیلندر، ممان اینرسی، اصطکاک، و خاصیت فنری پیستون صرف نظر می‌شود.

۴. برای مدل کردن شیر از مسیر Simscape >>SimHydraulic >> Hydraulic >>Directional valves شیر سه راهه 3-Way Directional Valve و از مسیر Simscape >>SimHydraulic >> Hydraulic مدل 2-position valve actuators را انتخاب کنید.

توجه ۱: بلوک 3-way directional valve برای مدل کردن شیر سه راهه سه وضعیتی است و سه پورت هیدرولیکی دارد که عبارتند از: پورت ورودی P، پورت خروجی A و پورت بازگشت T و یک پورت سیگنال فیزیکی S که برای کنترل موقعیت اسپول است. شیر از دو اریفیس متغیر ساخته شده است. در شکل (۸-۴) مدل گسترده شیر نشان داده شده است.



شکل(۸-۴) بلوک 3-way directional valve

یکی از اریفیسها Orifice_ PA نامیده می‌شود که در مسیر P به A نصب می‌شود. اریفیس دیگر Orifice _ AT است که در مسیر A به T قرار دارد. از طرفی چون از مدل اریفیس متغیر استفاده می‌کنیم یکی از سه حالت زیر را انتخاب می‌نمائیم:

۱. با ماکزیمم سطح و بازشدگی: هرگاه دیتا براساس مقدار ماکزیمم سطح اریفیس و بیشترین کورس المان کنترلی موجود باشد.

۲. براساس سطح مقطع و درصد بازشدگی: هرگاه در جدول تغییرات سطح اریفیس بر حسب جابه‌جایی اسپول موجود باشد$A=A(h)$.

۳. مشخصات دبی و فشار: هر گاه جدول دو بعدی فشار و دبی در کاتالوگ موجود باشد $q=q(p,h)$.

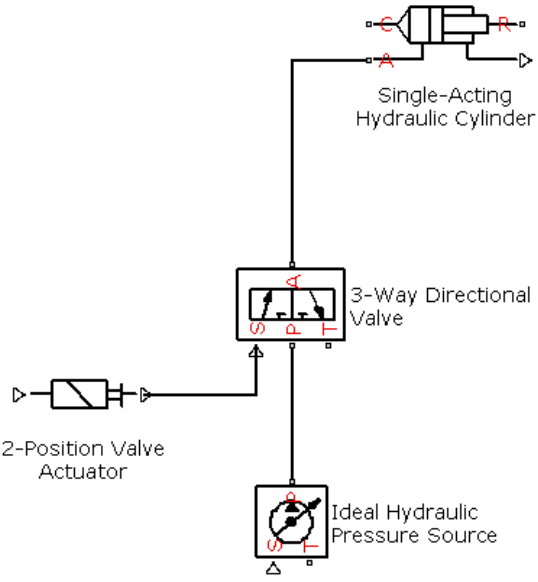
فرضیات

- اینرسی سیال در نظر گرفته نمی‌شود.
- از اینرسی و اصطکاک شفت اسپول صرف نظر می‌شود.
- هر دو اریفیس هم شکل و هم اندازه فرض می‌شوند.

توجه۲: بلوک (2-Position Valve Actuator) برای مدل کردن عملگری است که در شیرهای چند وضعیتی برای کنترل موقعیت شیر استفاده می‌شود. این عملگر برای کنترل موقعیت شیرهای دو وضعیتی استفاده می‌شود. مهمترین پارامترهای این عملگر کورس، مدت زمانهای Switch-on, Switch-off آن می باشد.

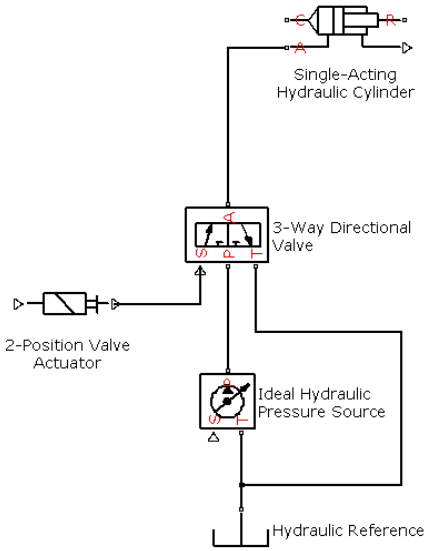
فرضیات: از اینرسی و اصطکاک، نیروی فنری و هیدرولیکی اعمالی بر روی پین عملگر صرف نظر می‌شود.

۵. بلوکها را همانند شکل (۸-۵) به هم وصل کنید.



شکل(۸-۵) نحوه اتصال بلوک‌ها به‌یکدیگر

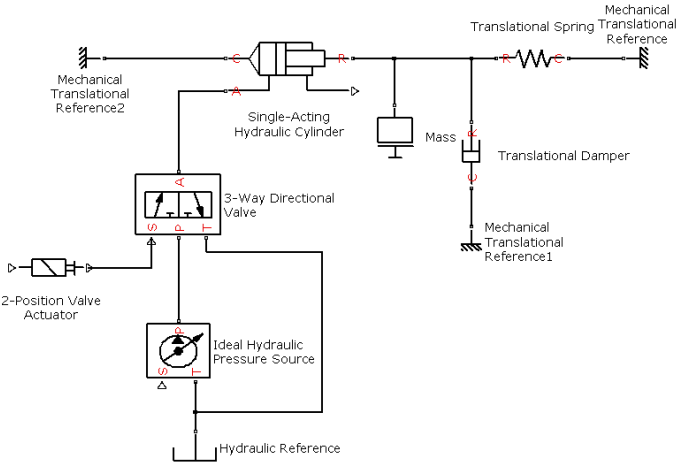
۶. پورت T شیر و منبع فشار باید به تانک متصل شوند. برای این منظور از مسیر Simscape >>Foundation library >>Hydraulic >> Hydraulic element بلوک Hydraulic Reference را به مدار اضافه کنید.



شکل(۸-۶) نحوه اتصال بلوک‌ها به‌یکدیگر

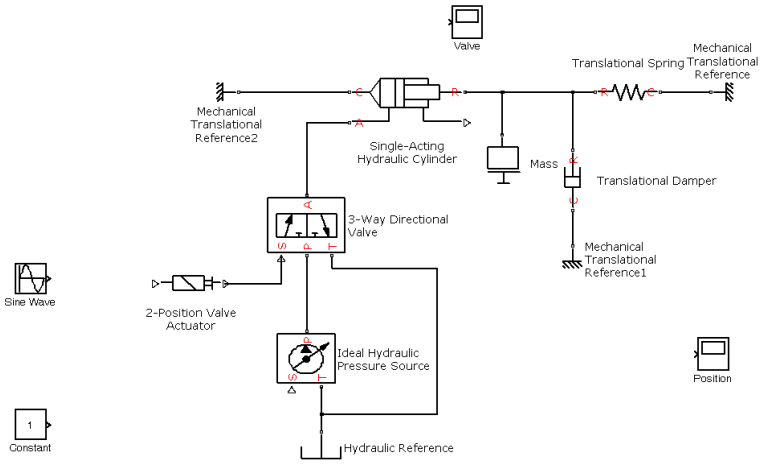
بلوک رفرنس (Hydraulic Reference) برای مدل کردن اتصال به فشار اتمسفر می‌باشد. رفرنس تمام المانهای هیدرولیکی مثل پورت مکش پمپ، پورت برگشت شیر و سیلندر، و خطوط لوله اتمسفر می‌باشد (این المانها از طریق این بلوک به اتمسفر وصل می‌شوند).

۷. برای مدل کردن فنر، جرم، و دمپر از مسیر Simscape >>Foundation library>Mechanical Translation Elements >> بلوکهای فنر، جرم، و دمپر و Mechanical Translational Reference را به مدار اضافه کنید. یک طرف سیلندر را ثابت و طرف دیگر را به المانهای مکانیکی وصل کرده، و بقیه بلوکها را مانند شکل به مدار وصل کنید.



شکل(۸-۷) نحوه اتصال بلوک‌ها به‌یکدیگر

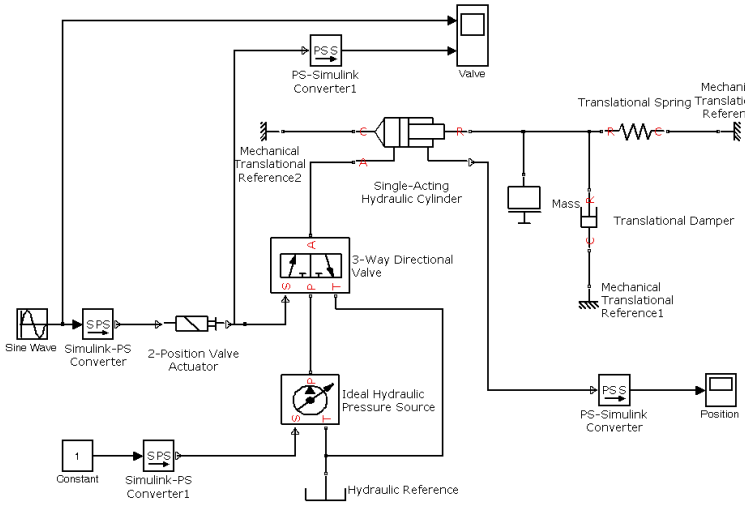
۸. برای اضافه کردن Scope و Source از مسیر Simulink >> Source library بلوک Constant , Sine wave و از مسیر Simulink >> Sink library دو بلوک اسکوپ کپی کنید و اسامی آنها را مانند شکل عوض کنید.



شکل(۸–۸) نحوه اتصال بلوک‌ها به‌یکدیگر

۹. روی اسکوپ Valve دوبار کلیک کرده و روی Parameters کلیک نمائید و Number of axes را مساوی ۲ قرار دهید.

۱۰. برای وصل کردن Source یا Scope به مدار هیدرولیکی باید از بلوک تبدیل مناسبی استفاده کرد. از مسیر Simscape >> Utilities دو بلوک Simulink–ps converter و دو بلوک ps-simulink converter را به مدل اضافه کنید.

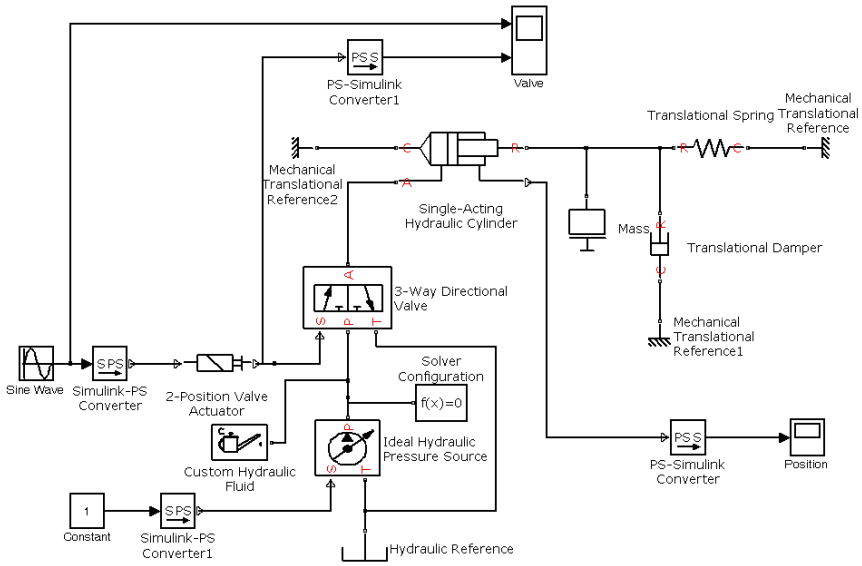


شکل(۸–۹) نحوه اتصال بلوک‌ها به‌یکدیگر

۱۱. برای وارد کردن مشخصات سیال هیدرولیکی از مسیر Simscape >>Simhydraulic>> Hydraulic utilities بلوک Hydraulic fluid را اضافه کنید و مانند شکل در مدار قرار دهید .

بلوک Hydraulic fluid به شما اجازه می‌دهد تا نوع سیال را تعریف کنید. خواص سیال مانند ویسکوزیته سینماتیکی، چگالی و مدول بالک را می‌توان برای تمام بلوکهای موجود در مدار تعریف کرد. این خواص در طول شبیه سازی ثابت با زمان فرض می‌شود و چگالی نیز با توجه به نوع سیال تعیین می‌شود. اما برای تعیین ویسکوزیته باید دمای کاری و برای تعیین مدول بالک باید نسبت هوای مخلوط در سیال[1] داده شود.

۱۲. برای اضافه کردن بلوک Solver configuration از مسیر Simscape >> Utilities library بلوک آن را به مدار اضافه کنید.

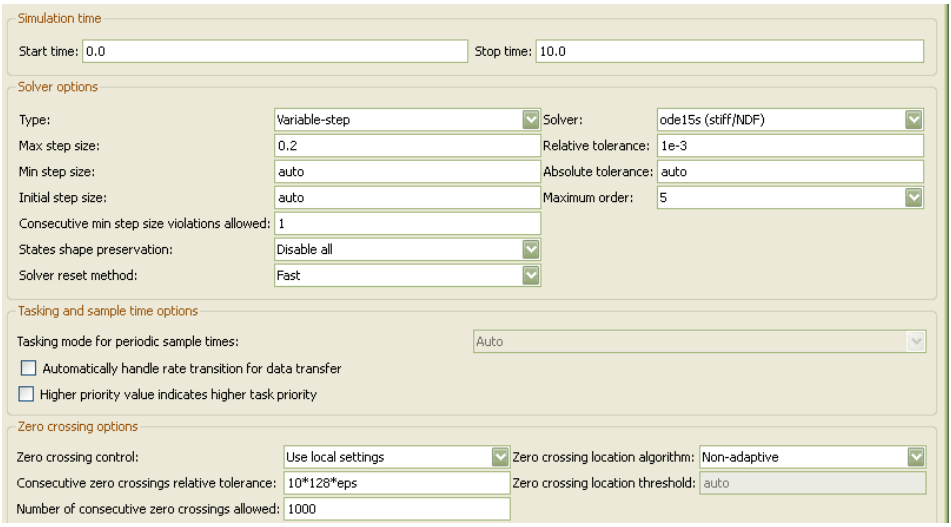


شکل(۸-۱۰) سیستم هیدرولیکی شبیه‌سازی شده

۱۳. مدل کامل شده را با نام hydraulic_model.mdl ذخیره کنید.

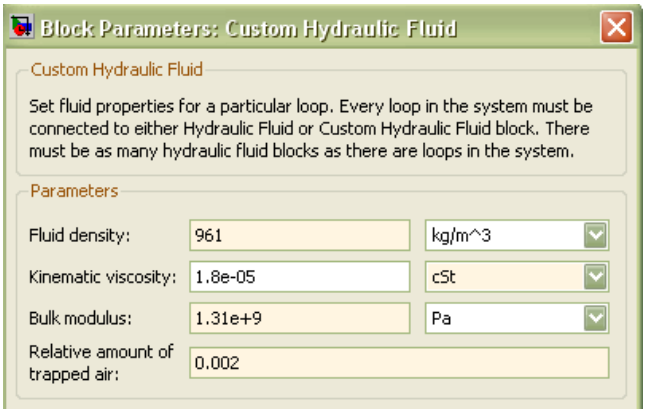
۱۴. بعد از اتصال بلوکها به‌هم، باید پارامترهای آنها را تنظیم کنید، سپس روش محاسبه را انتخاب کنید. برای این کار روی Simulink solver کلیک کرده و از منوی Simulation گزینه Configuration parameters را انتخاب کنید. در قسمت Solver options برای Solver گزینه ode15s را انتخاب کرده و Max step size را مساوی 0.2 قرار دهید.

[1] relative amount of trapped air



شکل(۸-۱۱) تنظیم پارامترهای حلگر

۱۵. روی بلوک Hydraulic Fluid دوبار کلیک کرده و در صفحه ظاهر شده پارامترها را مانند شکل(۸-۱۲) تنظیم کنید.



شکل(۸-۱۲) تنظیم بلوک Hydraulic Fluid

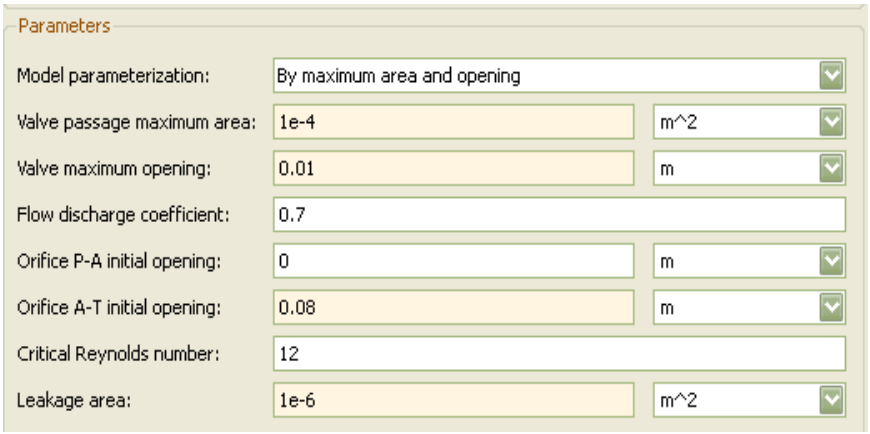
۱۶. روی Simulink – ps converter 1 کلیک کرده و در صفحه ظاهر شده، واحد آن را برابر Pa قرار دهید

۱۷. روی Constant block دو بارکلیک کرده و در صفحه ظاهر شده، مقدار آن را برابر 10e5 قرار دهید.

۱۸. روی بلوک 2- Position valve actuator دو بارکلیک کرده و در صفحه ظاهر شده مقدار Nominal signal value را برابر 24 تنظیم کنید.

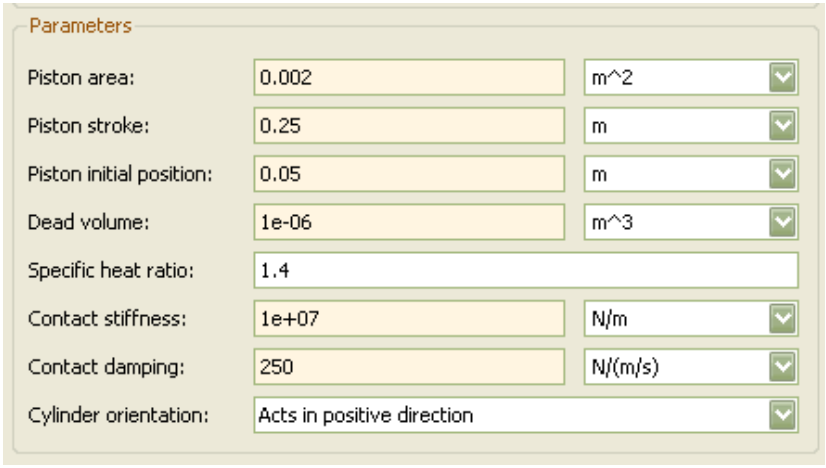
۱۹. روی بلوک Sin wave دو بارکلیک کرده و در صفحه ظاهر شده، مقدار Amplitude را بیش 50% ومقدار 2- Position valve actuator را برابر 20 تنظیم کنید .

۲۰. پارامترهای شیر سه راهه را مانند شکل (۸-۱۳) تنظیم کنید.



شکل(۸-۱۳) پارامترهای شیر سه راهه

۲۱. پارامترهای سیلندر را مانند شکل (۸-۱۴) تنظیم کنید.



شکل(۸-۱۴) پارامترهای سیلندر

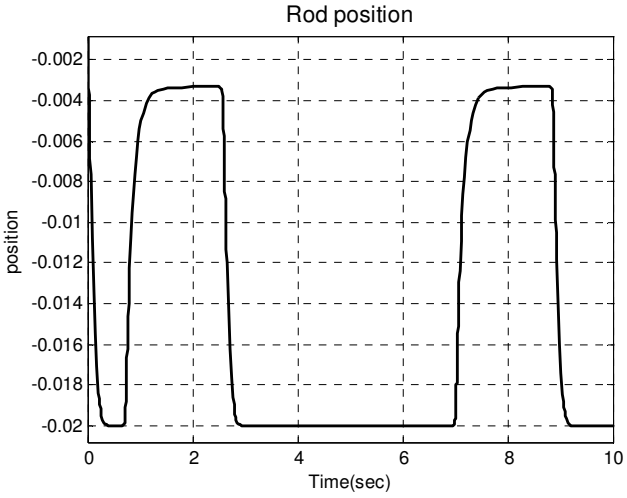
۲۲. مقدار جرم را برابر 4.5Kg قرار دهید.

۲۳. ضریب میرایی دمپر را برابر 250 N.s/m قرار دهید.

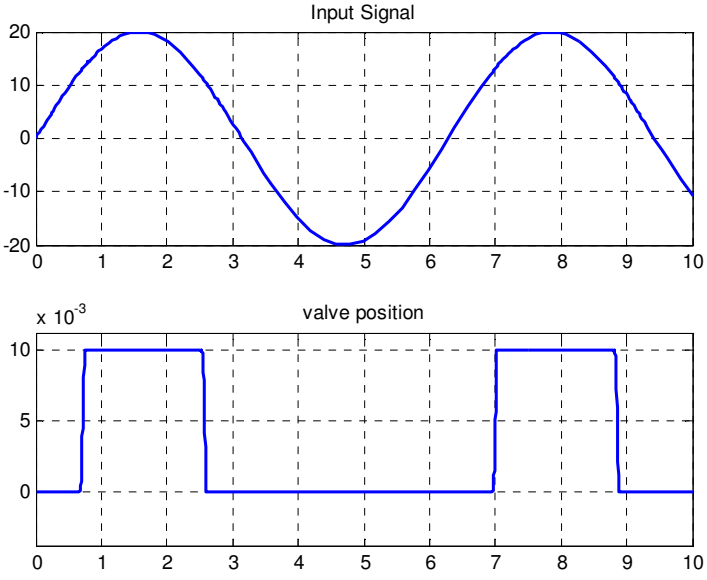
۲۴. ضریب فنر را برابر 6000 N/m و تغییر طول اولیه آن‌را برابر 0.02 m قرار داده، و مدل را ذخیره کنید.

۸–۳ شبیه سازی سیستم هیدرولیکی

روی Run کلیک کنید تا شبیه سازی شروع شود. سپس روی هر یک از اسکوپ‌ها کلیک کنید تا نتایج حاصل از شبیه سازی را مشاهده کنید. در این مدل Sine wave سیگنال لازم را برای راه اندازی شیر تامین کرده و اسکوپ سیلندر حرکات پیستون را نمایش می‌دهد. نتایج حاصل به صورت زیر است:



شکل(۸–۱۵) موقعیت دسته سیلندر



شکل(۸–۱۶) سیگنال ورودی و موقعیت شیر

۸-۴ بلوک‌های موجود در SimHydraulic

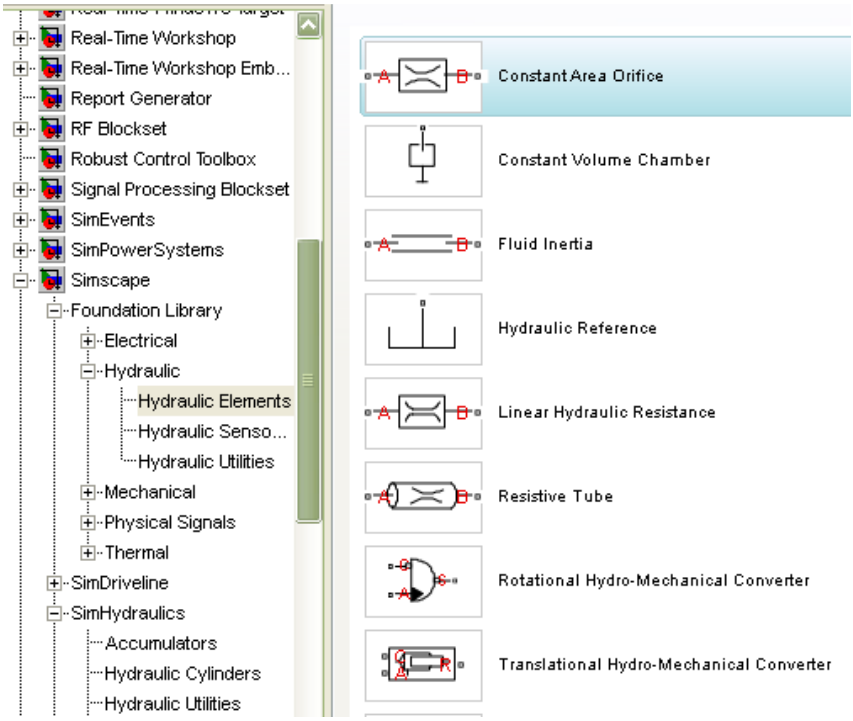
در جدول (۸-۱) فهرست کامل بلوک‌های موجود SimHydraulic و کاربرد هریک، آورده شده است.

جدول(۸-۱) فهرست بلوک‌های موجود در SimHydraulic

نام بلوک	توضیح
Accumulators • Gas-Charged Accumulator • Spring-Loaded Accumulator	شبیه سازی آکومولاتورهای هیدرولیکی همراه با گاز قابل تراکم و یا استفاده از فنر برای ذخیره سازی انرژی
Hydraulic Cylinders • Cylinder Friction • Double-Acting Hydraulic Cylinder • Double-Acting Rotary Actuator • Single-Acting Hydraulic Cylinder • Single-Acting Rotary Actuator	شبیه سازی اصطکاک درسیلندرهای هیدرولیکی، عملگرهای هیدرولیکی، عملگرهای چرخان دوجهته و تک جهته
Hydraulic Utilities • Hydraulic Fluid • Reservoir	شبیه سازی سیالات هیدرولیکی و تانک ذخیره سازی سیال
Local Hydraulic Resistances • Elbow • Gradual Area Change • Local Resistance • Pipe Bend • Sudden Area Change • T-junction	شبیه سازی مقاومت‌های هیدرولیکی در زانویی‌ها، لوله‌ها ، اتصالات T شکل، و تغییرات ناگهانی در قطر لوله‌ها
Orifices • Annular Orifice • Fixed Orifice • Orifice with Variable Area Round Holes • Orifice with Variable Area Slot • Variable Orifice	شبیه سازی اریفیس‌های متغیر هیدرولیکی با سطح مقطع ثابت و متغیر، برش مستطیلی و حلقه‌ای
Pipelines • Hydraulic Pipeline • Segmented Pipeline	شبیه سازی لوله‌ها با در نظر گرفتن مقاومت و خاصیت تراکم پذیری سیال
Pumps and Motors • Centrifugal Pump • Fixed-Displacement Pump • Hydraulic Motor • Variable-Displacement Motor	شبیه سازی موتورهای جابه‌جایی ثابت، جابه-جایی متغیر و پمپ‌های گریز از مرکز، جابه-جایی ثابت و متغیر

• Variable-Displacement • Pressure-Compensated Pump • Variable-Displacement Pump	
Directional Valves • 2-Way Directional Valve • 3-Way Directional Valve • 4-Way Directional Valve • Cartridge Valve Insert • Check Valve • Pilot-Operated • Check Valve • Shuttle Valve	شبیه سازی شیرهای کنترل جهت
Flow Control Valves • Ball Valve • Needle Valve • Poppet Valve • Pressure-Compensated Flow Control Valve	شبیه سازی شیرهای کنترل جریان
Pressure Control Valves • Pressure Compensator • Pressure Reducing Valve • Pressure Relief Valve	شبیه سازی شیرهای کنترل فشار
Valve Actuators • 2-Position Valve Actuator • 3-Position Valve Actuator • Hydraulic Cartridge Valve Actuator • Hydraulic Double-Acting Valve Actuator • Hydraulic Single-Acting Valve Actuator • Proportional and Servo-Valve Actuator	شبیه سازی عملگرها برای شیرهای دو و سه موقعیته، شیرهای هیدرولیکی تک و دو عملکردی و...
Valve Forces • Spool Orifice Hydraulic Force • Valve Hydraulic Force	شبیه سازی نیروهای هیدرولیکی محوری وارده به اسپول و نیروهای هیدرولیکی استاتیکی اعمال شده به شیر

۸-۵ المان‌های هیدرولیکی



شکل(۸-۱۷) المان‌های Hydraulic

۸-۶ شرح موردی بلوکهای المان‌های هیدرولیکی

Accumulator: دارای دو محفظه است که به وسیله دیافراگم یا پیستون از هم جدا شده اند. یکی از محفظه‌ها حاوی گاز تحت فشار و دیگری به سیستم هیدرولیکی متصل است. فرضیات در نظر گرفته شده عبارتند از:

۱. گاز از نظر ترمودینامیکی ایده آل فرض می‌شود.
۲. نوع فرایند پلی تروپیک است.
۳. از ممان اینرسی، اصطکاک، و ... دیافراگم چشم‌پوشی می‌شود.
۴. سیال تراکم ناپذیر فرض می‌شود .

در Spring –loaded accumulator به جای گاز از فنر استفاده شده است. فشار سیستم در فنر ذخیره شده، و در مواقع لازم آزاد می‌شود.

Constant area orifice: اوریفیس با سطح مقطع ثابت برای مدل کردن تغییر سطح مقطع ثابت در سیستمهای هیدرولیکی به‌کار می‌رود. جریان به دو نوع آرام و تربولانس تقسیم می‌شود. دبی سیال عبوری از

اوریفیس متناسب با اختلاف فشار دو طرف اوریفیس است. در اینجا از اینرسی سیال صرف نظر شده و حالت گذرای جریان (حالت بین آرام و تربولانس) در نظر گرفته نشده است.

Constant volume chamber: مخزن با حجم ثابت که دارای دیواره‌های صلب یا انعطاف پذیر می‌باشد و در مدل کردن قسمتهای مختلف هیدرولیکی مانند شیرها، پمپها، لوله‌ها ، شیلنگها، و ... استفاده می‌شود. در مواقعی که سیال تراکم پذیر است از این بلوک استفاده می‌شود. اگر اختلاف فشار مخزن افت کرده و منفی شود کاویتاسیون اتفاق می‌افتد. در حالت واقعی سیال هیدرولیکی با هوا مخلوط است و باید مدول بالک مرکب را محاسبه کرد.

Flow inertia: بلوکِ اینرسی سیال، برای مدل کردن اختلاف فشار از طریق تغییر دبی در عبور از یک سطح مقطع ثابت، استفاده می‌شود.

Linear hydraulic resistance: این بلوک برای مدل کردن مقاومت هیدرولیکی است، که در آن فشار نسبت به دبی افت می‌کند.

Resistive Tube: این بلوک برای مدل سازی خاصیت مقاومتی خطوط لوله با سطح مقطع دایره‌ای یا غیردایره‌ای استفاده می‌شود. بلوک با در نظر گرفتن اندازه حرکت در حالت پایا گسترش می‌یابد. ممان اینرسی و تراکم پذیری در نظر گرفته نشده است و خواصی مانند ضربه قوچ مدل نمی‌شود. برای محاسبه ممان اینرسی و تراکم پذیری می‌توان از بلوک‌های دیگر استفاده کرد. برای محاسبه مقاومتهای موضعی مانند خم، سه راهی، ورودی‌ها، و خروجی‌ها می‌توان آنها را به طول مقاومتی معادل تبدیل کرده و به طول کلی لوله اضافه کرد.

Cylinder Friction: این بلوک برای مدل کردن اصطکاک بین سیلندر و پیستون استفاده می‌شود و به عنوان بلوک اولیه قبل از بلوک سیلندر یک طرفه یا دو طرفه می‌آید. اصطکاک بر حسب تابعی از فشار و دبی شبیه سازی می‌شود و حاصل جمع نیروی اصطکاک Stibeck coulomb و ویسکوز می‌باشد. نیروی coulomb حاصل از آب بندی بوده و متناسب با فشار می‌باشد.

Double acting hydraulic cylinder: این بلوک برای مدل کردن سیلندر دو طرفه است و از نشتی سیلندر، ممان اینرسی، اصطکاک، و خاصیت فنری پیستون چشم‌پوشی می‌شود .

Double acting rotary actuator: عملگر دورانی دو طرفه برای تبدیل انرژی هیدرولیکی به انرژی دورانی مکانیکی استفاده می‌شود. سیال هیدرولیکی با فشار به داخل یکی از محفظه‌ها پمپ شده و شفت را می چرخاند. این عملگر دو جهته می‌باشد.

Single acting rotary actuator: عملگر دورانی یک طرفه برای تبدیل انرژی هیدرولیکی به انرژی دورانی مکانیکی استفاده می‌شود سیال هیدرولیکی با فشار به داخل یکی از محفظه‌ها پمپ شده و شفت را می چرخاند. این عملگر یک جهته می‌باشد.

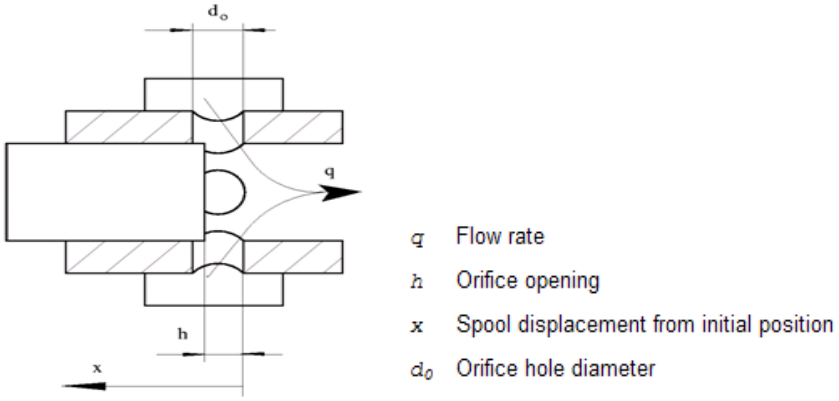
Elbow: این بلوک مقاومت موضعی زانویی را مدل میکند معمولاً زانویی‌ها دارای دو نوع یکی با گوشه ملایم و دیگری تیز می‌باشد. این بلوک دارای محدوده قطری بین ۵ تا ۱۰۰ میلیمتر و زاویه‌ای بین ۰ تا ۹۰ درجه را

پوشش می‌دهد. از اینرسی سیال صرف نظر می‌شود. سیال در حالت تربولانس در نظر گرفته می‌شود و همچنین جنس لوله فولاد تجاری فرض می‌شود.

Local resistance: این بلوک برای وارد کردن مقاومت موضعی مانند زانویی‌ها ، فیلترها، شیرها و تغییر سطح مقطع و... به‌کار می‌رود. افت فشار ناشی از اصطکاک به وسیله ضریب افت فشار که در کاتالوگ‌ها یا در جداولی داده شده است، تعیین می‌شود. این داده‌ها بر حسب عدد رینولدز مرتب می‌شوند.

Sudden area change: این بلوک برای مدل کردن تغییر ناگهانی در سطح مقطع می‌باشد. این تغییر می‌تواند در جهت افزایش یا کاهش سطح مقطع باشد. همچنین ضریب افت فشار را می‌توان به صورت جدولی از اعداد داد، که در این صورت باید ضریب افت فشار بر حسب عدد رینولدز باشد.

Orifice whit variable area: در این نوع اریفیس اسپول در داخل اسلیو قرار گرفته و روی اسلیو یک دسته از سوراخ‌ها قرار دارند. دبی اریفیس با میزان باز شدن آن رابطه خطی دارد. در شکل (۸-۱۸) نمای برش خورده اریفیس دیده می‌شود.



شکل(۸-۱۸) نمای برش خورده اریفیس

Hydraulic pipeline: این بلوک برای مدل کردن خطوط لوله با سطح مقطع دایره‌ای یا غیر دایره‌ای در سیستم‌های هیدولیکی بوده و برای محاسبه افت فشار و تراکم پذیری سیال در خطوط لوله از آن استفاده می شود. از اینرسی سیال صرف نظر شده است. در این بلوک ضربه قوچ پیش بینی نمی‌شود.

Centrifugal pump: این بلوک برای مدل کردن هر نوع پمپ گریز از مرکز می‌باشد. پارامترهای این پمپ ضرایب یک چند جمله‌ای می‌باشند که به صورت تجربی یا به وسیله آزمایش تعیین می‌شوند. مدل برای حالتی است که شفت راستگرد می‌چرخد .

Fixed displacement pump: این بلوک برای مدل کردن پمپ جابه‌جایی ثابت می‌باشد. پارامترهای کلیدی لازم برای این پمپ جابه‌جایی پمپ، سرعت زاویه‌ای پمپ، بازده حجمی و کلی و فشار نامی است که در کاتالوگ پمپ موجود می‌باشد.

Hydraulic motor: این بلوک برای مدل کردن موتور جابه‌جایی ثابت می‌باشد. پارامترهای کلیدی لازم برای این موتور جابه‌جایی موتور، سرعت زاویه‌ای موتور، بازده حجمی و کلی و فشار نامی است که در کاتالوگ موتور موجود می‌باشد.

Variable displacement pump: این بلوک برای مدل کردن پمپ جابه‌جایی متغیر با دور معکوس پذیر است. دبی پمپ متناسب با سیگنال ورودی از پورت c است. پارامترهای کلیدی لازم برای این پمپ، جابه‌جایی ماکزیمم پمپ، محدوده تنظیم، سرعت زاویه‌ای پمپ، بازده حجمی و کلی و فشار نامی است که در کاتالوگ پمپ موجود می‌باشد.

2-way directional valve: این بلوک برای مدل کردن دو راهه دو حالته استفاده می‌شود. شیر قطع سریع نیز به آن می‌توان گفت. این شیر دو پورت اتصال هیدرولیکی دارد: پورت ورودی P و پورت خروجی A و یک پورت سیگنال S که کنترل کننده موقعیت اسپول است. اساس شیر بر پایه اریفیس متغیر می‌باشد.

Check valve: این بلوک برای مدل کردن شیر یک طرفه می‌باشد. این شیر سیال را در یک جهت عبور داده، از برگشت سیال جلوگیری می‌کند. هرگاه اختلاف فشار دو طرف شیر از یک حدی زیادتر شد شیر باز می‌شود.

Pilot-operated check valve: این شیر مانند شیر یک طرفه است با این تفاوت که شیر در جهت مخالف با سیگنالی که از پورت x دریافت می‌کند باز می‌شود.

Flow control needle valve: این شیر برای کنترل دبی است و دارای اریفیس متغیر با سوپاپ مخروطی (سوزنی) می‌باشد.

Pressure compensator: این بلوک برای مدل کردن جبران کننده فشار می‌باشد. جبران کننده فشار، فشار مدار را در نقطه تنظیم شده ثابت نگه می‌دارد. شیر جبران کننده فشار به صورت ترکیبی با اریفیس متغیر در بالادست یا پایین دست استفاده می‌شود. شیر جبران کننده فشار در وضعیت عادی باز بوده و باز بودن آن متناسب با اختلاف فشار و نیروی فنر می‌باشد.

Pressure relief valve: این شیر برای مدل کردن شیر اطمینان می‌باشد. شیر در حالت عادی بسته است. هرگاه فشار ورودی شیر از مقدار تنظیمی تجاوز کند شیر باز می‌شود.

Hydraulic double acting valve actuator: این بلوک برای مدل کردن Hydraulic double acting valve actuator است که سیگنال فرمان را برای شیرهای چند وضعیتی، کنترل فشار، و کنترل دبی فراهم می‌کند. از تمام نیروها به غیر از نیروهای فنر و تراکم پذیری صرف نظر می‌شود. این المان از دو المان single acting valve actuator تشکیل شده است که بر ضد هم عمل می‌کنند هر یک از این المانها شامل یک پین، فنر، و واشر می‌باشند. هرگاه سیگنال فرمان به یکی از طرفین اعمال شود فنر متراکم شده و واشر حرکت کرده و اسپول از موقعیت میانی خود حرکت می‌کند.

فصل نهم

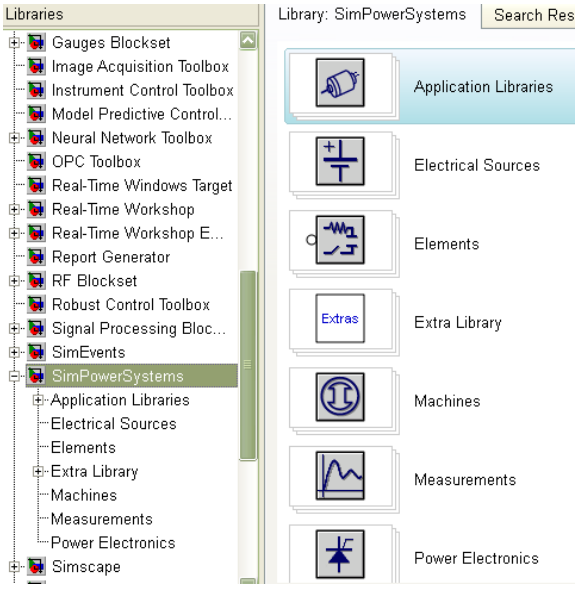
مدل‌سازی سیستم‌های الکتریکی در SimPowerSystems

۹– ۱ کتابخانه SimPowerSystems

از این بسته نرم افزاری برای مدل سازی و شبیه سازی مدارهای الکتریکی، که شامل المان‌های خطی و غیرخطی هستند، استفاده می‌شود. بلوک‌های موجود در کتابخانه SimPowerSystems عبارتند از:

۱. بلوک‌های منابع الکتریکی که سیگنال‌های الکتریکی تولید می‌کنند. به‌طور مثال منبع جریان سینوسی، منبع ولتاژ سینوسی، باتری، منبع جریان و ولتاژ کنترل شده، منبع ولتاژ DC، منبع ولتاژ سه فاز با قابلیت برنامه ریزی تغییرات دامنه، فاز، و فرکانس بر حسب زمان و ...

۲. بلوک‌های المان‌های مدارهای خطی و غیرخطی مثل سوئیچ قطع جریان، اتصال به زمین، ترانسفورماتور با دو یا سه و یا چند سیم پیچ، RLC موازی و سری تک فاز یا سه فاز، بارهای RLC موازی خطی و سری خطی به‌صورت تک فاز یا سه فاز و ...



شکل (۹–۱) بلوک‌های SimPowerSystem

۳. بلوک‌های قطعات الکترونیک قدرت مانند دیودها، مدل تریستور و GTO، سوئیچ ایده آل، MOSFET ، IGBT و ...

۴. بلوک‌های ماشین‌های الکتریکی مثل موتور آسنکرون سه فاز و تک فاز، موتور DC با راه انداز جداگانه، سیستم گاورنر PID، توربین هیدرولیکی، استپر موتور، موتورهای سنکرون مغناطیس دائم و ...

۵. بلوک‌هایی با کاربردهای متفاوت مثل درایورهای الکتریکی DC,AC ، کاهنده سرعت و ...

۶. بلوک‌های اندازه گیری ولتاژ و جریان و امپدانس

۷. کتابخانه Extras که شامل چندین بلوک متفاوت دیگر مثل اندازه‌گیری توان اکتیو و راکتیو، ژنراتور PWM ، ژنراتور پالس سنکرون، اجرا کننده آنالیز فوریه بر روی سیگنال و...

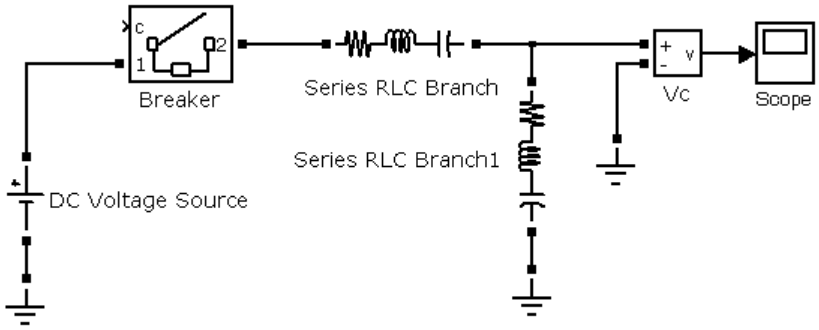
۹– ۲ بلوک‌های اندازه‌گیری ولتاژ، زمین، منبع DC، RLC سری، و Breaker

مثال: آشنایی با بلوک‌های اندازه‌گیری ولتاژ، زمین، منبع DC، RLC سری، و Breaker

جهت ایجاد مدل، یک پنجره جدید باز کنید و بلوک‌های زیر را از کتابخانه سیمولینک وارد پنجره مدل کنید. سپس مطابق شکل(۹-۲) آنها را مرتب کرده، و به‌هم متصل نمائید.

جدول(۹–۱) مسیر بلوک‌های مورد استفاده در این مثال

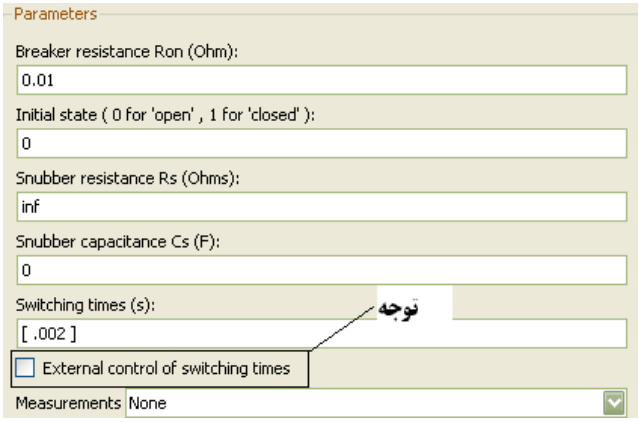
تعداد	مسیر بلوک
1	Simulink >>Sinks>>Scope
1	SimPowerSystems>> Electrical Sources>>DC voltage Source
2	SimPowerSystems>> Elements>>Series RLC Branch
1	SimPowerSystems>> Elements>>Breaker
1	SimPowerSystems>> Measurements>>Voltage Measurement
3	SimPowerSystems>> Elements>>Ground



شکل (۹–۲) نحوه اتصال بلوک‌ها به‌یکدیگر

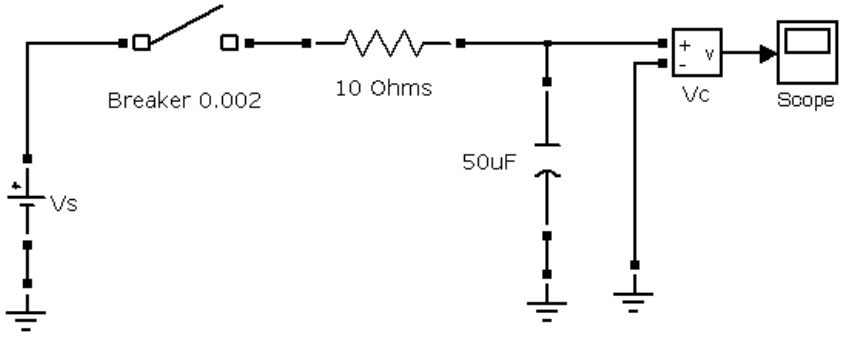
تنظیمات بلوک‌ها

- روی بلوک DC Voltage Source دوبار کلیک کرده و در پنجره ظاهر شده مقدار ولتاژ را برابر ۱۰۰ ولت وارد کنید. نام المان را به Vs تغییر دهید.
- روی Breaker دوبار کلیک کرده و در پنجره ظاهر شده تنظیمات زیر را انجام دهید. مقدار مقاومت Breaker برابر 0.01 اهم و مقاومت Rs بینهایت (inf) است. از طرفی گزینه External control of switching times را غیرفعال کنید. مقدار switching times را روی 0.002 ثانیه تنظیم کنید.



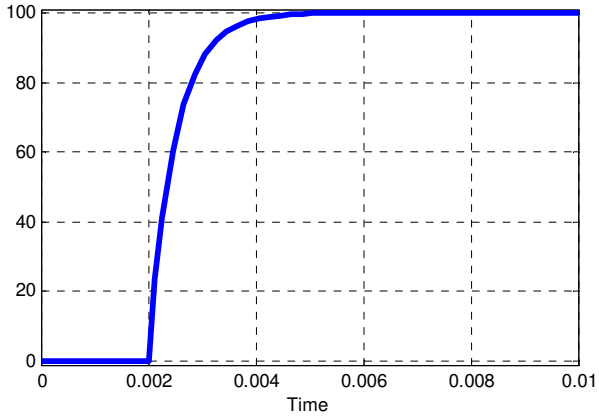
شکل (۹-۳) تنظیمات Breaker

- روی Series RLC Branch دوبار کلیک کرده و در پنجره ظاهر شده Branch type را R انتخاب کرده و مقدار آن را ۱۰ اهم وارد کنید و نام المان را به 10Ohms تغییر دهید.
- روی Series RLC Branch1 دوبار کلیک کرده و در پنجره ظاهر شده Branch type را C انتخاب کرده و مقدار آن را ۵۰ میکروفاراد وارد کنید. سپس نام المان را به 50uF تغییر دهید.



شکل (۹-۴) نحوه اتصال بلوک‌ها به‌یکدیگر

مدت زمان شبیه سازی را برابر 0.01 ثانیه قرار داده و روی start simulation کلیک کنید تا خروجی را در scope مشاهده کنید.



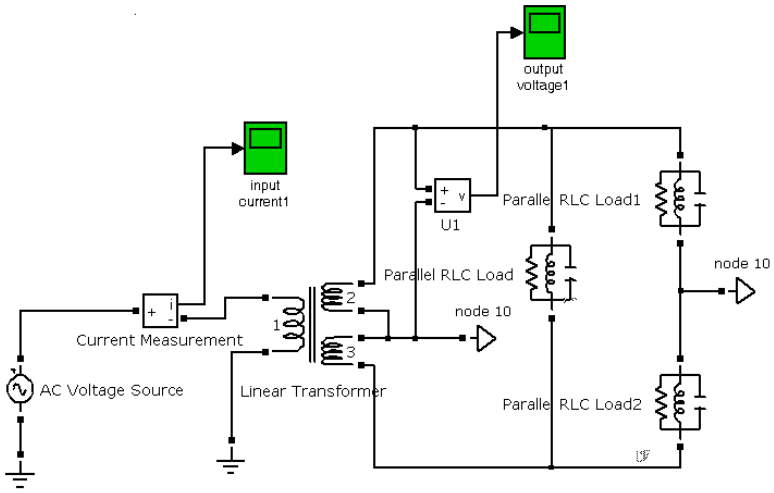
شکل (۹-۵) ولتاژ خروجی

۹- ۳ بلوک‌های ترانسفورماتور خطی، منبع ولتاژ AC، بار RLC موازی، و اندازه-گیری جریان

مثال: آشنایی با بلوک‌های ترانسفورماتور خطی، منبع ولتاژ AC، بار RLC موازی و اندازه‌گیری جریان جهت ایجاد مدل، یک پنجره جدید باز کنید و بلوک‌های زیر را از کتابخانه سیمولینک وارد پنجره مدل کنید. سپس مطابق شکل (۹-۶) آنها را مرتب نموده و به‌هم متصل کنید.

جدول(۹-۲) مسیر بلوک‌های مورد استفاده در این مثال

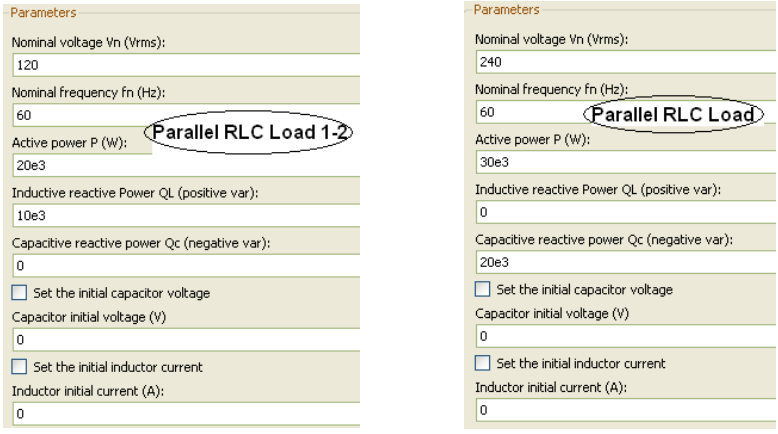
تعداد	مسیر بلوک
2	Simulink >>Sinks>>Scope
1	SimPowerSystems>> Electrical Sources>>AC voltage Source
3	SimPowerSystems>> Elements>>Parallel RLC Load
1	SimPowerSystems>> Elements>>Linear transformer
1	SimPowerSystems>> Measurements>>Voltage Measurement
1	SimPowerSystems>> Measurements>>Current Measurement
2	SimPowerSystems>> Elements>>Ground
2	SimPowerSystems>> Elements>>Neutral



شکل (۹-۶) نحوه اتصال بلوک‌ها به‌یکدیگر

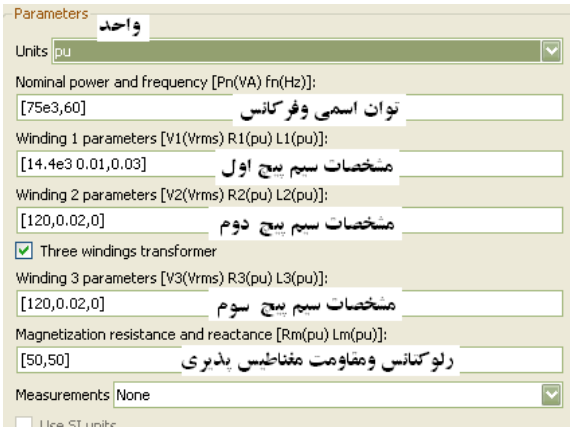
تنظیمات بلوک‌ها

- روی بلوک AC Voltage Source دوبار کلیک کرده و در پنجره ظاهر شده مقدار پیک ولتاژ رابرابر 14400*sqrt(2) ولت وارد کنید. سپس نام المان را به 14400V,60Hz تغییر دهید.
- روی Parallel RLC Branch دوبارکلیک کرده و در پنجره ظاهر شده مقادیر زیر را مطابق شکل وارد کنید و نام المان را به 240V,30KW تغییر دهید. ولتاژ نامی در این المان برابر 240V و فرکانس نامی برابر 60Hz می‌باشد. توان اکتیو نیز برابر 20000W است.
- روی Parallel RLC Branch1, 2 دوبارکلیک کرده و در پنجره ظاهر شده مقادیر زیر را مطابق شکل (۹-۷) وارد کنید و نام المان را به 120V,20KW تغییر دهید. ولتاژ نامی در این المان برابر 120V و فرکانس نامی برابر 60Hz می‌باشد. توان اکتیو نیز برابر 20000W است.



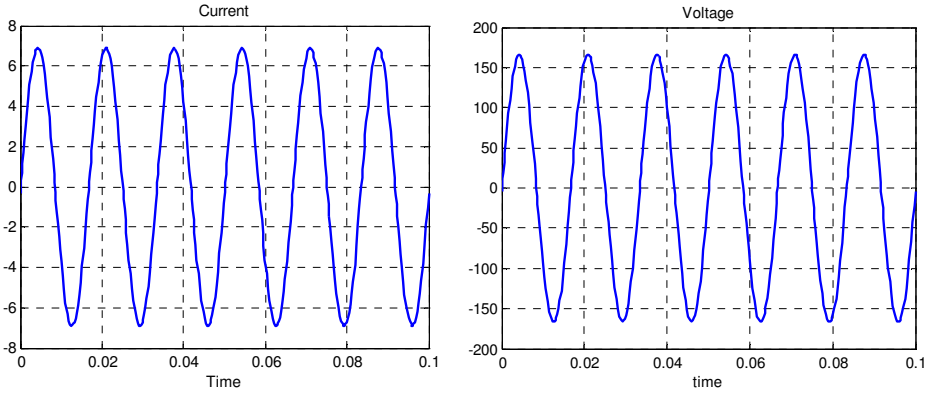
شکل (۹-۷) تنظیمات Parallel RLC Branch1, 2

- روی هر دو بلوک Node کلیک کرده و شماره Nodeها را 50 قرار دهید. این بلوک برای ایجاد نقطه مشترک دو نقطه از مدار و یا اتصال دو نقطه، که نمی‌خواهیم بین آنها خط ارتباطی بکشیم، استفاده می شود.
- روی Linear Transformer دوبارکلیک کرده و در پنجره ظاهر شده مقادیر زیر را وارد کنید. گزینه Three windings for Transformer را فعال کنید.



شکل (۹-۸) تنظیمات ترانسفورماتور خطی

مدت زمان شبیه سازی را برابر 0.1 ثانیه قرار داده و روی start simulation کلیک کنید تا خروجی را در scopeها مشاهده کنید.



شکل (۹-۹) ولتاژ و جریان خروجی

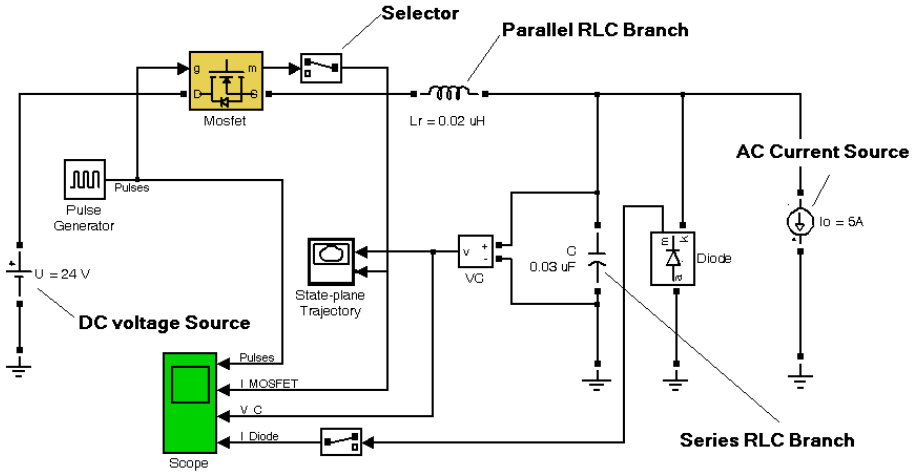
۹- ۴ بلوک‌های الکترونیک قدرت MOSFET, Diode

مثال: آشنایی با بلوک‌های MOSFET, Diode و سلکتور

جهت ایجاد مدل یک پنجره جدید باز کنید و بلوک‌های زیر را از کتابخانه سیمولینک وارد پنجره مدل کنید. سپس مطابق شکل(۹-۱۰) آنها را مرتب نموده و به‌یکدیگر متصل کرده و نام آنها را تغییر دهید.

جدول(۹-۳) مسیر بلوک‌های مورد استفاده در این مثال

تعداد	مسیر بلوک
1	Simulink >>Sinks>>Scope
1	Simulink >>Sinks>>XY Graph
2	Simulink >>Signal Routing>>Selector
1	SimPowerSystems>> Electrical Sources>>AC Current Source
1	SimPowerSystems>> Electrical Sources>> DC voltage Source
1	SimPowerSystems>> Elements>>Parallel RLC Branch
1	SimPowerSystems>> Elements>>Series RLC Branch
1	SimPowerSystems>> Measurements>>Voltage Measurement
4	SimPowerSystems>> Elements>>Ground
1	SimPowerSystems>> Power Electronics>>Diode
1	SimPowerSystems>> Power Electronics>>MOSFET

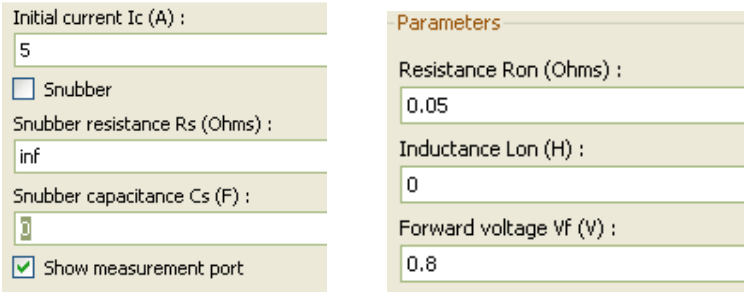


شکل (۹-۱۰) نحوه اتصال بلوک‌ها به‌یکدیگر

تنظیمات بلوک‌ها

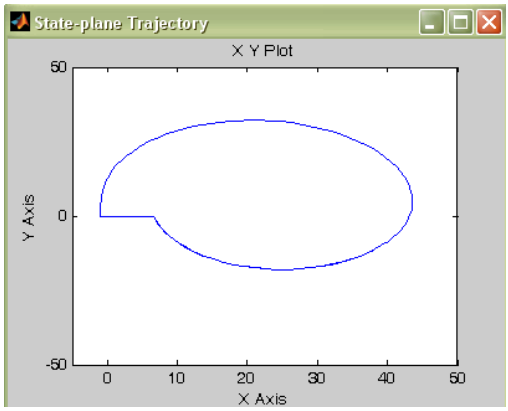
- روی بلوک AC Current Source دوبار کلیک کرده و در پنجره ظاهر شده مقدار پیک جریان را برابر ۵ آمپر، فاز ۹۰ درجه، و فرکانس 0.01Hz وارد کنید.

- روی بلوک DC Voltage Source دوبار کلیک کرده و در پنجره ظاهر شده مقدار پیک ولتاژ را ۲۴ ولت وارد کنید.
- روی بلوک Parallel RLC Branch دوبارکلیک کرده و در پنجره ظاهر شده Branch type را L انتخاب کرده و مقدار آن را 0.02e-6 هانری وارد کنید.
- روی بلوک Series RLC Branch دوبارکلیک کرده و در پنجره ظاهر شده Branch type را C انتخاب کرده و مقدار آن را 0.03e-6 فاراد وارد کنید.
- روی بلوک Pulse Generator دوبار کلیک کرده و مقدار period را 0.5e-6sec ، مقدار duty Cycle را 20% ، دامنه را ۱ و زمان شروع را 0.25e-6 وارد کنید.
- روی بلوک Scope دوبار کلیک کرده و درپنجره ظاهر شده روی Parameters در نوار ابزار کلیک کرده و سپس در صفحه ظاهر شده مقدار number of axis را ۴ وارد کنید.
- روی بلوک‌های Selector دوبارکلیک کرده و مقدار Number of input dimensions را برابر۱ و در قسمت index option گزینه one-based را انتخاب کرده و مقدار index را برابر ۱ قرار دهید. مقدار input port size را برابر ۲ تنظیم کنید.
- روی بلوک Mosfet دوبار کلیک کرده و در پنجره ظاهر شده مقدار internal diode resistance را برابر 0.01 اهم و مقدار internal diode forward voltage رابرابر 0.8 ولت قرار دهید.
- روی بلوک diode دوبار کلیک کرده و در پنجره ظاهر شده مقادیر زیر را وارد کنید:

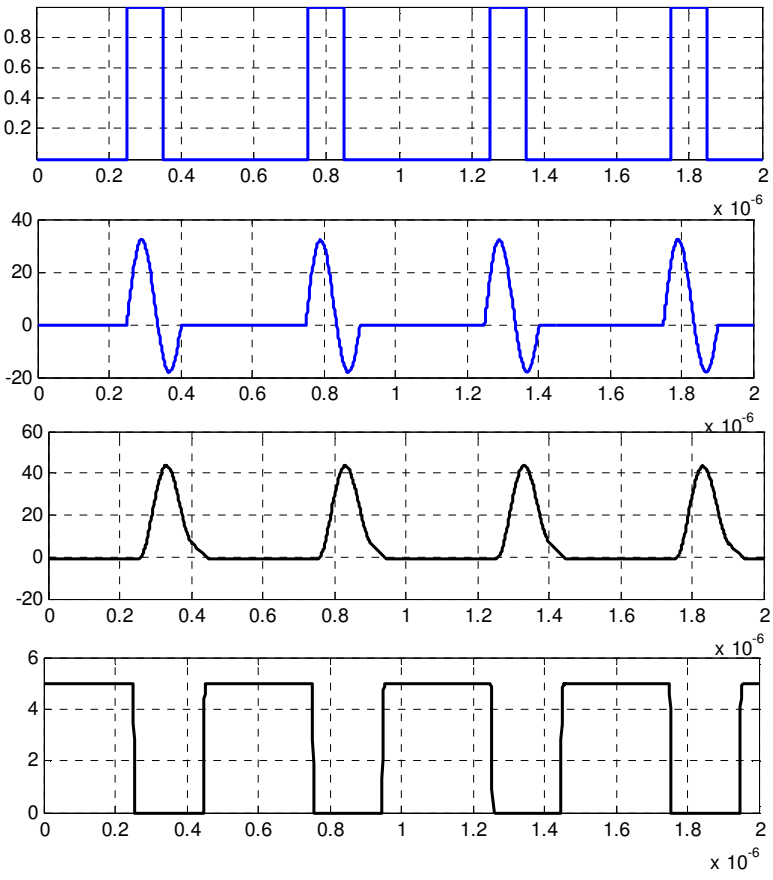


شکل (۹-۱۱) تنظیمات دیود

مدت زمان شبیه سازی را برابر 2e-6 ثانیه قرار داده و روی start simulation کلیک کنید تا خروجی را در scope و XYGraph مشاهده کنید.



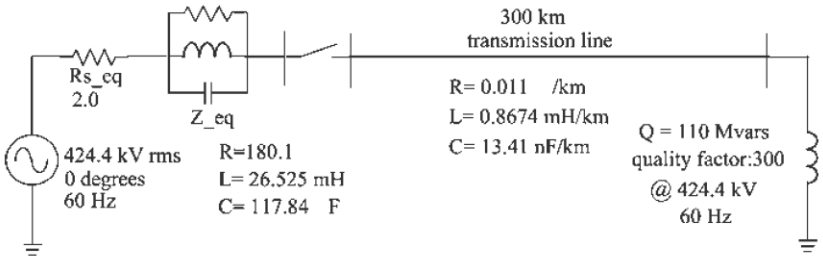
شکل (۹-۱۲) XYGraph



شکل (۹-۱۳) خروجی Scope، شبیه‌سازی سیستم الکترونیکی

۹– ۵ خط انتقال قدرت

مثال۴: مدار زیر یک سیستم قدرت توان را که تغذیه کنندهٔ یک خط انتقال ۳۰۰ کیلومتری است، نشان می‌دهد. خط در طرف تغذیه با القاگر شنت جبران شده است. یک Breaker اجازه باردار و بی‌بار کردن خط را می‌دهد. برای ساده کردن بحث، فقط یکی از سه فاز نمایش داده شده است.

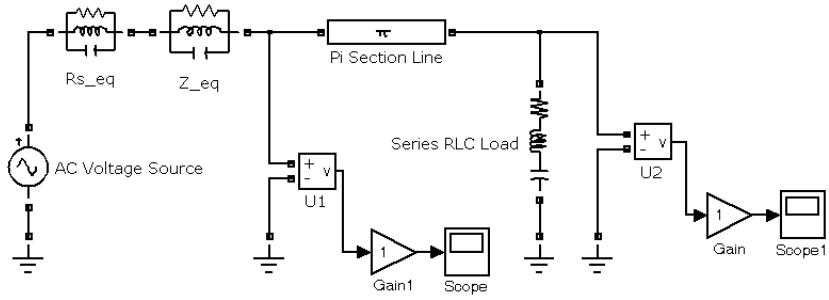


شکل (۹–۱۴) خط انتقال قدرت

یک صفحه جدید از منوی File باز نمائید و آن را با نام circuit ذخیره کنید. سپس از کتابخانه SimPoweSystems بلوک‌های زیر را وارد پنجره مدل سازی کرده و مطابق شکل(۹–۱۵) مرتب نموده، و تغییر نام دهید.

جدول(۹–۴) مسیر بلوک‌های مورد استفاده در این مثال

تعداد	مسیر بلوک
1	SimPowerSystems>> Electronic Sources>>AC voltage Source
4	SimPowerSystems>> Elements>>Parallel RLC Branch
1	SimPowerSystems>> Elements>>Pi section Line
4	SimPowerSystems>> Elements>>Ground
2	SimPowerSystems>> Measurements>>Voltage Measurement
2	Simulink>>Math Operation>> Gain
2	Simulink>>Sinks>> Scope



شکل (۹–۱۵) نحوه اتصال بلوک‌های خط انتقال قدرت

تنظیمات زیر را روی بلوک‌ها انجام دهید:

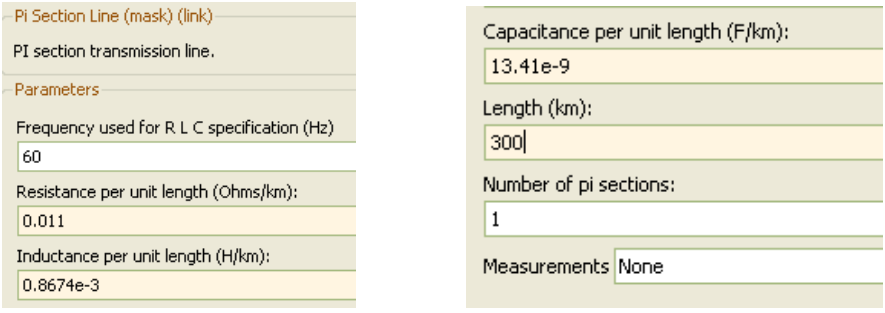
- اسم منبع ولتاژ را به Vs تغییر دهید. سپس با دو بار کلیک کردن روی منبع ولتاژ AC پنجره تنظیمات آن‌را باز نموده و پارامترها را به‌صورت زیر تنظیم کنید:

Peak amplitude=424.4e3*sqrt(2)
Phase=0 deg
Frequency=60Hz

- پارامترهای RLC آن‌را همانگونه که در مدار نشان داده شده است تغییر دهید و اسم آن را Z_eq بگذارید.

Branch type=RLC
Resistance(R) =180.1 ohm
L=26.425e-3H
C=117.84F

- مقاومت Rs_eq را می‌توان از شاخه موازی RLC درست کرد. روی RLC دوبار کلیک کرده و در پنجره ظاهر شده Branch type را R انتخاب کنید. سپس مقدار مقاومت را طبق مدار برابر ۲ اهم قرار دهید. وقتی پنجره تنظیمات را می‌بندید متوجه می‌شویدکه قطعات L,C نشان داده نمی‌شوند و آیکون در حال حاضر یک مقاومت ساده را نمایش می‌دهد. برای قرار دادن المان مقاومت می‌توان در المان RLC موازی، مقدار پارامتر Inductance L (H) را برابر بینهایت و پارامتر Capacitance C(F) را برابر صفر قرار داد.
- مدل یک خط با توزیع یکنواخت پارامترهای C,L,R در طول خط، شامل یک تأخیر زمانی است که مساوی با زمان انتشار موج در طول خط است. این مدل نمی‌تواند به عنوان یک سیستم خطی شبیه سازی گردد، زیرا یک تأخیر، برابر یک سری بینهایت از حالت‌ها است. بنابراین یک تقریب خوب از یک خط با یک سری از حالات پایدار، با سری کردن چندین مدار PI که هر کدام به عنوان بخش کوچکی از خط ظاهر می‌شود قابل دست یابی است. بخش PI شامل یک شاخه سری RL و دو شاخه شنت PI می‌باشد. دقت مدل به تعداد بخش‌های PI که برای مدل استفاده شده است، بستگی دارد. تنظیمات زیر را طبق مدار روی Pi انجام دهید:

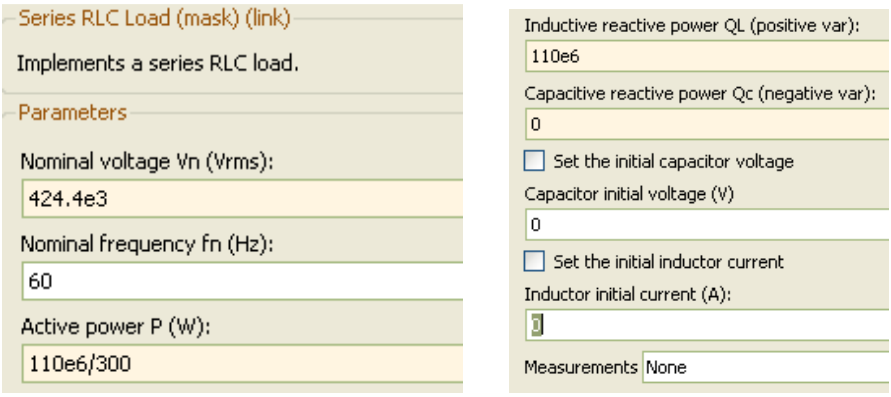


شکل (۹-۱۶) تنظیمات PI

- راکتور شنت با یک مقاومت و القاگر سری مدل شده است. می‌توان از شاخه سری RLC (Series RLC Branch) برای مدل کردن راکتور شنت استفاده نمود، اما باید مقادیر R,L را به صورت دستی ازتوان اکتیو و راکتیو که در مدار نشان داده شده است، محاسبه کرد.

P=110e6/300=0.37MW, Q=110Mvars, V=424.4kV, f=60Hz

بنابراین برای راحتی کار از RLC (Series RLC Load) استفاده می‌نمائیم که به ما این اجازه را می‌دهد که مستقیماً توان اکتیو و راکتیو شنت را تعیین نماییم. نام این بلوک را به 110Mvar تغییر دهید و پارامترها را مطابق شکل(۹-۱۷) تنظیم کنید.

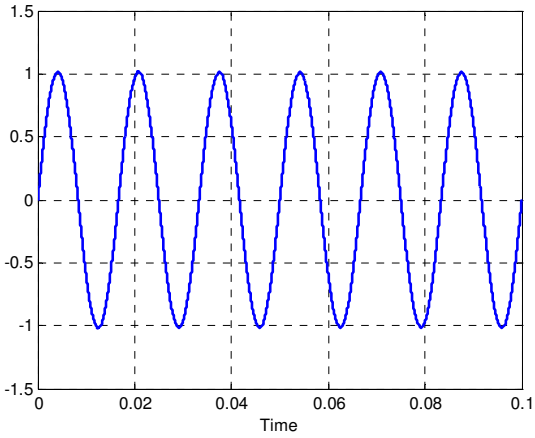


شکل (۹-۱۷) تنظیمات Series RLC Load

توجه کنید وقتی توان راکتیو خازنی تعیین نشده است، با بسته شدن پنجره تنظیمات، خازن درآیکون قطعه نمایش داده نمی‌شود.

- نام بلوک Voltage Measurement را U1 بنامید. برای مشاهده ولتاژ اندازه‌گیری شده توسط قطعه اندازه‌گیر ولتاژ که U1 نامیده شده یک سیستم نمایش مورد نیاز است. این سیستم می‌تواند هر وسیله‌ای که در کتابخانه Sinks از Simulink یافت می‌شود، باشد. اگر اسکوپ مستقیماً به خروجی اندازه‌گیر ولتاژ وصل شود ولتاژ را به ولت نمایش می‌دهد. اما، مهندسان قدرت در سیستم‌های قدرت با مقادیر نرمال شده کار می‌کنند. ولتاژ با تقسیم به مقدار پیک ولتاژ نامی سیستم به عنوان مقدار مبنا نرمالیزه می‌شود. بنابراین مقدار بلوک Gain را برابر $k=\frac{1}{424.4\times10^{3}\times\sqrt{2}}$ وارد نمائید. خروجی گین را به بلوک اسکوپ و خروجی قطعه اندازه گیر ولتاژ را به ورودی بلوک گین اتصال دهید.

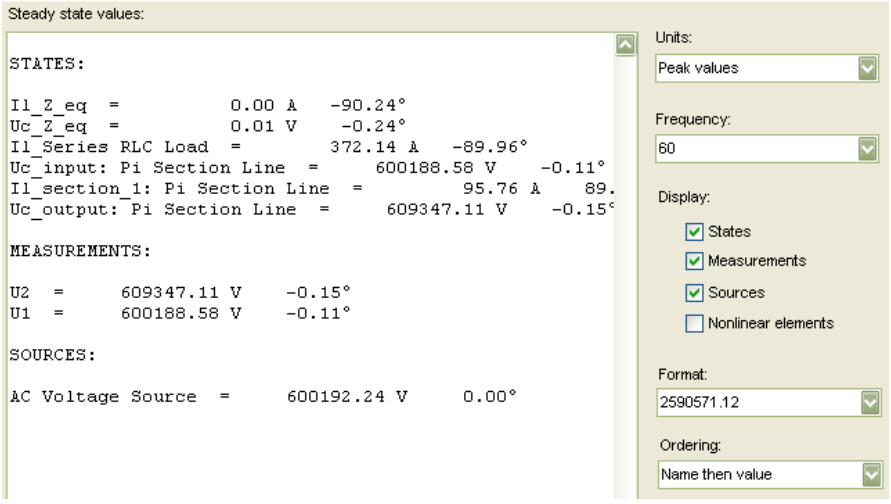
مدت زمان شبیه سازی را برابر 0.1 ثانیه قرار داده و روی start simulation کلیک کنید تا خروجی را در scopeها مشاهده کنید.



شکل (۹-۱۸) نمایش ولتاژهای خروجی

تحلیل حالت پایدار

برای آسان کردن تحلیل حالت پایدار، یک نمایشگر گرافیکی (powergui) در کتابخانه powerlib در نظر گرفته شده است. بلوک را به پنجره circuit کپی نمائید و روی آن دو بار کلیک کنید تا صفحه آن باز شود. پنجره حالت پایدار را با کلیک روی دکمه Steady State Voltages and Currents باز نمائید. در این پنجره فازورهای اندازه‌گیری شده توسط دو قطعه اندازه‌گیری به فرم قطبی نشان داده شده است.



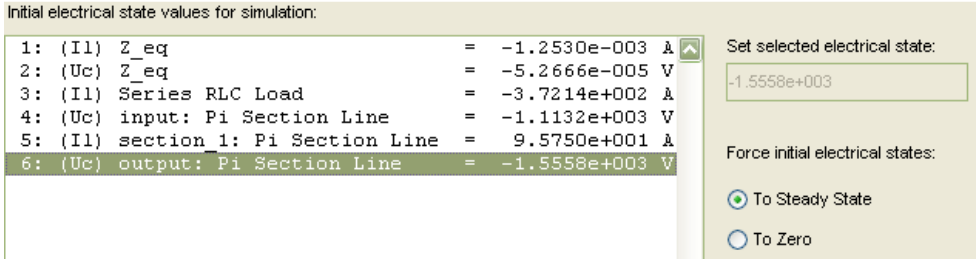
شکل (۹-۱۹) نمایشگر گرافیکی(powergui)

خروجی هر اندازه‌گیر با تطبیق آن با اسم قطعه اندازه‌گیر مشخص می‌شود. اندازه فازورهای U1,U2 با مقدار پیک ولتاژ سینوسی تطبیق یافته اند. همچنین می‌توان در پنجره حالت پایدار مقدار حالت پایدار منبع ولتاژ یا

مقدار حالت پایدار شرایط را با تیک زدن چک باکس‌هایSources یا States نمایش داد. اسم متغیرهای حالت شامل اسم قطعه‌ای می‌باشد که در آن القاگر یا خازن وجود دارد.

قرارداد علامت منابع ولتاژ و جریان و متغیرهای حالت بر مبنای وضعیت در مدار است. به این صورت که جریان القاگرها اگر در جهت فلش جاری باشد مثبت است و ولتاژ خازن‌ها عبارت است از ولتاژ خروجی قطعه منهای ولتاژ ورودی قطعه.

اکنون از جعبه ابزار powergui روی دکمه تنظیم مقادیر اولیه (Initial States Setting) کلیک نمائید. مقادیر اولیه شش شرط مدار (جریان سه القاگر و ولتاژ سه خازن) نشان داده می‌شوند. این مقادیر اولیه برای شروع شبیه سازی حالت پایدار تنظیم شده اند.



شکل (۹-۲۰) مقادیر حالت‌های الکتریکی اولیه برای شبیه‌سازی

فصل دهم

GUI(Graphical User Interface)

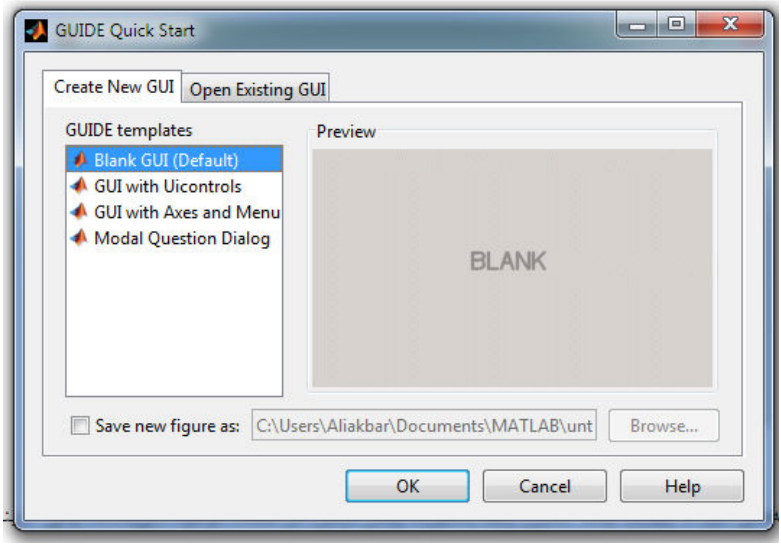
۱۰-۱ GUI چیست؟

GUI محیط گرافیکی است که با استفاده از آن می‌توان برنامه‌ای ایجاد کرد، که بتواند بین کاربر و برنامه کامپیوتری ارتباط برقرار کند، به‌طوری‌که کاربر فقط با استفاده از صفحه‌کلید و ماوس اطلاعات مورد نیاز را وارد کرده، برنامه اجرا شود. برای وارد شدن به فضایGUI می‌توان از دو روش استفاده نمود:

- مسیر File>>New>>GUI را دنبال کنید.
- دستور guide را در پنجره دستورات MATLAB وارد کنید.

در این هنگام پنجره Guide Quick Start باز می‌شود. این پنجره شامل گزینه‌های زیر می‌باشد:

- Blank GUI
- GUI with UiControl
- GUI with Axes an Menu
- Model Question Dialog



شکل (۱۰-۱) پنجره Guide Quick Start

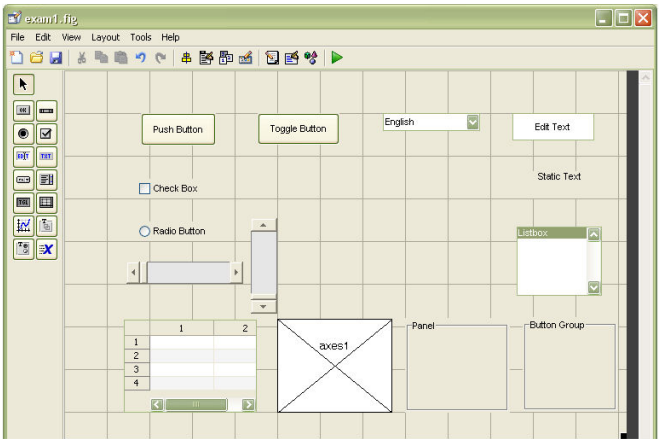
درگزینه‌های دوم تا چهارم، چند کنترل رابط به‌صورت پیش فرض، در صفحه موجود می‌باشد، ولی گزینه Blank GUI شامل یک فضای خالی است. این گزینه را انتخاب کرده و روی Ok کلیک کنید تا پنجره طراحی ظاهر شود.

۱۰– ۲ کنترل‌های رابط

کنترل‌های رابط MATLAB در سمت چپ صفحه طراحی قرار دارند که در ادامه هریک توضیح داده می‌شوند:

- **Push Button**: دکمه مستطیل شکلی است که با کلیک کردن روی این دکمه دستوراتی که برای کلید توسط کاربر تعریف شده‌اند، اجرا می‌شوند.
- **Slider**: با جابه‌جا کردن لغزنده کنترل و قرار دادن آن در مکان خاصی از طول Slide مقداری از بازه [min-max] انتخاب می‌شود.
- **Radio Button**: با انتخاب این گزینه دایره مزبور سیاه رنگ شده و ارزش کلیدهای Radio برابر مقدار Max می‌شود.
- **Check Box**: زمانی که مربع کوچک علامت می‌خورد کنترل فعال شده و خاصیت Value آن برابر ۱ می‌گردد. در حالت غیرفعال بودن مقدار آن برابر صفر است.
- **Edit Text**: متن یا عددی را نمایش می‌دهد که می‌توان آن را تغییر داد. این متن از طریق ویژگی String مربوط به کنترل قابل دسترسی است.
- **Static Text**: کنترلی است که متن یا مقدار عددی را به صورت یک نوشته نمایش می‌دهد و بیشتر به عنوان برچسب برای مقادیر جاری استفاده می‌شود.
- **Pop-Up-Menu**: با کلیک کردن روی فلش سمت راست این کنترل، منویی شامل انتخاب‌هایی که کاربر در خاصیت String کنترل وارد نموده است باز می‌شود.
- **List Box**: یک کنترل گرافیکی است برای نمایش چند رشته که می‌توان با کلیک‌کردن روی رشته‌ها، آنها را انتخاب کرد.
- **Toggle Button**: وقتی‌که این دکمه مستطیل شکل فشار داده می‌شود، پایین رفته و مقدار Max در Value قرار داده می‌شود. ولی اگر فشار داده نشود مقدار Min در Value قرار می‌گیرد.
- **Axes**: یک کنترل گرافیکی برای نمایش نمودارهای رسم شده می‌باشد.
- **Panel**: کنترل مستطیل شکلی است برای تفکیک دکمه‌ها و متن‌ها و نظم بخشیدن به صفحه.
- **Button Group**: همانند Panel است ولی با این تفاوت که در این صفحه در هر لحظه یک کلید می‌تواند فعال باشد.
- **ActiveX Control**: این کنترل برای استفاده از امکانات و برنامه‌های مختلف Windows به کار گرفته می‌شود.

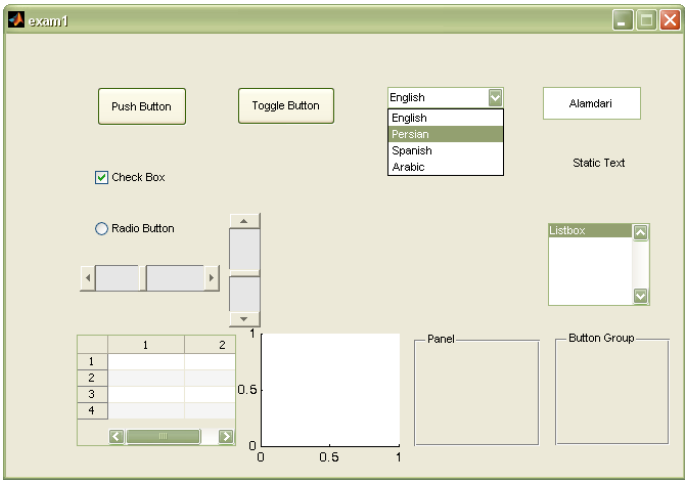
در شکل (۱۰-۲) کنترل‌های رابط را مشاهده می‌کنید.



شکل (۱۰-۲) انواع کنترل‌های رابط

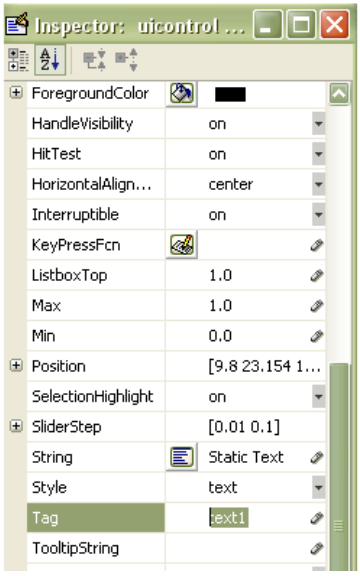
در گوشه سمت راست و پایین صفحه مربع سیاه رنگی مشاهده می‌شود که با جابه‌جا کردن توسط ماوس می‌توان اندازه صفحه را تغییر داد. با کلیک راست کردن روی صفحه خالی و انتخاب GUI Option صفحه‌ای با همین نام ظاهر می‌شود. در قسمت Resize behavior اگر گزینه Non-Resizable انتخاب شود با تغییر اندازه صفحه طراحی مکان و اندازه کنترل‌ها ثابت باقی می‌ماند ولی با انتخاب گزینه proportional این مقادیر تغییر می‌کنند.

پس از طراحی و ساخت پنجره با فشار دادن Run Figure همزمان با پنجره نهایی زیر پنجره M-file Editor نیز باز می‌شود. در صورت بسته شدن، در بالای صفحه روی دکمه M-file Editor کلیک کنید تا دوباره باز شود.



شکل (۱۰-۳) حالت اجراشدهٔ کنترل‌های رابط

کنترل‌های رابط دارای خصوصیاتی هستند که قسمتی از آنها قابل تغییر و تنظیم می‌باشند. آن دسته از ویژگی‌های کنترلرهای گرافیکی که قابل تغییر هستند در پنجره Property Inspector (منوی بالای صفحه) کنترل وجود دارند. با دوبار کلیک کردن روی هر یک از رابط‌های کنترلی، این پنجره ظاهر می‌شود.



شکل (۱۰-۴) پنجره Property Inspector

۱۰- ۳ آشنایی با بلوک‌های static text، Edit text، Push Button، pop-up menu، Radial Button...

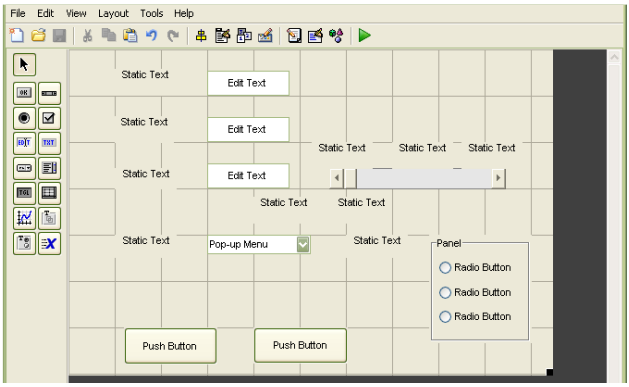
مثال: پنجره‌ای مشابه شکل (۱۰-۵) با کنترل‌های موجود بسازید که قابلیت‌های زیر را داشته باشد:

- بتوان با استفاده از منوهای بالای صفحه یک فایل جدید برای مشخصات هر فرد ایجاد کرد.
- مشخصات فرد، به‌عنوان مثال نام، نام خانوادگی، معدل، کشور، زبان شخص را در فایلی ذخیره کند.
- روی یک slider سطح معدل شخص را مشخص کند.
- معدل شخص را از ۲۰ دریافت کند و از ۴ محاسبه کند.
- قابلیت خروج از صفحه هم از طریق دکمه Cancel و هم زیر منوی Exit در منوی فایل را داشته باشد.

در پنجره command کلمه guide را وارد کنید و سپس گزینه Blank GUI را از پنجره ظاهر شده انتخاب کرده و روی ok کلیک کنید. سپس مطابق جدول (۱۰-۱) کنترل‌های مورد نیاز را در صفحه نشان داده شده در شکل (۱۰-۴) قرار دهید.

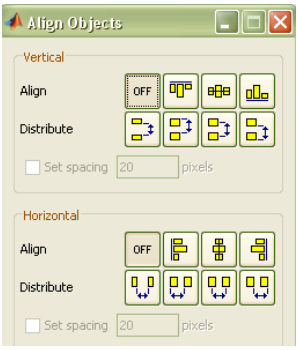
جدول (۱۰-۱) تعداد رابط‌های کنترلی مورد استفاده در این مثال

رابط کنترلی	تعداد
Static Text	10
Edit Text	3
Push Button	2
Pop-Up menu	1
Radio Button	3
Panel	1
Slider	1



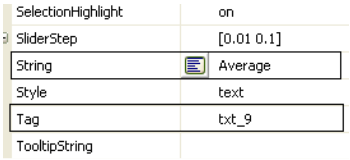
شکل(۱۰-۵) موقعیت کنترل‌های رابط

برای همردیف کردن و تنظیم فاصله بین کنترل‌های رابط می‌توان از پنجره Align Objects استفاده کرد. با انتخاب هر یک از این دکمه‌ها می‌توان نوع آرایش کنترل‌ها را تعیین نمود.



شکل(۱۰-۶) پنجره Align object

با دوبار کلیک کردن روی هر کنترل رابط، پنجره Property Inspector مربوط به آن باز می‌شود. مطابق جدول زیر تغییرات مربوطه را اعمال کنید. به‌طور مثال، در static text9 تغییرات موجود در شکل زیر را انجام دهید.



شکل(۱۰-۷) پنجره Property Inspector

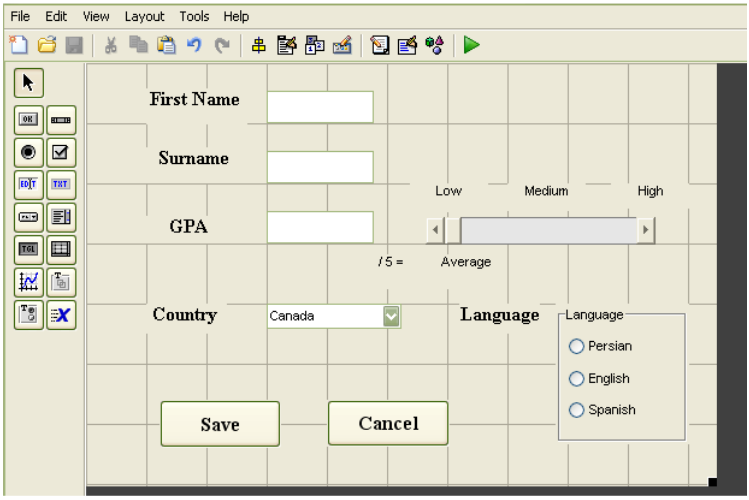
توجه: بین حروف شناسه Tag نباید فاصله وجود داشته باشد.

جدول(۱۰-۲) تنظیم مشخصات کنترل‌های رابط

رابط کنترلی	String	Tag	توضیحات
Static text 1	First Name	--	Fontsize=12
Static text 2	Surname	--	Fontsize=12
Static text 3	GPA	--	Fontsize=12
Static text 4	Country	--	Fontsize=12
Static text 5	Language	--	Fontsize=12
Static text 6	Low	--	
Static text 7	Medium	--	
Static text 8	High	--	
Static text 9	Average	txt_9	
Static text 10	/5=	--	
Edit Text1	متن موجود پاک شود	edit_1	**Tooltipstring**= Enter your First name
Edit Text2	متن موجود پاک شود	edit_2	**Tooltipstring**= Enter your surname
Edit Text3	متن موجود پاک شود	edit_3	**Tooltipstring**= Enter your Grade Point Average(GPA)
Push button 1	Save	push_1	
Push button 2	Cancel	push_2	
Pop-up menu	Canada Iran Spanish USA	pop_1	**Tooltipstring** = Select your country where you live
Radio Button1	Persian	rad_1	
Radio Button2	Spanish	rad_2	
Radio Button3	English	rad_3	
Slider1	--	slider_1	Max=4 ,Min=0 Slide step=0.1
Panel	---	panel_1	Title=Language

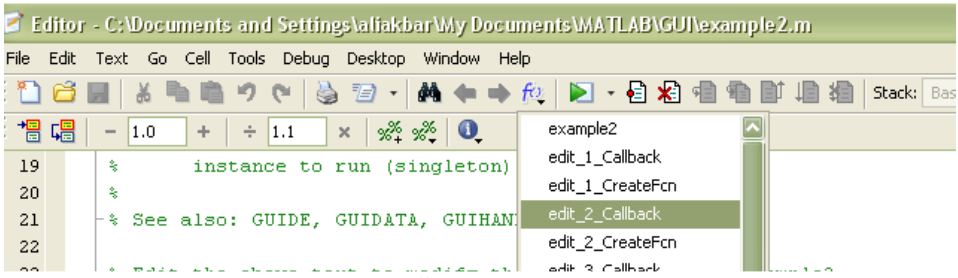
توجه: از دستور Tooltipstring برای نوشتن توضیحات اضافی و راهنمای کاربر استفاده می‌شود، به‌طوری‌که با نگاه داشتن ماوس روی کنترل رابط بعد از اجرای برنامه، توضیحات نمایش داده می‌شوند.

در پایان آرایش صفحه نهایی به‌صورت شکل(۱۰-۸) خواهد بود.



شکل(۱۰-۸) پنجره نهایی کنترل‌های رابط مورد استفاده در این مثال

دکمه Run را فشار دهید. پنجره save as باز شده و برنامه figure ساخته شده را با نام example در پوشه work ذخیره کنید. همزمان با اجرای Run پنجره Editor نیز باز می‌شود. با فشردن دکمه show–f() function در نوار ابزار صفحه Editor گزینه push_1_callback را انتخاب نموده، دستورات زیر را در ادامه function مربوطه وارد کنید.



شکل(۱۰-۹) پنجره Editor برنامه

دستورات زیر را در فضای push_1_Callback وارد کنید.

```
Function push_1_Callback(hObject, eventdata, handles)
a1=get(handles.edit_1,'string');
a2=get(handles.edit_2,'string');
a3=get(handles.edit_3,'string');
a4=get(handles.pop_1,'value');
save data_1 a1 a2 a3 a4
```

دستور close را در فضای push_2_callback وارد کنید.

```
Function push_2_Callback(hObject, eventdata, handles)
close
```

دستورات زیر را در فضای edit_3_Callback وارد کنید.

```
Function edit_3_Callback(hObject, eventdata, handles)
a=str2num(get(handles.edit_3,'string'));
set(handles.txt_9,'string',a/5);
set(handles.slider_1,'value',a/5);
```

دستورات زیر را در فضای rad_1_Callback وارد کنید.

```
Function rad_1_Callback(hObject, eventdata, handles)
set(handles.rad_1,'value',1);
set(handles.rad_2,'value',0);
set(handles.rad_3,'value',0);
```

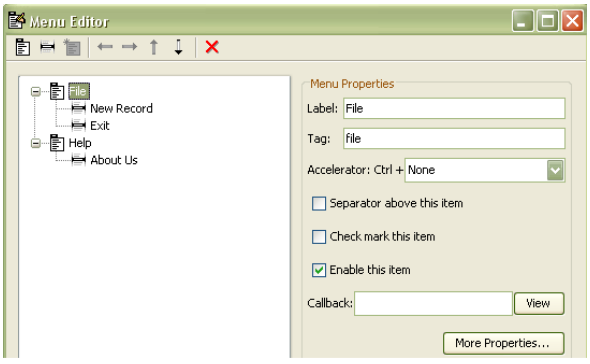
دستورات زیر را در فضای rad_2_Callback وارد کنید.

```
Function rad_2_Callback(hObject, eventdata, handles)
set(handles.rad_1,'value',0);
set(handles.rad_2,'value',1);
set(handles.rad_3,'value',0);
```

دستورات زیر را در فضای rad_3_Callback وارد کنید.

```
Function rad_3_Callback(hObject, eventdata, handles)
set(handles.rad_1,'value',0);
set(handles.rad_2,'value',0);
set(handles.rad_3,'value',1);
```

با کلیک کردن بر روی دکمه Menu Editor در ابزار پنجره GUI می‌توان منو و زیر منو ایجاد کرد. منوها معمولاً در نواری هستند که در بالای پنجره‌ها قرار دارند و شامل لیستی از انتخاب‌ها می‌باشند. با کلیک‌کردن روی عنوان منو لیستی از انتخاب‌ها ظاهر می‌شود. منوها می‌توانند دارای زیر منو باشند. با کلیک روی دکمه New Menu، منوی جدیدی ایجاد می‌شود. سپس روی گزینه ایجاد شده کلیک کنید و عبارت File را در مستطیل روبه‌روی خصوصیت Label وارد کنید و نام Tag را به file تغییر دهید.



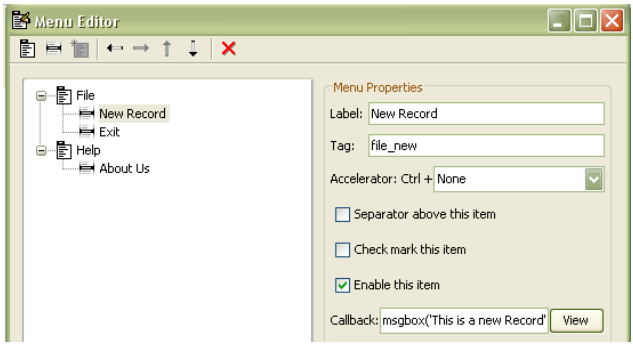
شکل(۱۰-۱۰) پنجره Menu Editor

برای ایجاد زیر منو نیز روی File کلیک کرده و سپس روی کلید New Menu Item کلیک نمایید. سپس در مستطیل روبروی قسمت‌های Label و Tag عبارات New Record و file_new را به ترتیب تایپ کنید. در

قسمت callback نیز دستور msgbox(‘This is a new Record’) را تایپ کنید. برای ایجاد منوی Help و زیر منوهای About Us , Exit نیز به همین طریق عمل کنید.

جدول(۱۰-۳) تنظیمات پنجره Menu Editor

Label	Tag	Callback
Help	help	msgbox(‘This is a new Record’)
Exit	File_exit	----
About Us	Help_us	helpdlg(‘Contact us 0912- 0000000’,help)

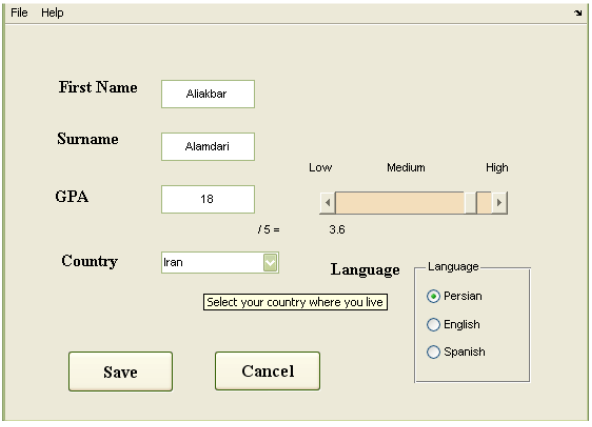


شکل(۱۰-۱۱) پنجره Menu Editor برای تنظیم منوها

دستور زیر را در فضای file_exit_Callback وارد کنید.

```
Function file_exit_Callback(hObject, eventdata, handles)
close
```

با اجرای برنامه، صفحه شکل (۱۰-۱۲) ظاهر می‌شود که می‌توانید اطلاعات را وارد و آنها را save کنید.



شکل(۱۰-۱۲) پنجره نهایی اجرا شده

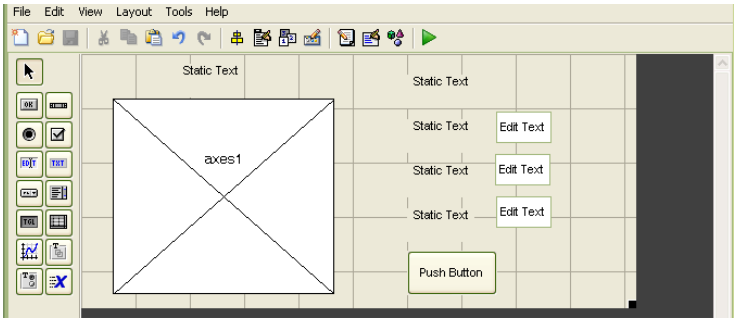
۱۰- ۴ آشنایی با بلوک Axes

مثال: صفحه گرافیکی طراحی کنید که پارامترهای یک تابع را گرفته و نمودار آن را رسم کند.

در پنجره command کلمه guide را وارد کنید و سپس گزینه Blank GUI را از پنجره ظاهر شده انتخاب کرده و روی Ok کلیک کنید. سپس مطابق جدول (۱۰-۴) کنترل‌های مورد نیاز را در صفحه خالی مطابق شکل قرار دهید.

جدول(۱۰-۴) تعداد رابط‌های کنترلی مورد استفاده در این مثال

رابط کنترلی	تعداد
Static Text	5
Edit Text	4
Push Button	1
Axes	1



شکل(۱۰-۱۳) موقعیت کنترل‌های رابط

با دوبار کلیک کردن روی هر کنترل رابط، پنجره Property Inspector مربوط به آن باز می‌شود. مطابق جدول (۱۰-۵) تغییرات مربوطه را اعمال کنید.

جدول(۱۰-۵)

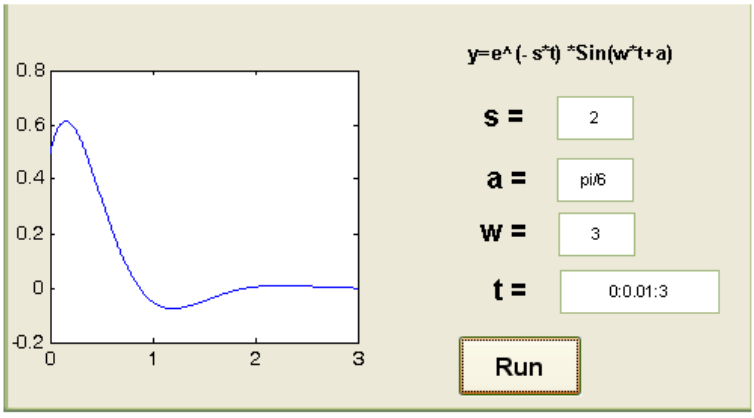
رابط کنترلی	String	Tag	توضیحات
Static text 1	y=e^(-st)*sin(wt+a)	--	Fontsize=10
Static text 2	s =	--	Fontsize=12
Static text 3	a =	--	Fontsize=12
Static text 4	w =	--	Fontsize=12
Static text 5	t =	--	Fontsize=12
Edit Text1	متن موجود پاک شود.	Edit_1	
Edit Text2	متن موجود پاک شود.	Edit_2	
Edit Text3	متن موجود پاک شود.	Edit_3	
Edit Text3	متن موجود پاک شود.	Edit_4	
Push button 1	Run	Push_1	Fontsize=14
Axes1	--	Axes_1	

دکمه Run را فشار دهید. پنجره save as باز شده و برنامه figure ساخته شده را با نام example_1 در پوشه work ذخیره کنید. همزمان با اجرای Run پنجره Editor نیز باز می‌شود. با فشردن دکمه show–

f() function در نوار ابزار صفحه Editor گزینه push_1_callback را انتخاب کنید و دستورات زیر را در ادامه function مربوطه وارد کنید.

دستورات زیر را در فضای push_1_Callback وارد کنید.

```
function push_1_Callback(hObject, eventdata, handles)
s=str2num(get(handles.edit_1,'string'));
a=str2num(get(handles.edit_2,'string'));
w=str2num(get(handles.edit_3,'string'));
t=eval(get(handles.edit_4,'string'));
y=exp(-s*t).*sin(w*t+a);
axes(handles.axes_1)
plot(t,y)
```



شکل(۱۰-۱۴) پنجره نهایی اجرا شده

❖ تمرینات تکمیلی

۱- پنجره‌ای شامل دو Axes، سه static text، سه Edit text و یک push Button بسازید که دو فرکانس f1,f2 و بردار زمان را بگیرد و با فشردن push button در نمودار اول مجموع دو تابع سینوسی $\sin(2\pi tf_1)$ و $\sin(2\pi tf_2)$ و در نمودار دوم تابع $\frac{\sin(x)}{x}$ را رسم کند.

۲- پنجره‌ای شامل سه push button، یک static text، سه Edit text، یک pup-up menu، و یک Axes بسازید. سپس برنامه‌ای بنویسید که با انتخاب هر یک از گزینه‌های peaks, sinc از pup-up menu نمودار آن را در Axes رسم کند و با فشردن هر یک از دکمه‌های push button1,2,3 به ترتیب surf, mesh, contour نمودار جاری را رسم کند.

فصل یازدهم

حل معادلات دیفرانسیل

۱۱–۱ حل عددی معادلات دیفرانسیل معمولی (IVP[1])

روشهای زیادی برای حل عددی معادلات دیفرانسیل وجود دارد که هر کدام از این روش‌ها برای نوعی از معادلات کاربرد دارند. در MATLAB نیز توابع متفاوتی بر اساس این روش‌ها وجود دارد؛ برای مثال ode45 که براساس روش رانگ کوتا عمل می‌کند.

[t,Y] = solver(odefun,tspan,y0)

فرم کلی حل عددی معادلات دیفرانسیل به این صورت است که به جای solver در دستور بالا نوع ode مورد نظر باید ذکر شود.

منظور از odefun تابعی است که در آن معادلات دیفرانسیل مورد نظر تعریف شده است. t متغیر مستقل اسکالر و y متغیر وابسته به صورت بردار ستونی است.

dydt = odefun(t,y)

منظور از tspan بازه‌ای است که معادله در آن حل می شود و y0 شرایط اولیه است.

حتماً می‌دانید که هر معادله دیفرانسیل با درجه n را می‌توان به n معادله درجه اول تبدیل کرد. از این روش برای حل معادلات با درجه بالاتر از یک استفاده می‌شود. در فصل پنجم مثال‌های متعددی از حل این نوع معادلات دیفرانسیل را مشاهده نمودید. به‌طور مثال معادله دیفرانسیل $4x\ddot{y}+6\dot{y}-y=0$ را در نظر بگیرید. برای حل تحلیلی این معادله کافی است بنویسیم:

```
f=dsolve('D2y =(y-6*Dy)/t/4','y(1)=2','y(2)=3');
ezplot(f,[1 , 10]);
```

و اما برای حل عددی باید تابع odefun را بنویسید:

```
function dy=odefun(t,y)
A = [0 1;1/t/4 -6/t/4];
dy = A*y;
```

و در خط فرمان MATLAB دستور زیر را وارد کنید:

```
[t,y]=ode45('odefun',[1 ,10],[2;3]);
```

متغیر y دو ستون دارد که ستون اول به y1 و ستون دوم به y2 اختصاص دارد. لازم به ذکر است که y1 جواب معادله و y2 مشتق آن است.

۱۱–۲ حل معادلات دیفرانسیل با استفاده از روش رانگ کوتای مرتبه چهار

[1] Initial Value Problems

مثال۱: برنامه‌ای بنویسید که معادله دیفرانسیل $y'' = -0.1y' - x$ با شرایط $y(0) = 0$ و $y'(0) = 1$ را در محدوده $0 \le x \le 2$ با گام‌های h=0.25 و با روش رانگ کوتا مرتبه چهار حل کند.

اجازه دهید فرض کنیم:

$$y_1 = y, y_2 = y'$$

بنابراین معادله بالا را می‌توان به فرم زیر نوشت:

$$F(x,y) = y' = \begin{bmatrix} y_1' \\ y_2' \end{bmatrix} = \begin{bmatrix} y_2 \\ -0.1y_2 - x \end{bmatrix}$$

سپس در یک m-file معادله بالا را وارد می کنیم:

```
function F = f(x,y)
% Differential. eqs. used in Example 7.4
F = zeros(1,2);
F(1) = y(2); F(2) = -0.1*y(2) - x;
```

همانطور که می‌دانیم، تابع رانگ کوتا مرتبه چهار نیز به صورت زیر تعریف می‌شود:

$$k_1 = hF(x,y)$$

$$k_2 = hF(x + \frac{h}{2}, y + \frac{k_1}{2})$$

$$k_3 = hF(x + \frac{h}{2}, y + \frac{k_2}{2})$$

$$k_4 = hF(x + h, y + k_3)$$

$$y(x + h) = y(x) + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

بنابراین معادلات بالا را در یک m-file جدید وارد می‌کنیم:

```
function [xSol,ySol] = runKut4(dEqs,x,y,xStop,h)

% INPUT:
% dEqs = handle of function that specifies the
% 1st-order differential equations
% F(x,y) = [dy1/dx dy2/dx dy3/dx ...].
% x,y = initial values; y must be row vector.
% xStop = terminal value of x.
% h = increment of x used in integration.

% OUTPUT:
% xSol = x-values at which solution is computed.
% ySol = values of y corresponding to the x-values.

if size(y,1) > 1
    y = y';% y must be row vector
```

```
end
xSol = zeros(2,1);
ySol = zeros(2,length(y));
xSol(1) = x;
ySol(1,:) = y;
i = 1;
while x < xStop
    i = i + 1;
    h = min(h,xStop - x);
    K1 = h*feval(dEqs,x,y);
    K2 = h*feval(dEqs,x + h/2,y + K1/2);
    K3 = h*feval(dEqs,x + h/2,y + K2/2);
    K4 = h*feval(dEqs,x+h,y + K3);
    y = y + (K1 + 2*K2 + 2*K3 + K4)/6;
    x = x + h;
    xSol(i) = x; ySol(i,:) = y;
end
```

برای مشاهده پاسخ، دستورات زیر را وارد کنید:

```
[x,y] = runKut4(@f,0,[0 1],2,0.25);
disp([x, y(:,1),y(:,2)])
```

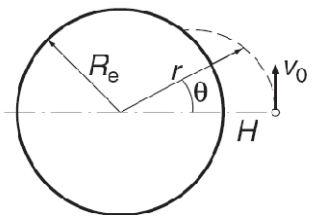
مثال۲: یک فضاپیما در ارتفاع H=772km بالای سطح دریا با سرعت اولیه v0=6700m/s در جهت نشان داده شده در شکل پرتاب می‌شود. معادله دیفرانسیلی که حرکت فضاپیما را بیان می کند به صورت زیر بیان می‌شود:

$$\ddot{r} = r\dot{\theta}^2 - \frac{GMe}{r^2} \qquad \ddot{\theta} = \frac{-2\dot{r}\dot{\theta}}{r}$$

که در آن، r, θ مختصات قطبی فضاپیما هستند. در این مساله داریم:

$$\begin{cases} G = 6.67\times10^{-11} m^3 kg^{-1} s^2 \\ M_e = 5.97\times10^{24} kg \\ R_e = 6378.1 km \end{cases} \rightarrow GMe = 3.98\times10^{14}\ \frac{m^3}{s^2}$$

از روش رانگ کوتای مرتبه چهار، برای انتگرال از معادلات از زمان پرتاب تا زمانی‌که فضاپیما به زمین برخورد می‌کند استفاده کرده و مقدار θ را در لحظه برخورد به‌دست می‌آوریم.



شکل(۱-۱۱) حرکت فضاپیما

فرض می کنیم:

$$y=\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix}=\begin{bmatrix} r \\ \dot{r} \\ \theta \\ \dot{\theta} \end{bmatrix} \rightarrow \quad \dot{y}=\begin{bmatrix} \dot{y}_1 \\ \dot{y}_2 \\ \dot{y}_3 \\ \dot{y}_4 \end{bmatrix}=\begin{bmatrix} y_1 \\ y_2 y_3^{\,2}-3.98\times10^{14}/y_0^{\,2} \\ y_3 \\ -2y_1y_3/y_0 \end{bmatrix}$$

$$r(0)=R_e+H=7.15\times10^6\,m$$

$$\dot{r}(0)=0$$

$$\theta(0)=0$$

$$\dot{\theta}(0)=\frac{v_0}{r(0)}=\frac{6700}{7.15\times10^6}=0.93\times10^{-3}\,rad/s$$

$$y(0)=\left[7.15\times10^6;0;0;0.93\times10^{-3}\right]$$

حال در یک m-file معادله دیفرانسیل حاکم بر حرکت فضاپیما را وارد می کنیم:

```
function F = f2(x,y)
F = zeros(1,4);
F(1) = y(2);
F(2) = y(1)*y(4)^2 - 3.98e14/y(1)^2;
F(3) = y(4);
F(4) = -2*y(2)*y(4)/y(1);
```

برای مشاهده خروجی در پنجره فرامین، دستورات زیر را وارد کنید:

```
x = 0; y = [7.15e6 0 0 0.93e-3];
xStop = 1200; h = 50; freq = 2;
[xSol,ySol] = runKut4(@f2,x,y,xStop,h);
disp([xSol ySol(:,1) ySol(:,2) ySol(:,3) ySol(:,4)])
```

زمان برخورد فضاپیما به زمین زمانی است که r=Re است. بنابراین زمان برخورد برابر 34.18s و زاویه برخورد برابر 1.048 رادیان یا 60 درجه خواهد بود.

۳–۱۱ حل عددی معادلات BVP[۲] یا مسایل با مقادیر مرزی

حل معادلات BVP با استفاده از دستور bvp4c انجام می‌شود. ساده‌ترین حالت استفاده از این دستور به صورت زیر است:

sol = bvp4c(odefun,bcfun,solinit)

که odefun تابعی است که معادله را تعریف می‌کند و bcfun تابعی است که شرایط مرزی را بیان نموده و برای شرایط مرزی دو نقطه‌ای(a,b) به فرم زیر بیان می‌شود:

res = bcfun(ya,yb)

[2] Boundary Value Problems

solinit نیز به فرم زیر بوده، و شامل یک حدس اولیه برای مساله است.

solinit = bvpinit(x, yinit, params)

به‌طور مثال فرض کنید y در معادله زیر صدق کند:

D2y = sin(2*x) - x

و داشته باشیم:

y(-pi) = 2
y(pi) = 2.3

با توجه به شرایط داده شده، مساله از نوع BVP و از مرتبه دوم است. هر زمان مرتبه معادله دیفرانسیل از یک بیشتر باشد، باید آن را به معادلات مرتبه اول تجزیه کنیم.

اگر

y(1) = y
y(2) = Dy

خواهیم داشت:

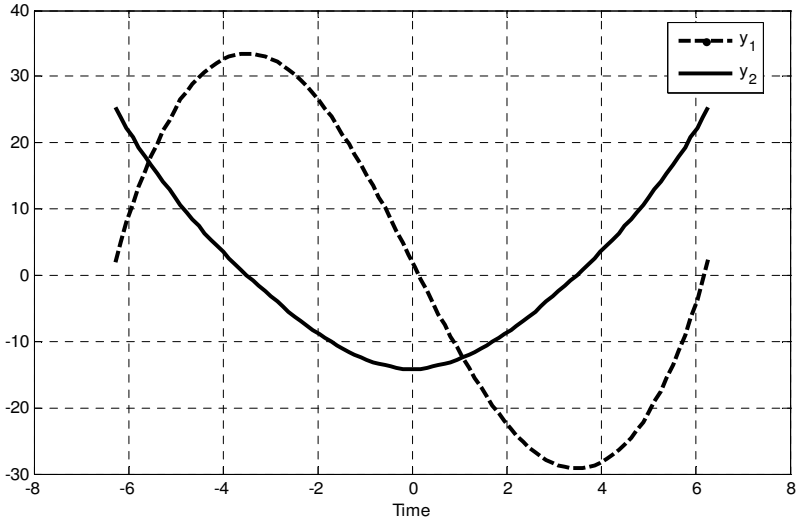
dydx = [y(2);sin(2*x)-x];

حال شرایط مرزی را می نویسیم. اگر res مشخص کننده شرایط مرزی باشد، شرایط مرزی را طوری می نویسیم که مقدار res در آن نقاط صفر شود، یعنی:

```
res = [ ya(1)-2;yb(1)-2.3];
```

منظور از a,b نقاطی است که مقدار مرزی در آن تعریف شده است.

```
function bvptest
solinit = bvpinit(linspace(-2*pi,2*pi,10),[-1 0]);
sol = bvp4c('bvpode1','bvpbc1',solinit);
x = linspace(-2*pi,2*pi);
y = deval(sol,x);
plot(x,y(1,:),'r',x,y(2,:),'b');
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
function dydx = bvpode1(x,y)
dydx = [ y(2);sin(x)+2*x];
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
function res = bvpbc1(ya,yb)
res = [ ya(1)-2;yb(1)-2.3];
```



شکل(۱۱-۲) پاسخ y1 و y2 بدست آمده از حل معادله BVP

همان‌طور که می‌بینید در قسمت دوم برنامه از دستور bvpint استفاده شده است. این دستور برای تعریف حدس اولیه به‌کار می‌رود.

solinit = bvpinit(x,yinit)

بردار x بازه‌ای است که معادله در آن حل می شود و y حدس اولیه است. در این مثال، معادله را در بازه [-2pi,2pi] حل کرده و این بازه را به ۱۰ قسمت مساوی تقسیم کرده‌ایم. کار دستور deval محاسبه جواب معادله در یک بازه دیگر است، این بازه باید زیر مجموعه بازه قبلی باشد.

مثال: معادله دیفرانسیل زیر را با شرایط مرزی معلوم حل کنید.

$$\varepsilon y' = \sin^2(x) - \lambda \frac{\sin^4(x)}{y}$$

در صورتی‌که شرایط مرزی برابر $y(\frac{\pi}{2}) = 1$, $y(-\frac{\pi}{2}) = 1$ باشد.

پارامتر ε معلوم و برابر 0.1 است. ابتدا تابع مربوط به معادله ODE را در یک function نوشته و ذخیره کنید.

```
function dydx = ode11(x,y,lambda)
epsilon = 0.1;
dydx = (sin(x)^2 - lambda*sin(x)^4/y)/epsilon;
```

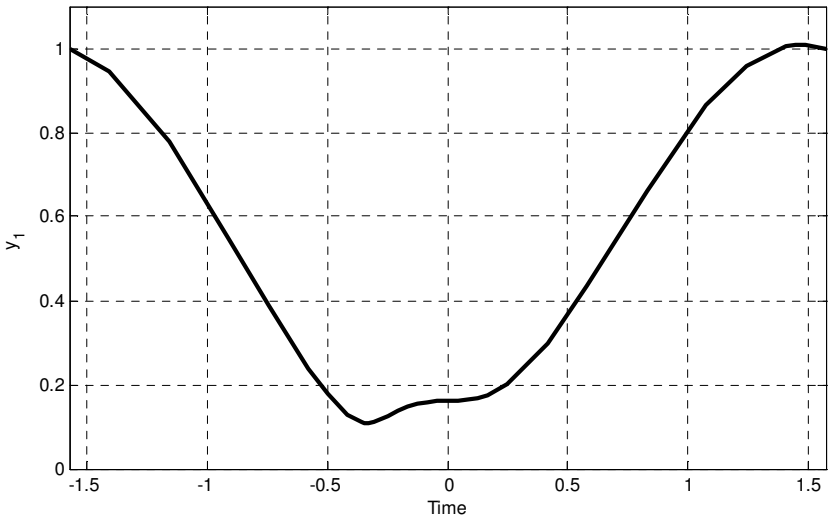
تابع مربوط به شرایط مرزی را نوشته و ذخیره کنید.

```
function res = bcs(ya,yb,lambda)
res = [ ya-1; yb-1 ];
```

حال در یک m-file جدید دستورات زیر را وارد کرده و اجرا کنید:

```
function sol = exam
solinit = bvpinit(linspace(-pi/2,pi/2,20),0.5,1);
sol = bvp4c(@ode11,@bcs,solinit);
```

```
plot(sol.x,sol.y(1,:));
axis([-pi/2 pi/2 0 1.1]);
```



شکل(۱۱-۳) پاسخ y1 بر حسب x

۱۱-۴ معادلات دیفرانسیل با مشتقات جزئی

اگر معادله‌ای شامل چند متغیر و مشتقات آنها باشد، آن معادله یک معادله دیفرانسیل نامیده می‌شود. معادله دیفرانسیل با مشتقات جزئی معادله‌ای است که شامل یک تابع و متغیرهای آن و مشتقات جزئی مربوط می باشد. فرم کلی معادلات دیفرانسیل با مشتقات جزئی به صورت زیر می‌باشد.

$$G(u,u_x,u_{xx},.....,u_x^{(n)},u_y,u_{yy},.....,u_y^{(m)},u_{xy}^{(i+j)},.....)=0$$

که در آن

$$u_x=\frac{\partial y}{\partial x}\quad \frac{\partial u}{\partial y}=u_y\quad u_{xy}=\frac{\partial^2 u}{\partial x\partial y}$$

مجموعه معادلات PDE یک مجموعه معادلات نامحدود می‌باشد. از بین این مجموعه تنها بخش کوچکی دارای کاربردهای خاص می‌باشد. غالب مسائلی که در کاربردهای PDE مطرح می‌شوند دارای فرم‌های محاسباتی زیر می‌باشند، که در آن G ,..... , B , A ضرایب معادله PDE هستند.

$$Au_{xx}(x,y)+Bu_{xy}+Cu_{yy}(x,y)+Du_x(x,y)+Eu_y(x,y)+Fu(x,y)+G=0$$

چنانچه A,B, …G توابعی از x,y باشند، معادله درجه دو خطی(به طور مثال $zu_{xx}(x,y)+xy^2u_{yy}(x,y)+e^xu(x,y)=10$ نامیده می‌شود، ولی چنانچه A, B, …..,G تابعی از U باشد، معادله غیرخطی (به طور مثال $u_{xx}+(\sin u)_{uy}=0$) نامیده می‌شود.

معادلات دیفرانسیل با مشتقات جزئی روی خط، صفحه و یا فضا تعریف می‌شوند. در بسیاری از کاربردها، مقادیر متغیر U در مرز ناحیه معلوم می‌باشد و نیز در برخی کاربردها مقدار تابع U در نقطه شروع زمان و مکان معلوم

می‌باشد. مسائل گروه اول Boundary value Problems (مسائل با مقادر مرزی) و مسائل گروه دوم را Initial Value Problems می‌نامیم.

حالت ترکیبی از مقادیر مرزی و اولیه نیز وجود دارد. یک معادله دیفرانسیل، خطی نامیده می‌شود وقتی که معادله و شرایط اولیه یا مقادیرمرزی آن خطی باشد.

مسأله‌ای را مسأله دیریشله یا نوع اول گویند که در شرایط مرزی، مقدار u روی مرز دانسته شود، و مساله‌ای را مسأله نیومن گویند که مقدار مشتق u روی بخشی از مرز داده شده باشد.

معادلات به فرم $Au_{xx}+Bu_{xy}+Cu_{yy}+Du_x+Eu_y+Fu=G$ خطی درجه ۲ هستند وقتی A تا F تابعی از x , y , t باشند. چنانچه ضرایب F-A عدد ثابتی باشند، معادله را معادله خطی درجه ۲ با ضریب ثابت می‌نامیم. حال اگر $B^2-4Ac>0$ معادله را هذلولی یا Hyperbolic گوئیم و اگر $B^2-4Ac=0$ باشد، معادله را بیضوی یا Elliptic نامیم و اگر $B^2-4Ac>0$ باشد، آنرا سهموی گوئیم (Parabolic).

در جدول زیر چند نمونه از معادلات دیفرانسیل با مشتقات جزئی را مشاهده می‌کنید.

معادله موج درتک بعد از نوع معادلات هذلولی است.	$\frac{\partial^2 u}{\partial t^2}=c^2\frac{\partial^2 u}{\partial x^2}$
معادله حرارت درتک بعد از نوع سهموی است.	$\frac{\partial u}{\partial t}=c^2\frac{\partial^2 u}{\partial x^2}$
معادله لاپلاس در دو بعد از نوع بیضوی است.	$\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}=0$
معادله پواسون در دو بعد از نوع بیضوی است.	$\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}=f(x,y)$

۱۱–۵ حل معادلات دیفرانسیل با مشتقات جزئی وابسته به زمان در یک بعد

فرض کنید u تابعی است که در معادله زیر صدق می‌کند. دستور pdepe می‌تواند PDE هایی به فرم معادله زیر را حل کند.

$$c(x,t,u,\frac{\partial u}{\partial x})\frac{\partial u}{\partial t}=x^{-m}\frac{\partial}{\partial x}(x^m f(x,t,u,\frac{\partial u}{\partial x})+s(t,x,u,\frac{\partial u}{\partial x})$$

$$t_0<t<t_f$$

$$a\le x\le b$$

در این معادله، بازه [a,b] باید محدود و m می‌تواند برابر صفر، یک و دو باشد. هر یک از این اعداد به ترتیب مربوط به صفحه تخت، استوانه و کره می باشند یعنی m به هندسه مساله بر می‌گردد.

در این معادله ترم $f\ (x,t,u,\frac{\partial u}{\partial x})$ مربوط به شار انتقال حرارت و s مربوط به تولید و مصرف انرژی می باشد. مشتقات پاره‌ای نسبت به زمان با ماتریس قطری $c(x,t,u,\frac{\partial u}{\partial x})$ محدود می‌شوند. از طرفی شرایط مرزی معادلهبه صورت زیر می‌باشد:

$$p(x,t,u)+q(x,t)f(x,t,u,\frac{\partial u}{\partial x})=0$$

الگوی نوشتاری دستور pdepe به صورت زیر است:

Solution=pdepe(m,pdefun,icfun,bcfun,xmesh,tspan)

که m مشخص کننده هندسه مساله است که برای ورق m=0 ، برای استوانه m=1 و برای کره m=2 است.

pdefun: تابعی است که معادله را تعریف می کند.

$$(c,f,s)=\text{pdefun}(x,t,u,du,dx)$$

که c,f,s همان پارامترهای معادله دیفرانسیل پاره‌ای هستند.

icfun: تابعی است که شرایط اولیه را تعریف می‌کند.

u=icfun(x)

bcfun: تابعی است که شرایط مرزی را تعریف می کند.

$$[p_L,q_L,p_r,q_r]=\text{bcfun}(x_L,u_L,x_r,u_r,t)$$

u_L حل تقریبی در مرز چپ x_L=a است.

u_r حل تقریبی در مرز راست xr=b است.

p_L,q_L بردارهای ستونی متناظر با p,q در x_L و متعاقباً p_r و q_r بردارهای ستونی متناظر با p و q در x_r هستند.

xmesh برداری شامل نقاط x_L تا x_n است.

tspan بردار زمان متناظر با بردار xmesh است .

مثال۱: معادله دیفرانسیل با مشتقات جزئی زیر را با شرایط اولیه و مرزی داده شده حل کنید.

$$\pi^2\frac{\partial u}{\partial t}=\frac{\partial}{\partial x}(\frac{\partial u}{\partial x})$$

$$0\le x\le 1,\quad 0\le t$$

```
%~~~~~~~~~~~~~~~~~~~~~~~~~
function [c,f,s] = pdex1pde(x,t,u,DuDx)
c = pi^2;
f = DuDx;
s = 0;
%~~~~~~~~~~~~~~~~~~~~~~~~~
```

شرایط اولیه مسئله به صورت زیر می باشد:

$U(x,0)=\sin\pi x$

```
%~~~~~~~~~~~~~~~~~~~~~~~~
function u0 = pdex1ic(x)
u0 = sin(pi*x);
%~~~~~~~~~~~~~~~~~~~~~~~~
```

و شرایط مرزی مساله به صورت زیر بیان می شود:

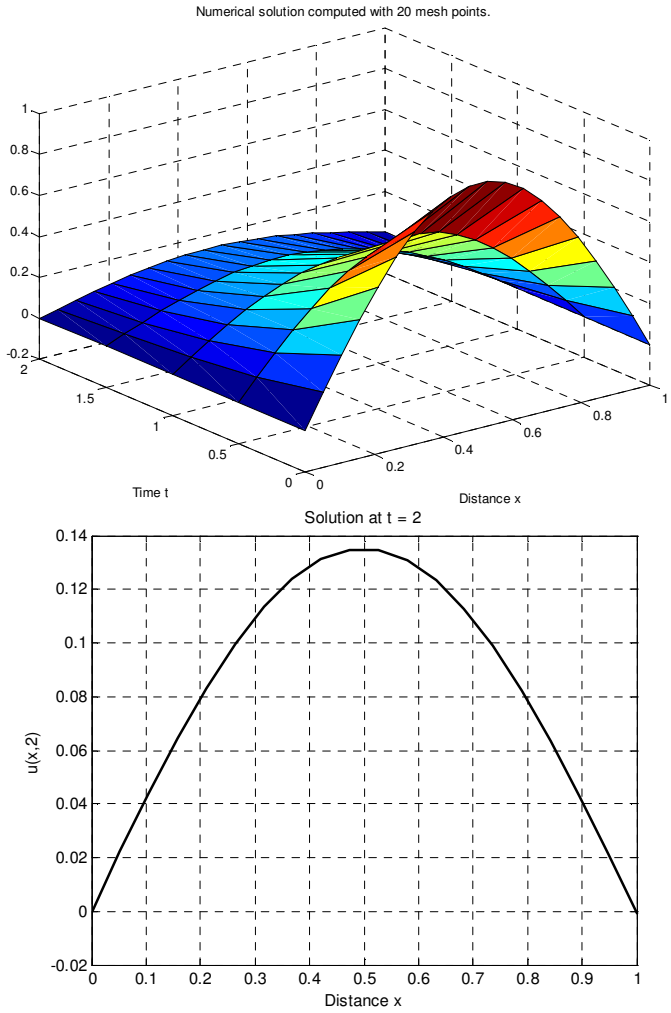
$$u(0,t)=0$$

$$\pi e^{-t}+\frac{\partial u}{\partial x}(1,t)=0$$

```
%~~~~~~~~~~~~~~~~~~~~~~~~
function [pl,ql,pr,qr] = pdex1bc(xl,ul,xr,ur,t)
pl = ul;
ql = 0;
pr = pi * exp(-t);
qr = 1;
%~~~~~~~~~~~~~~~~~~~~~~~~
```

برای حل مساله از دستور pdepe استفاده می‌کنیم.

```
function pdex1

m = 0;
x = linspace(0,1,20);
t = linspace(0,2,5);

sol = pdepe(m,@pdex1pde,@pdex1ic,@pdex1bc,x,t);
% Extract the first solution component as u.
u = sol(:,:,1);

% A surface plot is often a good way to study a solution.
surf(x,t,u)
title('Numerical solution computed with 20 mesh points.')
xlabel('Distance x')
ylabel('Time t')

% A solution profile can also be illuminating.
figure
plot(x,u(end,:))
title('Solution at t = 2')
xlabel('Distance x')
ylabel('u(x,2)')
%~~~~~~~~~~~~~~~~~~~~~~~~
```



شکل(۱۱-۴) پاسخ u(x) در t=2

مثال۲: معادلات دیفرانسیل با مشتقات جزئی زیر را حل کنید.

$$\frac{\partial u_1}{\partial t} = 0.02\frac{\partial^2 u}{\partial x^2} - F(u_1 - u_2)$$

$$\frac{\partial u_2}{\partial t} = 0.17\frac{\partial^2 u}{\partial x^2} + F(u_1 - u_2)$$

$$F(y) = e^{5.73y} - e^{-11.46y}$$

$$0 \le x \le 1, 0 \le t$$

شرایط اولیه مسئله برابر است با:

$$u_1(x,0)=1,\ u_2(x,0)=0$$

```
%~~~~~~~~~~~~~~~~~~~~~~~~
function u0 = pdex4ic(x);
u0 = [1; 0];
%~~~~~~~~~~~~~~~~~~~~~~~~
```

شرایط مرزی مساله برابر است با:

$$\frac{\partial u_1}{\partial x}=(0,t)=0$$

$$u_2(0,t)=0,\ \ u_1(1,t)=1$$

$$\frac{\partial u_2}{\partial x}=(1,t)=0$$

برای حل این مساله از دستور pdepe استفاده می‌کنیم، ولی در ابتدا باید معادلات را به فرم معادله عمومی تبدیل کنیم.

$$\begin{bmatrix}1\\1\end{bmatrix}\times\frac{\partial}{\partial t}\begin{bmatrix}u_1\\u_2\end{bmatrix}=\frac{\partial}{\partial x}\begin{bmatrix}0.024(\frac{\partial u_1}{\partial x})\\0.17(\frac{\partial u_2}{\partial x})\end{bmatrix}+\begin{bmatrix}-F(u_1-u_2)\\F(u_1-u_2)\end{bmatrix}$$

```
%~~~~~~~~~~~~~~~~~~~~~~~~
function [c,f,s] = pdex4pde(x,t,u,DuDx)
c = [1; 1];
f = [0.024; 0.17] .* DuDx;
y = u(1) - u(2);
F = exp(5.73*y)-exp(-11.47*y);
s = [-F; F];
%~~~~~~~~~~~~~~~~~~~~~~~~
```

شرایط مرزی چپ مساله را می‌توان به صورت زیر نوشت:

$$\begin{bmatrix}0\\u_2\end{bmatrix}+\begin{bmatrix}1\\0\end{bmatrix}\times\begin{bmatrix}0.024(\frac{\partial u_1}{\partial x})\\0.17(\frac{\partial u_2}{\partial x})\end{bmatrix}=\begin{bmatrix}0\\0\end{bmatrix}$$

شرایط مرزی راست مساله را می‌توان به صورت زیر نوشت:

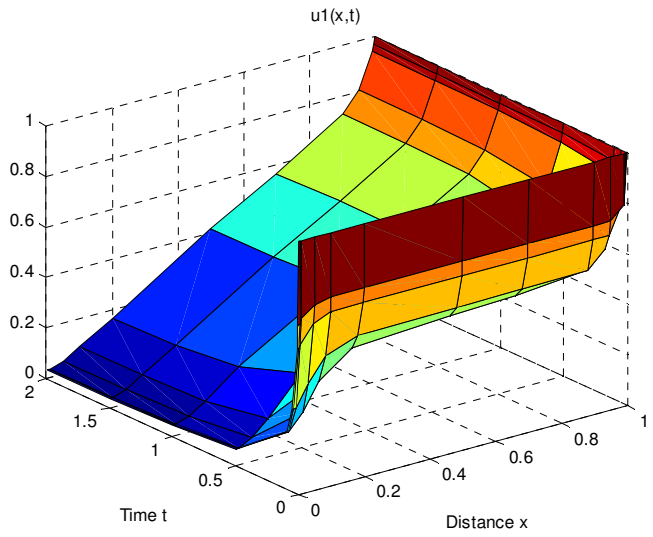
$$\begin{bmatrix}u_1-1\\0\end{bmatrix}+\begin{bmatrix}0\\1\end{bmatrix}\times\begin{bmatrix}0.024(\frac{\partial u_1}{\partial x})\\0.17(\frac{\partial u_2}{\partial x})\end{bmatrix}=\begin{bmatrix}0\\0\end{bmatrix}$$

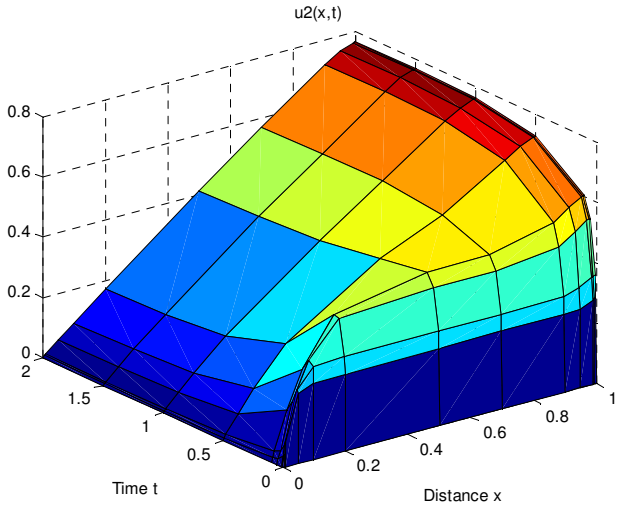
```
%~~~~~~~~~~~~~~~~~~~~~~~~
function [pl,ql,pr,qr] = pdex4bc(xl,ul,xr,ur,t)
pl = [0; ul(2)];
```

```
ql = [1; 0];
pr = [ur(1)-1; 0];
qr = [0; 1];
%~~~~~~~~~~~~~~~~~~~~~~~~~
```

برای حل مساله از دستور pdepe استفاده می‌کنیم.

```
function pdex4
m = 0;
x = [0 0.005 0.01 0.05 0.1 0.2 0.5 0.7 0.9 0.95 0.99 0.995
1];
t = [0 0.005 0.01 0.05 0.1 0.5 1 1.5 2];

sol = pdepe(m,@pdex4pde,@pdex4ic,@pdex4bc,x,t);
u1 = sol(:,:,1);
u2 = sol(:,:,2);

figure
surf(x,t,u1)
title('u1(x,t)')
xlabel('Distance x')
ylabel('Time t')

figure
surf(x,t,u2)
title('u2(x,t)')
xlabel('Distance x')
ylabel('Time t')
%~~~~~~~~~~~~~~~~~~~~~~~~~
```





شکل(۱۱-۵) پاسخ u1 و u2

مثال ۳: معادله حاکم بر یک سیستم به صورت زیر است:

$$\frac{\partial u}{\partial t} = D\frac{\partial^2 u}{\partial x^2}$$

$$D = 10^{-5}$$

شرایط اولیه عبارت است از :

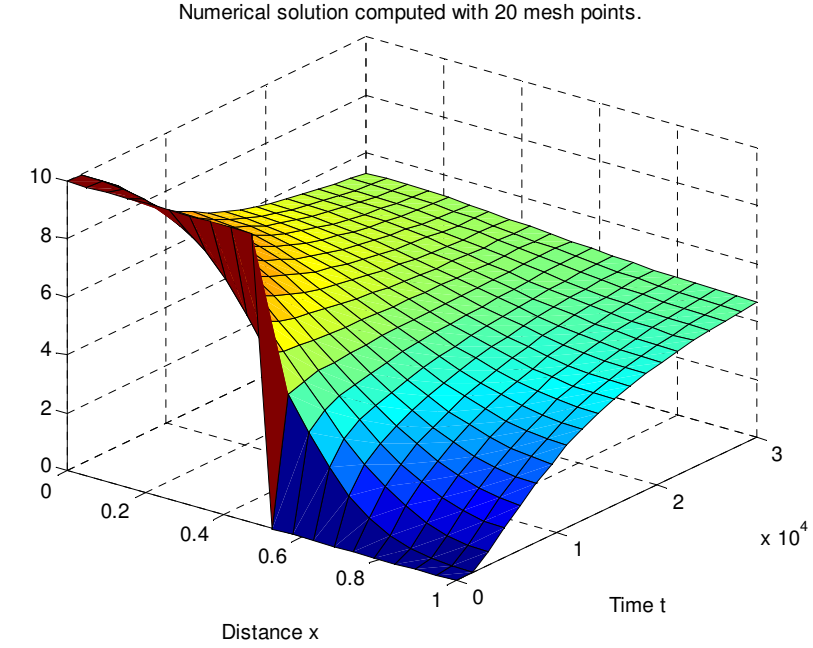
$$u(x,0) = \begin{cases} 10 & 0 \le x \le 0.5 \\ 0 & 0.5 \le x \le 1 \end{cases}$$

و شرایط مرزی مساله برابر است با:

$$\frac{\partial u(0,t)}{\partial x} = \frac{\partial u(1,t)}{\partial x} = 0$$

```
function pdex
x=linspace(0,1,20);
t=linspace(0,30000,20);
m=0;
sol = pdepe(m,@pdefun0,@pdeic0,@pdebc0,x,t);
u=sol(:,:,1);
uu=u;
surf(x,t,u)
title('Numerical solution computed with 20 mesh points.')
xlabel('Distance x')
ylabel('Time t')
%--------------------------------------------
function [c,f,s] = pdefun0(x,t,u,DuDx)
c = 1;
```

```
f = 1e-5*DuDx;
s = 0;
%---------------------------------------------
function u0 = pdeic0(x)
if ((x >= 0) & (x <= .5))
u0=10;
elseif ((x >= 0.5) & (x <= 1))
u0=0;
end
%---------------------------------------------
function [pl,ql,pr,qr] = pdebc0(xl,ul,xr,ur,t)
pl = 0;
ql = 100000;
pr = 0;
qr = 100000;
```



شکل(۱۱-۶) حل عددی معادله دیفرانسیل با ۲۰ نقطه

۱۱-۶ حل معادله موج

معادله موج از نوع معادلات هذلولی است و برای حل آن از دستور hyperbolic استفاده می‌کنیم. الگوی نوشتارهای این دستور به صورت زیر است:

$$u_1 = \text{hyperbolic}(u_0, ut_0, tlist, b, p, e, t, c, a, f, d)$$

$$d\frac{\partial^2 u}{\partial t^2} - \nabla.(c\nabla u) + au = f$$

که در آن، u_0, ut_0 شرایط اولیه

tlist بازه زمانی حل مساله

B شرایط مرزی

P,e,t تقسیم بندی مساله

c,a,f,d ضرایب معادله می باشند که به طرق مختلفی ممکن است داده شوند. این ضرایب همگی به t وابسته هستند. معادله موج زیر را در نظر بگیرید:

$$\frac{\partial^2 u}{\partial t^2} = \Delta u$$

در یک هندسه مربعی شکل (squareg) $0 \le x, y \le 1$ با شرایط مرزی دیریشلت $u = 0$ $x = \pm 1$ و شرایط مرزی نیومن (Newman) $\frac{\partial u}{\partial n} = 0$ $y = \pm 1$ داریم:

$$u(x, y, 0) = a\tan(\cos(\pi x))$$

$$\frac{du(0)}{d(t)} = 3\sin(\pi x).\exp(\cos \pi y))$$

```
 %~~~~~~~~~~~~~~~~~~~~~~~~
[p,e,t]=initmesh('squareg');
x=p(1,:)';
y=p(2,:)';
u0=atan(cos(pi/2*x));
ut0=3*sin(pi*x).*exp(cos(pi*y));
tlist=linspace(0,5,31);
uu=hyperbolic(u0,ut0,tlist,'squareb3',p,e,t,1,0,0,1);
%~~~~~~~~~~~~~~~~~~~~~~~~RESULTS~~~~~~~~~~~~~~~~
462 successful steps
70 failed attempts
1066 function evaluations
1 partial derivatives
156 LU decompositions
1065 solutions of linear systems
```

۷-۱۱ حل معادله حرارت

معادله حرارت از نوع معادلات سهموی است و برای حل آن از دستور parabolic استفاده می‌کنیم. الگوی نوشتاری این دستور به صورت زیر است:

u1=parabolic(u0,tlist,g,b,p,e,t,c,a,f,d)

که در دستور بالا از دستور initmesh برای ایجاد مش‌های مثلثی شده اولیه استفاده می‌شود. الگوی نوشتاری این دستور به صورت زیر است:

$$[p, e, t] = \text{initmesh}(g)$$

که در آن ، خروجی های p,e,t داده های مش بندی هستند و دستور refinemesh مش‌های مثلثی شده را تصحیح می‌کند.

$$[p_1, e_1, t_1] = \text{refinemesh}(g, p, e, t)$$

g به هندسه مساله بر می گردد و در مثال بالا g='squareg' می باشد.

به طور مثال معادله حرارت روبرو را در نظر بگیرید:

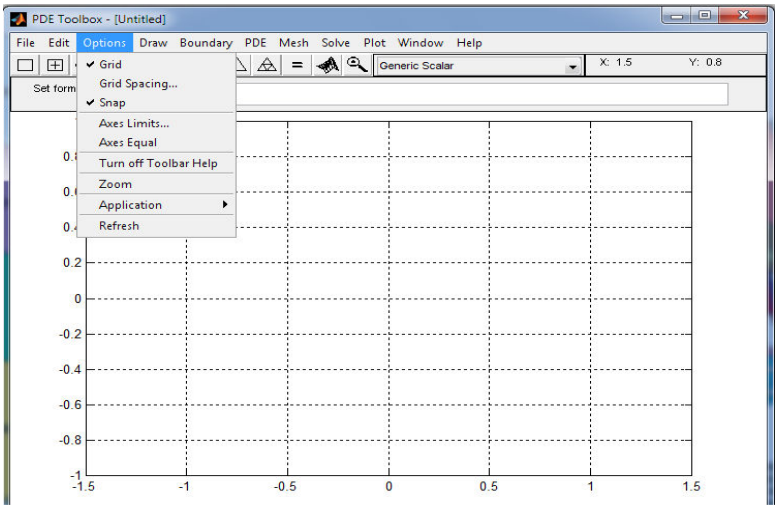
$$\frac{\partial u}{\partial t} = \Delta u$$

در یک هندسه مربعی شکل که (squareg) $0 \le x, y \le 1$است. برای حل مساله در زمان‌های linspace(0,.1,20) دستورات زیر را می نویسیم:

```
[p,e,t]=initmesh('squareg');
[p,e,t]=refinemesh('squareg',p,e,t);
u0=zeros(size(p,2),1);
ix=find(sqrt(p(1,:).^2+p(2,:).^2)<0.4);
u0(ix)=ones(size(ix));
tlist=linspace(0,0.1,20);
u1=parabolic(u0,tlist,'squareb1',p,e,t,1,0,1,1);
%~~~~~~~~~~~~~~~~~~~~~~~~~RESULTS~~~~~~~~~~~~~~~~
96 successful steps
0 failed attempts
194 function evaluations
1 partial derivatives
20 LU decompositions
193 solutions of linear systems
```

۸-۱۱ جعبه ابزار گرافیکی PDE

برای حل معادلات دیفرانسیل با مشتقات جزئی می‌توان از جعبه ابزار گرافیکی PDE (GUI) ها استفاده کرد و به این طریق معادله PDE را گام به‌گام حل کرد. مسئله‌ای که می‌خواهیم شروع به حل آن کنیم معادله پواسون $-\Delta u = f$ است که شرایط مرزی از نوع دیریشلت و نیومن می‌باشد. برای شروع کار در پنجره دستورات دستور pdetool را تایپ کنید. سپس از منوی options گزینه‌های snap و grid را فعال کنید.



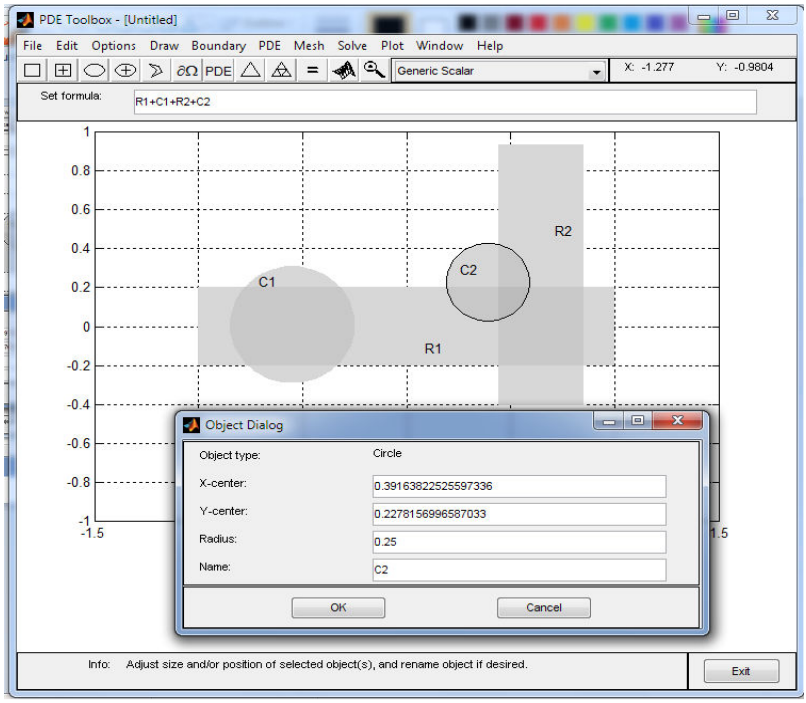
شکل(۱۱-۷) جعبه ابزار گرافیکی حل معادلات PDE

حال باید هندسه مساله را رسم کنید. این GUI چهار شکل هندسی شامل چند ضلعی، مستطیل، دایره و بیضی را برای ما ایجاد می‌کند. از طرفی هر شی یک بر چسب واحد دارد.

با استفاده از ماوس مستطیل را انتخاب کرده و ماوس را روی صفحه بکشید. اگر این کار را با کلیک چپ انجام دهید یک مستطیل، و اگر با کلیک راست انجام دهید یک مربع رسم می‌شود. سپس روی مستطیل دوبار کلیک کرده و نقاط شروع و طول و عرض آن را مشخص کنید.

دایره‌ای را انتخاب کرده سپس با فشار کلیک چپ ماوس آن را روی صفحه بکشید. در آن صورت یک بیضی و اگر با کلیک راست این کار را انجام دهید یک دایره رسم می‌شود. نتیجه حاصل یک مدل CSG (مدل هندسی توپر) می باشد که در قسمت set formula با R_1+C_1 نمایش داده شده است. سطوحی که روی هم افتاده اند کاملاً در شکل واضح‌اند و تا حدودی سایه پررنگ‌تری دارند. شما می‌توانید بر روی هر یک از این اشکال اعمال کپی، حرکت، دوران و یا پاک کردن را انجام دهید.

با استفاده از دکمه ماوس و نگه داشتن shift می‌توانید اشیا را انتخاب کنید. از منوی Edit و گزینه select all، می‌توان تمام اشکال را انتخاب کرد. با استفاده از ابزارهای گفته شده، اشکالی همانند شکل زیر را رسم کنید. همان‌طور که می‌بینید set formula برابر $R_1+C_1+R_2+C_2$ است. این فرمول را به صورت زیر تغییر دهید:

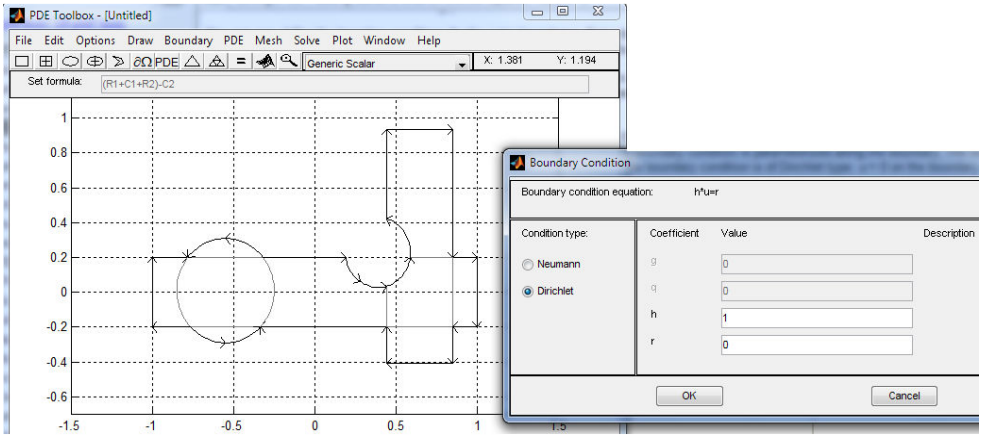


شکل(۱۱-۸) رسم اشکال هندسی در جعبه ابزار PDE

$$(R_1 + C_1 + R_2) - C_2$$

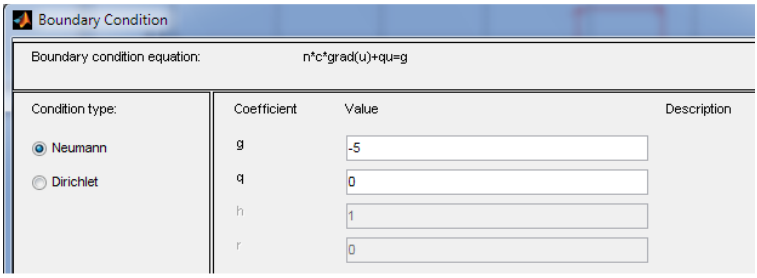
حال فایل را به صورت m-file ذخیره کنید تا تمام مراحل ذخیره شوند.

اکنون می توان شرایط مرزی مساله را با استفاده از آیکون $\partial\Omega$ یا انتخاب Boundary mode از منوی Boundary تعریف کرد. در حالت پیش فرض شرایط مرزی از نوع دیریشلت می‌باشد که با خط قرمز نمایش داده شده اند، می توانید آن را تغییر داده و از نوع نیومن نمائید.



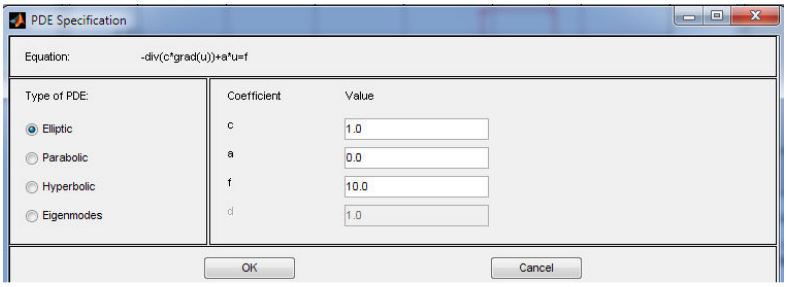
شکل(۱۱-۹) نمایش حالت Boundary mode

روی یکی از لبه‌ها دوبار کلیک کرده سپس نوع شرایط مرزی را به نیومن تغییر دهید. $\frac{\partial n}{\partial u} = -5$ یعنی باید مقدار g=-5 و q=0 باشد. درصورتی که q=0 باشد، شرایط نیومن خالص داریم. در انتها بر روی Ok کلیک کنید.

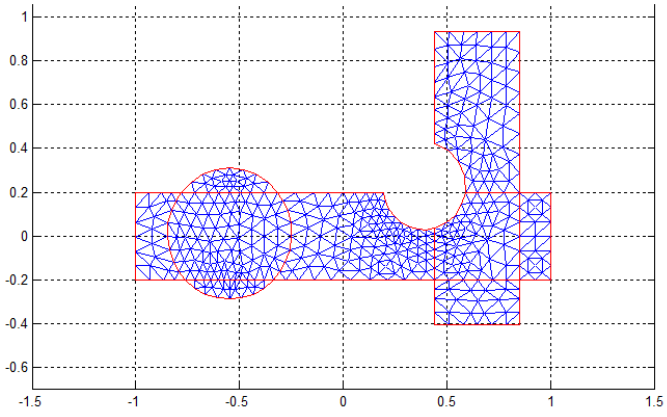


شکل(۱۱-۱۰) تنظیم پارامترهای شرایط نیومن

روی آیکون PDE در نوار ابزار کلیک کنید. به‌دلیل اینکه معادله‌ای که می‌خواهیم حل کنیم از نوع بیضوی است، گزینه Elliptic را انتخاب کرده و پارامترهای c=1,a=0,f=10 را تنظیم می‌کنیم.

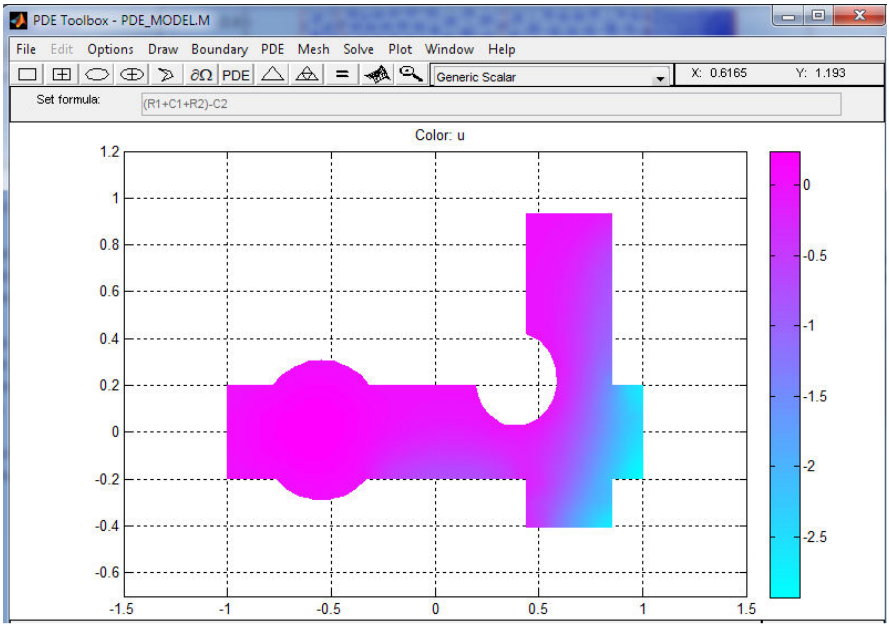


در پایان باید مش مثلثی را برای مدل تعریف کنیم؛ زیرا این جعبه ابزار برای حل PDE از روش FEM استفاده می‌کند. به این منظور روی آیکون مثلثی شکل کلیک کنید، یا از منوی mesh گزینه initialize meshرا انتخاب کنید. اگر می‌خواهید پاسخ دقیق‌تری داشته باشید روی refine کلیک کنید. با استفاده از jiggle mesh می‌توانید کیفیت مثلث ها را بالا ببرید.



شکل(۱۱-۱۱) بالا بردن کیفیت مش بندی با jiggle mesh

حال از منوی solve گزینه solve PDE را انتخاب کنید تا محاسبات انجام شده و نمودار رسم گردد.



شکل(۱۱-۱۲) نمایش حل با استفاده از دستور Solve PDE

مثال: یک ورق فلزی از گوشه پایین سمت چپ کلمپ شده و از لبه دایروی گوشه بالا از سمت راست کشیده می‌شود، بقیه لبه ها آزاد هستند. این ورق دارای خصوصیات زیر است:

- ابعاد ورق یک متر در یک متر
- ضخامت ورق 1mm
- برش ورق 1/3m*1/3m گوشه پایین سمت چپ، برش دایروی (2/3,1) تا (1,2/3)،
- مدول یانگ $196\times10^{6}\ \frac{N}{m^2}$ و ضریب پواسون 0.31 است.
- نیروی خارجی اعمال شده به لبه 500N است که به صورت عمود بر ضخامت 1mm است.

کرنش‌ها و تنش‌ها در جهت x,y و تنش برشی و تنش موثر فون مایزز را محاسبه کنید.

حل مساله:

ابتدا در پنجره دستور pdetool را تایپ نموده و در قسمت application mode گزینه Structural Mechanic, plane stress را انتخاب کنید. مدل CSG را با استفاده از آیکون polygon رسم کنید. گوشه ها در نقاط زیر می‌باشند:

$$x=[0\ \ 2/3\ \ 1\ \ 1\ \ 1/3\ \ 1/3\ \ 0]$$
$$y=[1\ \ 1\ \ 2/3\ \ 0\ \ 0\ \ 1/3\ \ 1/3]$$

سپس دایره‌ای با مرکز (2/3,2/3) و شعاع 1/3 رسم کنید.

نکته: با استفاده از دستورات pderect, pdecirc, pdepoly, pdeelip به ترتیب می‌توان مستطیل، دایره، چند ضلعی ،و بیضی رسم کرد.

الگوی نوشتاری هر یک از دستورات در زیر بیان شده است:

$$pderect(xy) \;\; \% \;\; xy = [x_{min} \quad x_{max} \quad y_{min} \quad y_{max}]$$

$$Ex1: pderect([0\ 1\ 0\ 1])$$

که x,y مختصات گوشه‌های چندضلعی می‌باشد.

$$pdecirc(x_c, y_c, radius) \quad \% \ Center = (x_c, y_c)$$

$$Ex2: pdecirc(0,0,1)$$

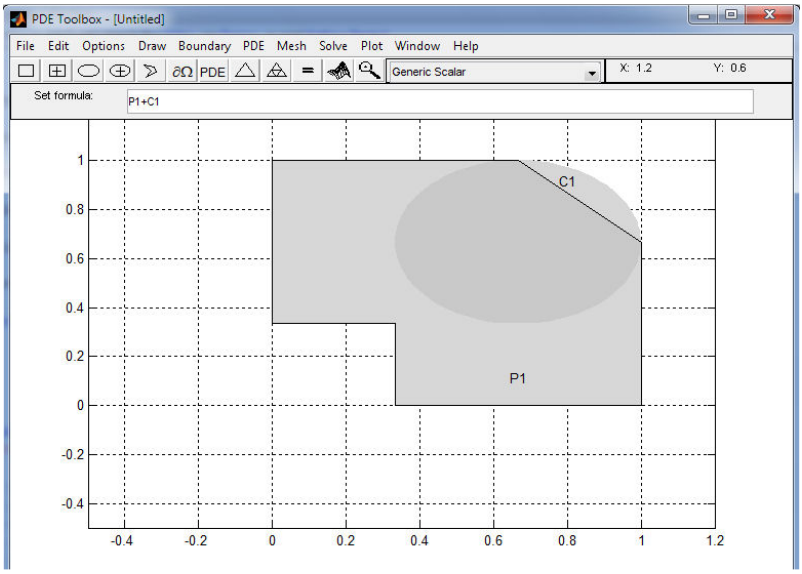
$$pdepoly(x, y)$$

$$Ex3: pdepoly([-1\ 0\ 0\ 1\ 1\ -1],[0\ 0\ 1\ 1\ -1\ -1];$$

$$pdeelip(x_c, y_c, a, b, phi)$$

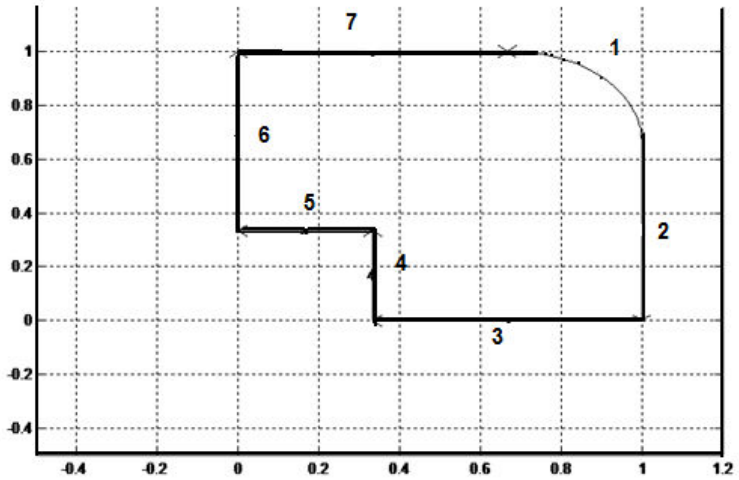
$$Ex4: pdeelip(0,0,1,0.3,pi/4)$$

مرکز بیضی برابر x_c, y_c ، نصف قطرها برابر a,b و میزان دوران با phi تعیین می شود.



شکل(۱۱-۱۳) مدل CSG

مدل حاصله $P_1 + C_1$ خواهد بود. در قدم بعدی از منوی Boundary گزینه Remove all subdomain borderرا انتخاب کنید. در مرحله بعد دو شرط اعمال شده را اعمال می‌کنیم، یعنی برای گوشه پایین سمت چپ شرایط مرزی دیریشلت را تعریف می‌کنیم، چون جابه‌جایی صفر است.



شکل(۱۱-۱۴) نمایش حالت Boundary Mode

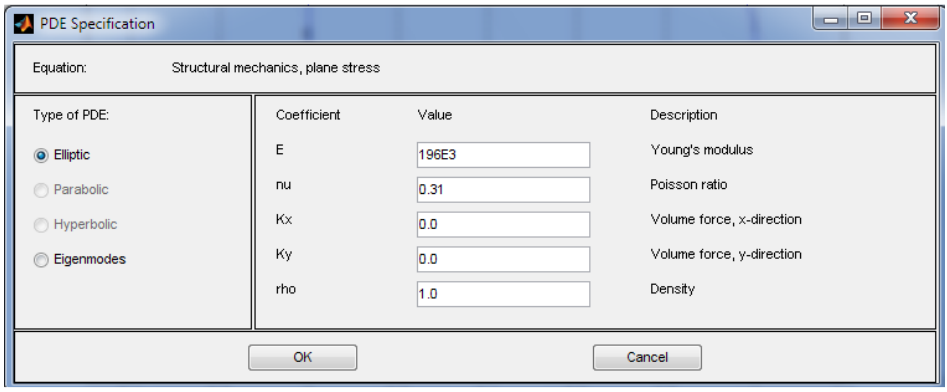
برای قسمت برش دایره ای خورده نیز شرایط نیومن را تعریف می‌کنیم:

$$g_1 = 0.5 \times nx$$

$$g_2 = 0.5 \times ny$$

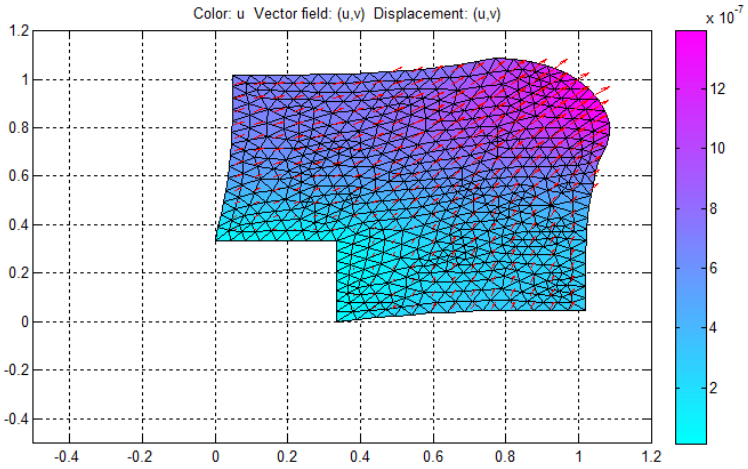
بقیه مرزها آزاد هستند، یعنی حالت بدون تنش وجود دارد. پس شرایط نیومن را در نظر می‌گیریم q=0, g=0 .

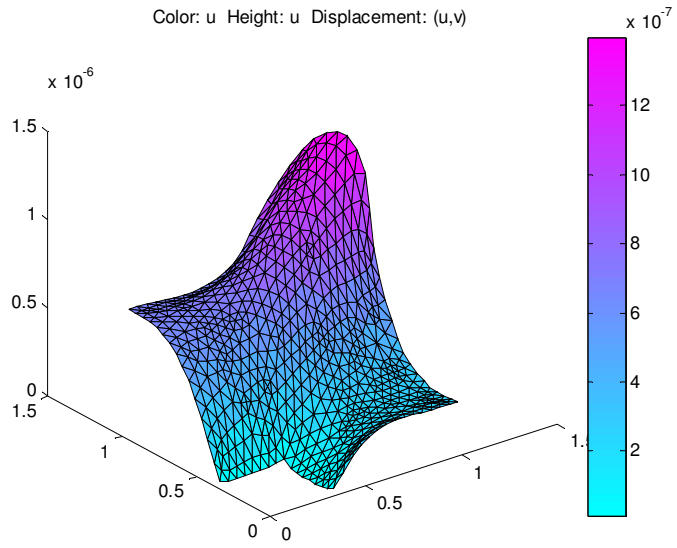
سپس از منوی PDE گزینه PDE specification را باز کرده و تنظیمات زیر را انجام دهید.



شکل(۱۱-۱۵) تنظیم پارامترهای معادله Elliptic

از منوی mesh گزینه initialize mode را انتخاب نموده و سپس روی jiggle mesh کلیک کنید تا مش بندی‌ها بهتر شوند. در پایان از منوی solve گزینه solve PDE را انتخاب کنید تا پاسخ را مشاهده کنید.





شکل(۱۱-۱۶) پاسخ معادله Elliptic

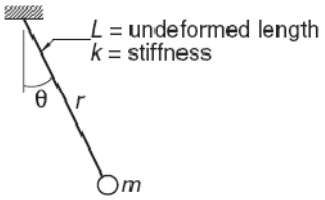
❖ تمرینات تکمیلی

۱- معادله دیفرانسیل توصیف کننده موقعیت زاویه‌ای θ بازوی مکانیکی به صورت زیر است:

$$\ddot{\theta} = \frac{a(b-\theta) - \theta\dot{\theta}^2}{1+\theta^2}$$

که در آن $a = 100s^{-2}, b = 15$ می باشند. اگر $\theta(0) = 2\pi, \theta'(0) = 0$ باشد، برنامه‌ای بنویسید که مقدار $\theta(0), \dot{\theta}(0)$ را وقتی t=0.5 sec محاسبه کند.

۲- جرمی از یک طناب الاستیک با سفتی k و طول L آویزان شده است. اگر طناب از زاویه $\theta = 60^0$ در حالتی که طناب کشیده نشده است رها شود، طول r را زمانی که $\theta = 0$ درجه است (اولین بار) به‌دست آورید. معادله دیفرانسیل حاکم بر حرکت به صورت زیر است:



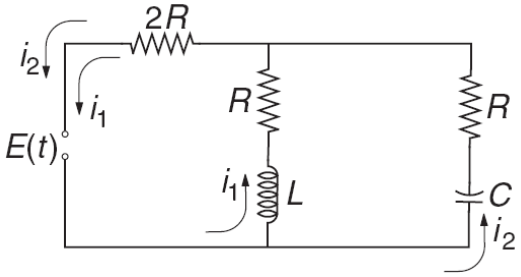
$$\ddot{r} = r\dot{\theta}^2 + g\cos\theta - \frac{k}{m}(r-L)$$

$$\ddot{\theta} = \frac{-2\dot{r}\dot{\theta} - g\sin\theta}{r}$$

که در این رابطه داریم:

M=0.25kg,L=0.5kg,k=40N/m,g=9.8m/s

۳- معادلات کیرشهف برای مدار نشان داده شده در شکل به صورت زیر است:



$$L\frac{di_1}{dt} + Ri_1 + 2R(i_1 + i_2) = E(t)$$

$$\frac{q_2}{C} + Ri_2 + 2R(i_1 + i_2) = E(t)$$

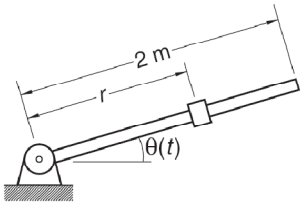
با مشتق‌گیری از رابطه بالا داریم:

$$\frac{di_1}{dt}=\frac{-3Ri_1-2Ri_2+E(t)}{L}$$
$$\frac{di_2}{dt}=\frac{-2}{3}\frac{di_1}{dt}-\frac{i_2}{3RC}+\frac{1}{3R}\frac{dE}{dt}$$

معادله اول را در معادله دوم قرار داده و $\frac{di_2}{dt}=f\ (t,i_1,i_2)$ را به‌دست آورید. فرض کنید منبع ولتاژ در لحظه t=0 روشن می شود، سپس جریان حلقه ها یعنی i_1,i_2 را در بازه زمانی $0\le t\le 0.05$ رسم کنید.

فرض کنید: $L=0.2\times10^{-3}H,\ C=3.5\times10^{-3}F,R=1\Omega,$ E(t)=240×sin(120πt)

۴- سیستم نشان داده شده در شکل زیر شامل یک جرم لغزان و یک میله راهنماست که جرم در r=0.75 قرار گرفته است. در لحظه t=0 موتور روشن شده و θ با رابطه $\theta(t)=\frac{\pi}{12}(\cos\pi t)$ تغییر می کند.

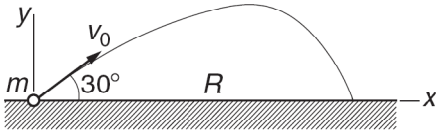


معادله دیفرانسیل توصیف کننده حرکت لغزنده به صورت زیر است(g=9.8):

$$\ddot{r}=(\frac{\pi^2}{12})^2 r\ \sin^2\pi t-g\ \sin(\frac{\pi}{12}\cos\pi t)$$

مدت زمانی را که جرم لغزان به نوک میله می رسد به‌دست آورید.

۵- یک توپ به جرم m=0.25kg با سرعت $v_0=50\frac{m}{s}$ در جهت نشان داده شده در شکل پرتاب می شود.



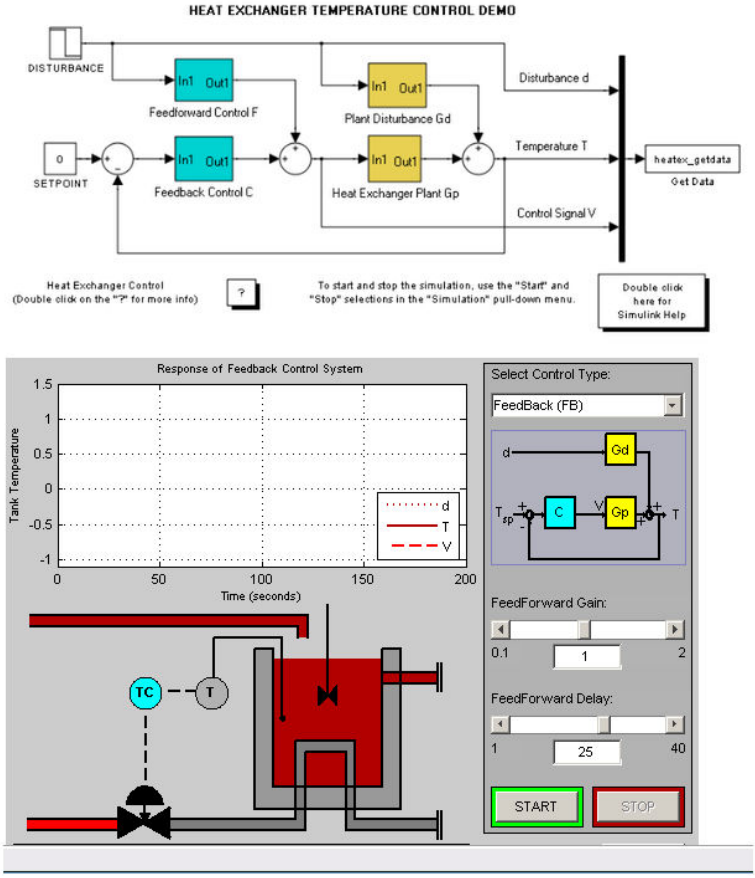
اگر نیروی آیرودینامیکی درگ روی توپ $F_D=C_D v^{\frac{3}{2}}$ باشد، معادله دیفرانسیل حرکت به صورت زیر خواهد بود:

$$\ddot{x}=-\frac{C_D}{m}\dot{x}v^{1/2}\qquad \ddot{y}=-\frac{C_D}{m}\dot{y}v^{1/2}-g$$

که در این رابطه $v=\sqrt{\dot{x}^2+\dot{y}^2}$ است. مدت زمانی را که طول می‌کشد تا توپ به زمین برخورد کند و همچنین مقدار برد توپ یعنی R را به‌دست آورید. در این مساله فرض کنید $C_D=0.03, g=9.8$ می-باشند.

قسمت دوم

- ✓ **فصل اول** مدل‌سازی و ساده‌سازی سیستم‌ها و مفهوم خطای حالت ماندگار
- ✓ **فصل دوم** تحلیل سیستم‌های کنترلی با استفاده از ترسیم مکان مرکزی ریشه‌ها و روش پاسخ فرکانسی
- ✓ **فصل سوم** طراحی و تحلیل سیستم‌های کنترلی در فضای حالت و دیجیتال



در طراحی هر سیستم کنترلی لازم است که مدل ریاضی سیستم کنترلی، معیارهای عملکردی سیستم‌های کنترلی در حوزه زمان مشخص شوند، تا طراح بتواند با این دانسته‌ها به طراحی سیستم کنترلی مناسبی برای سیستم تحت کنترل بپردازد. در این قسمت به مباحثی همچون مدل‌سازی سیستم‌ها، بررسی معیارهای عملکردی سیستم‌های کنترلی، ساده سازی بلوک دیاگرام‌ها، اصول طراحی کنترلرهای PID، فیدبک کلیه متغیرهای حالت، مفهوم فضای حالت، طراحی کنترلرهای دیجیتال، طراحی جبران‌کننده‌ها، طراحی به روش پاسخ فرکانسی، و مکان هندسی ریشه‌ها پرداخته خواهد شد.

فصل اول

مدل‌سازی و ساده‌سازی سیستم‌ها و مفهوم خطای حالت ماندگار

۱– ۱ مدل‌سازی ریاضی سیستم‌های دینامیکی

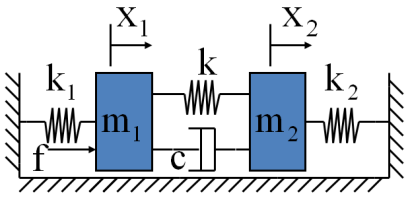
در بسیاری از روشهای مرسوم برای طراحی سیستمهای کنترلی، در دست داشتن یک مدل ریاضی نسبتاً دقیق از رفتار دینامیکی سیستم تحت کنترل الزامی است. در واقع اولین قدم در طراحی سیستم کنترل برای یک سیستم، مدل سازی آن سیستم توسط معادلات ریاضی است. بنابراین، به‌دست آوردن مدل ریاضی سیستم که نمایانگر مشخصه‌های آن است، از اهمیت فوق العاده‌ای برخوردار می‌باشد.[۱]

اغلب سیستم‌های کنترل فیدبک علاوه بر اجزای الکتریکی دارای اجزای مکانیکی، هیدرولیکی، و یا فرآیندهای ترمودینامیکی و امثال آنها نیز هستند. از نقطه نظر ریاضی، توصیف بعضی از اجزای الکتریکی و مکانیکی به طور مشابه انجام می‌شود. در واقع می‌توان نشان داد که برای اغلب مدارهای الکتریکی یک همتای مکانیکی با رفتار دینامیکی مشابه وجود دارد. به دست آوردن یک مدل کامل برای رفتار دینامیکی یک سیستم واقعی نه ممکن است و نه لزوماً مفید. در واقع توصیف نسبتاً کامل عملکرد و رفتار یک سیستم فیزیکی پیچیده بر حسب معادلات ریاضی، معمولاً به تعداد زیادی از روابط ریاضی خطی و غیرخطی منتهی می‌شود، که به علت پیچیدگی بیش از اندازه برای تحلیل و طراحی سیستمهای کنترلی مناسب نخواهد بود. از طرفی مدل ریاضی نباید بیش از حد ساده انگارانه باشد. ساده سازی بیش از اندازه موجب می‌شود که بعضی از مشخصه‌های رفتاری مهم سیستم در مدل آن وارد نشود. بدیهی است که تحلیل انجام شده براساس یک مدل نامناسب، نتایج مفید و قابل اعتمادی را به دست نخواهد داد.

معادلات مربوط به یک سیستم مکانیکی خطی، نخست با مدل‌سازی زیر سیستم‌های آن انجام می‌شود. این زیر سیستم‌ها شامل اعضایی هستند که با فرض انحرافات کوچک از نقطه تعادل، به صورت خطی قابل تقریب ریاضی هستند. رسم نمودار آزاد سیستم و اعمال قانون حرکت نیوتن به آن منتهی به مجموعه‌ای از معادلات دیفرانسیل خطی می‌شود که این مجموعه معادلات را در اصطلاح مدل ریاضی سیستم می‌گوئیم. حرکت اجزای مکانیکی را در ابعاد مختلف به صورت انتقالی، دورانی، یا ترکیبی از این دو می‌توان توصیف کرد.

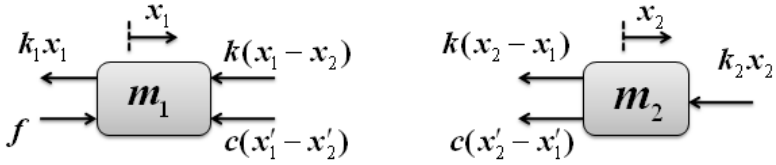
مثال: برای سیستم مکانیکی زیر نمودار آزاد را کشیده و معادلات سیستم را براساس قانون نیوتن استخراج کنید. ورودی سیستم نیروی f و خروجی متغیر x_2 می‌باشد.

۱– در بعضی دیگر از روش‌های کنترلی، مانند روش فازی، ممکن است به جای مدل ریاضی دقیق، از مدلهای ضمنی و ذهنی حاصل از سعی و خطا و یا مبتنی بر تجربه خبرگان استفاده نمود.



شکل (۱-۱) سیستم مکانیکی همراه با جرم، فنر و دمپر

پاسخ: نمودار آزاد برای دو جرم در شکل‌های زیر نمایش داده شده است و جهت‌های مبنا برای جابه‌جایی‌های x_1, x_2 مشخص شده اند. معادلات نیوتن برای سیستم مستقیماً از روی نمودارآزاد نوشته می‌شوند.



$$\sum F = m_1\ddot{x}_1 \Rightarrow \quad m_1\ddot{x}_1 = f - k_1x_1 - k_1(x_1 - x_2) - c(\dot{x}_1 - \dot{x}_2)$$

$$\sum F = m_2\ddot{x}_2 \Rightarrow \quad m_2\,\ddot{x}_2 = -k_2x_2 - k(x_2 - x_1) - c(\dot{x}_2 - \dot{x}_1)$$

از دو معادله دیفرانسیل بالا با فرض شرایط اولیه صفر تبدیل لاپلاس می‌گیریم:

$$m_1.s^2.x_1(s) = f - k_1x_1(s) - k_1(x_1(s) - x_2(s)) - c(s\,x_1(s) - s.x_2(s))$$

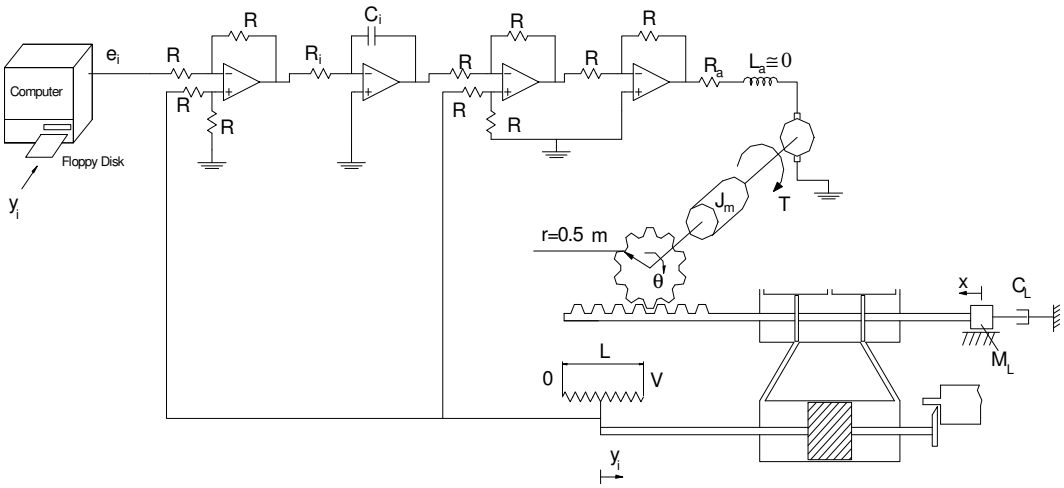
$$m_2.s^2.x_2(s) = -k_2x_2(s) - k(x_2(s) - x_1(s)) - c(s.x_2(s) - s.x_1(s))$$

باحذف $x_1(s)$ از دو معادله بالا به رابطه زیر می‌رسیم:

$$\frac{x_2(s)}{f(s)} = \frac{cs + k}{m_1m_2s^4 + c(m_1 + m_2)s^3 + [m_1k_2 + m_2k_1 + k(m_1 + m_2)]s^2 + c(k_1 + k_2)s + k_1k_2 + k(k_1 + k_2)}$$

اغلب سیستم‌های کنترلی واقعی دارای اجزای گوناگون بسیاری مانند قطعات الکترونیکی، مکانیکی، هیدرولیکی، نیوماتیکی، گرمایی و... می‌باشند. مثال زیر چنین سیستمی را نشان می‌دهد.

مثال: سیستم کنترل NC که نمایی از یک سیستم کنترل واقعی بوده و ترکیبی از قطعات الکترونیکی، مکانیکی و هیدرولیکی است، نمایش داده شده است. تابع تبدیل این سیستم را به‌دست آورید. توجه کنید که، ورودی سیستم Y_i (اطلاعات نرم افزاری متناظر با فرمان ولتاژ e_i) و خروجی Y_O (جابه‌جایی میله پیستون) می‌باشد.



شکل (۱-۲) سیستم کنترل NC

پاسخ: برای موتور الکتریکی DC می‌توان روابط زیر را نوشت:

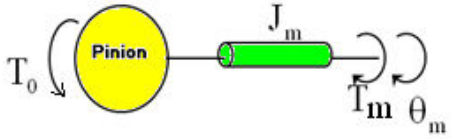
$$R_a i_a + L_a \frac{di_a}{dt} = e_a - e_b = e_a - k_b \frac{d\theta_m}{dt} \quad (۱)$$

$$T_m = K i_a \quad (۲)$$

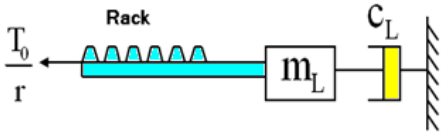
اثر تمام تقویت کننده‌های عملیاتی در هم ضرب شده و در کل در متغیری به نام K_T خود را در بلوک نمودار کنترلی نشان می‌دهند.

$$e_a = K_T (e_i - e_o) \quad (۳)$$
$$e_i = K_i y_i$$

در مسائل کنترل حرکت غالباً لازم است که حرکت دورانی به حرکت انتقالی تبدیل شود. در شکل بالا به این منظور از یک دنده شانه‌ای و پینیون به عنوان رابط مکانیکی استفاده شده است.



$$\sum T = J\ddot{\theta}_m \rightarrow T_m - T_0 = J_m \ddot{\theta}_m \quad (۴)$$



$$\begin{cases} T_0 = r \times F_{rack} \Rightarrow F_{rack} = \dfrac{T_0}{r} \\ x = r\theta_m \end{cases} \qquad (۵)$$

حال قانون نیوتن را برای جرم m_L می‌نویسیم:

$$\sum F = m_L \ddot{x} \;\Rightarrow\; \frac{T_0}{r} - c_L \dot{x} = m_L \ddot{x} \qquad (۶)$$

$$(4),(6) \rightarrow \begin{cases} T_m = J_m \ddot{\theta}_m + T_0 \\ \dfrac{T_0}{r} = m_L \ddot{x} + c_L \dot{x} \end{cases} \qquad (۷)$$

$$\Rightarrow T_m = J_m \ddot{\theta}_m + m_L r \ddot{x} + c_L r \dot{x} = (J_m + m_L r^2)\ddot{\theta}_m + c_L r^2 \dot{\theta}_m$$

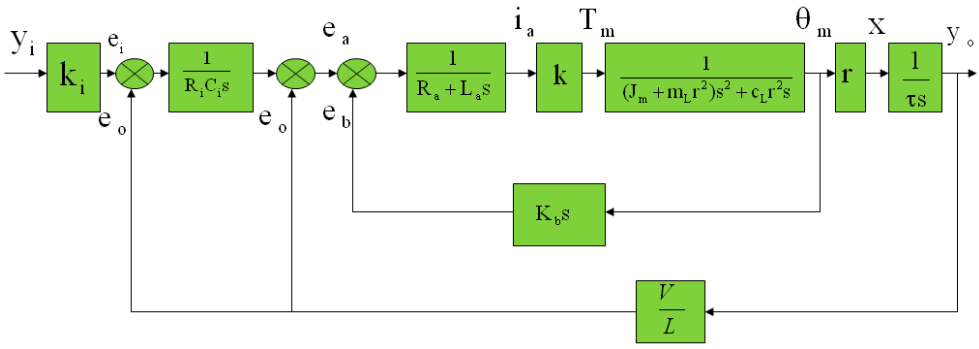
$$\theta_m(s) = \frac{T_m(s)}{(J_m + m_L r^2)s^2 + c_L r^2 s} \qquad (۸)$$

برای قسمت هیدرولیکی نیز می‌توان نوشت:

$$\begin{cases} Y_o(s) = \dfrac{1}{\tau s} X(s) \\ X(s) = r\theta_m(s) \end{cases} \rightarrow Y_o(s) = \frac{r}{\tau s}\theta_m(s) \qquad (۹)$$

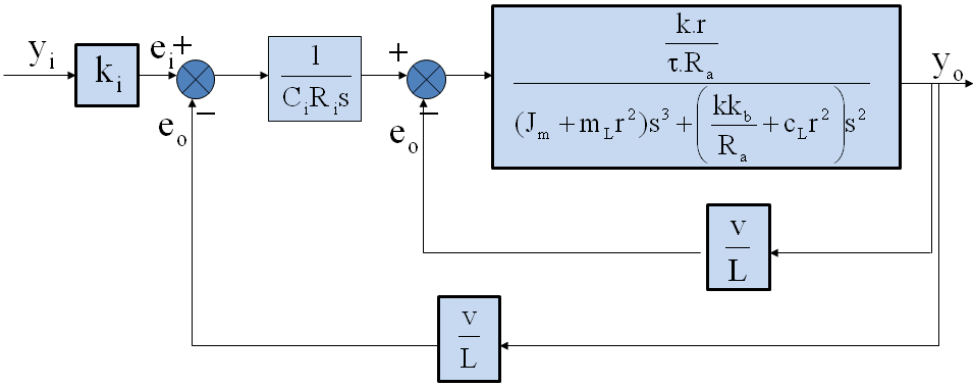
برای قسمت الکتریکی انتهای مدار می‌توان نوشت:

$$\frac{e}{v} = \frac{y_o}{L} \Rightarrow \quad e_o(s) = \frac{V}{L} Y_o(s) \qquad (۱۰)$$



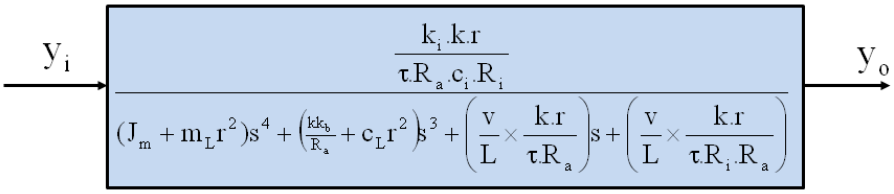
با فرض $R_a \cong 0$ می‌توان حلقه داخلی را ساده کرده و به صورت یک بلوک درآورد:

$$\frac{G}{1+GH} = \frac{\dfrac{k/R_a}{\left(J_m + m_L r^2\right)s^2 + c_L r^2 s}}{1 + \dfrac{k k_b . s/R_a}{\left(J_m + m_L r^2\right)s^2 + c_L r^2 s}} = \frac{k/R_a}{\left(J_m + m_L r^2\right)s^2 + \left(\dfrac{k.k_b}{R_a} + c_L r^2\right)s} \qquad (۱۱)$$

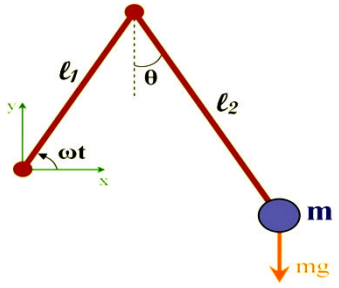


و با فرض $R_i C_i = \tau_i$ تابع تبدیل سیستم به صورت زیر به‌دست می‌آید:

$$\frac{Y_o(s)}{Y_i(s)} = \frac{\frac{k_i.k.r}{\tau.R_a.\tau_i}}{\left(J_m + m_L r^2\right)s^4 + \left(\frac{kk_b}{R_a} + c_L r^2\right)s^3 + \left(\frac{(V/L)k.r}{\tau.R_a}\right)s + \left(\frac{(V/L).k.r}{\tau.\tau_i.R_a}\right)} \qquad (۱۲)$$



مثال: پاندولی به جرم **m** و طول $\mathbf{l_2}$ به بازویی به طول $\mathbf{l_1}$ که با سرعت زاویه ای ثابت $\boldsymbol{\omega}$ در حال چرخش است، متصل شده است. معادله حرکت پاندول را به دست آورید.



شکل (۱-۳) پاندول با پایه چرخان

پاسخ: ابتدا مختصات جرم m را به‌دست می‌آوریم:

$$x = l_1\, Cos\,\omega t + l_2\, Sin\theta \rightarrow \dot{x} = -l_1\omega\, Sin\,\omega t + l_2\dot{\theta}\, Cos\theta \qquad (۱)$$
$$y = l_1\, Sin\,\omega t - l_2\, Cos\theta \rightarrow \dot{y} = l_1\omega\, Cos\,\omega t + l_2\dot{\theta}\, Sin\,\theta$$

$$v^2 = \dot{x}^2 + \dot{y}^2 = {l_1}^2\omega^2 + {l_2}^2\dot{\theta}^2 + 2l_1l_2\,\omega\dot{\theta}\, Sin(\theta - \omega t) \qquad (۲)$$

انرژی جنبشی T و انرژی پتانسیل U برای جرم m به صورت زیر به‌دست می‌آید:

$$T=\frac{1}{2}m\,v^2=\frac{m}{2}\left[l_1{}^2\omega^2+l_2{}^2\dot{\theta}^2+2l_1l_2\,\omega\dot{\theta}\,Sin(\theta-\omega t)\right]$$
$$U=m\,g\,h=m\,g\,l_1\,Sin\,\omega t-m\,g\,l_2\,Cos\,\theta \qquad (۳)$$

تابع لاگرانژ L، به صورت زیر قابل محاسبه است:

$$L=T-U$$
$$L(t,\theta,\dot{\theta})=\frac{m}{2}\left[l_1{}^2\omega^2+l_2{}^2\dot{\theta}^2+2l_1l_2\,\omega\dot{\theta}\,Sin(\theta-\omega t)\right]-mgl_1\,Sin\,\omega t+mgl_2\,Cos\,\theta \qquad (۴)$$

حال معادلات دیفرانسیل سیستم از رابطه زیر به دست می‌آید:

$$\frac{\partial L}{\partial\theta}-\frac{d}{dt}\left(\frac{\partial L}{\partial\dot{\theta}}\right)=0 \qquad (۵)$$

$$\frac{\partial L}{\partial\dot{\theta}}=ml_2{}^2\dot{\theta}+ml_1l_2\,\omega\,Sin(\theta-\omega t)\Rightarrow\frac{d}{dt}\left(\frac{\partial L}{\partial\dot{\theta}}\right)=ml_2{}^2\ddot{\theta}+ml_1l_2\omega(\dot{\theta}-\omega)\,Cos(\theta-\omega t) \qquad (۶)$$

$$\frac{\partial L}{\partial\theta}=ml_1l_2\,\omega\dot{\theta}\,Cos(\theta-\omega t)-mgl_2\,Sin\,\theta \qquad (۷)$$

$$\rightarrow\ddot{\theta}+\frac{g}{l_2}\,Sin\,\theta-\frac{l_1}{l_2}\omega^2\,Cos(\theta-\omega t)=0 \qquad (۸)$$

در طراحی‌های کلاسیک سیستم‌های کنترل مانند روش پاسخ فرکانسی یا مکان ریشه‌ها از مدل‌های تابع تبدیل[1] در حوزه s استفاده می‌شود. این مدل ها سیستم‌های فیزیکی و صنعتی یک ورودی و یک خروجی را با دقت قابل قبولی تقریب می‌زنند. تحلیل دقیق سیستم‌های صنعتی پیچیده مدل‌های کاملتری از سیستم را طلب می‌کند. مدل‌سازی سیستم‌های کنترل با استفاده از متغیرهای حالت در راستای هدف بالا است.

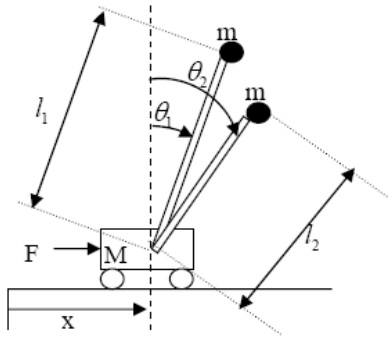
مدل تابع تبدیل تنها توصیفی از رفتار ورودی و خروجی سیستم ارائه می‌دهد و لذا آن را توصیف خارجی کرده، تنها برای سیستم‌های تغییر ناپذیر با زمان معتبر است؛ در صورتی‌که متغیرهای حالت، دینامیک داخلی سیستم را نیز توصیف می‌کنند. از اینرو به آن توصیف داخلی سیستم نیز می‌گویند و برای سیستم‌های خطی و نیز غیرخطی قابل اعمال هستند.

مثال: شکل(۱-۴) دو پاندول معکوس را نشان می‌دهد که در صفحه عمود، حول یک محور یکسان می‌توانند حرکت کنند. دو جسم متصل به انتهای هر پاندول دارای جرم برابر m هستند. نقطه تعادل سیستم به راحتی قابل تشخیص است:

$$\boldsymbol{\theta}_1=\boldsymbol{\theta}_2=\dot{\boldsymbol{\theta}}_1=\dot{\boldsymbol{\theta}}_2=\dot{x}=0,F=0,x=x^*$$

ضمن آن‌که جرم میله ها قابل صرف نظرکردن است.

[1] Transfer Function



شکل (۱-۴) دو پاندول معکوس

پاسخ: متغیرهای تعمیم یافته که حرکت سیستم را به طور کامل توصیف می‌کنند عبارتند از : x,θ_1,θ_2

برای حل معادله لاگرانژ، ابتدا باید انرژی‌های جنبشی و پتانسیل را جداگانه به‌دست آوریم:

m_1 بردار سرعت جرم $=(\dot{x}+l_1\dot{\theta}_1\cos\theta_1)\vec{i}+(l_1\dot{\theta}_1\sin\theta_1)\vec{j}$	(۱)
m_2 بردار سرعت جرم $=(\dot{x}+l_2\dot{\theta}_2\cos\theta_2)\vec{i}+(l_2\dot{\theta}_2\sin\theta_2)\vec{j}$	(۲)

انرژی جنبشی کل سیستم برابر است با:

$T=\frac{1}{2}M\dot{x}^2+\frac{1}{2}m[(\dot{x}+l_1\dot{\theta}_1\cos\theta_1)^2+(l_1\dot{\theta}_1\sin\theta_1)^2]+\frac{1}{2}m[(\dot{x}+l_2\dot{\theta}_2\cos\theta_2)^2+(l_2\dot{\theta}_2\sin\theta_2)^2]$	(۳)
$T=\frac{1}{2}(M+m)\dot{x}^2+\frac{1}{2}m(l_1^2\dot{\theta}_1^2+l_2^2\dot{\theta}_2^2)+m\dot{x}(l_1\dot{\theta}_1\cos\theta_1+l_2\dot{\theta}_2\cos\theta_2)$	(۴)

از طرفی انرژی پتانسیل تنها در اثر تغییر ارتفاع در جرم پاندول ایجاد می‌شود. اگر سطح انرژی پتانسیل ارابه را $U_o=0$ در نظر بگیریم، انرژی پتانسیل مجموعه عبارت است از:

$U=mgl_1\cos\theta_1+mgl_2\cos\theta_2=mg(l_1\cos\theta_1+l_2\cos\theta_2)$	(۵)

در این مرحله، به محاسبه تابع لاگرانژین که تفاضل انرژی‌های جنبشی و پتانسیل می‌باشد، می‌پردازیم. تابع لاگرانژ سیستم عبارت است از:

$L=T-U$ $L=\frac{1}{2}(M+m)\dot{x}^2+\frac{1}{2}m(l_1^2\dot{\theta}_1^2+l_2^2\dot{\theta}_2^2)+m\left[l_1\cos\theta_1\left(\dot{x}\dot{\theta}_1-g\right)+l_2\cos\theta_2\left(\dot{x}\dot{\theta}_2-g\right)\right]$	(۶)

نیروی تعمیم یافته در جهت x عبارت است از F ولی به علت عدم تحریک مستقل پاندول، نیروی تعمیم یافته‌ای در جهت θ_1,θ_2 وجود نخواهد داشت. حال معادله لاگرانژ را در امتداد x,θ می‌نویسیم:

$\frac{d}{dt}\left(\frac{\partial L}{\partial\dot{q}}\right)-\frac{\partial L}{\partial q}=Q \qquad q=\begin{bmatrix}x\\ \theta_1\\ \theta_2\end{bmatrix} \quad Q=\begin{bmatrix}F\\ 0\\ 0\end{bmatrix}$	(۷)
$for\ x: \quad \frac{d}{dt}\left(\frac{\partial L}{\partial\dot{x}}\right)-\frac{\partial L}{\partial x}=F$ $\Rightarrow(M+2m)\ddot{x}+m[l_1\ddot{\theta}_1\cos\theta_1-l_1\dot{\theta}_1^2\sin\theta_1+l_2\ddot{\theta}_2\cos\theta_2-l_2\dot{\theta}_2^2\sin\theta_2]=F$	(۸)

$$for\ \boldsymbol{\theta}_1 : \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\boldsymbol{\theta}}_1}\right) - \frac{\partial L}{\partial \boldsymbol{\theta}_1} = 0 \quad \Rightarrow l_1\ddot{\boldsymbol{\theta}}_1 + \ddot{x}Cos\boldsymbol{\theta}_1 - gSin\boldsymbol{\theta}_1 = 0 \tag{۹}$$

$$for : \boldsymbol{\theta}_2 : \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\boldsymbol{\theta}}_2}\right) - \frac{\partial L}{\partial \boldsymbol{\theta}_2} = 0 \quad \Rightarrow l_2\ddot{\boldsymbol{\theta}}_2 + \ddot{x}Cos\boldsymbol{\theta}_2 - gSin\boldsymbol{\theta}_2 = 0 \tag{۱۰}$$

بعد از عملیات خطی‌سازی حول نقطه تعادل فوق الذکر($Sin\boldsymbol{\theta}_2 \simeq \boldsymbol{\theta}_2, Sin\boldsymbol{\theta}_1 \simeq \boldsymbol{\theta}_1$) داریم:

$$\begin{bmatrix} M+2m & ml_1 & ml_2 \\ 1 & l_1 & 0 \\ 1 & 0 & l_2 \end{bmatrix}\begin{bmatrix} \ddot{x} \\ \ddot{\boldsymbol{\theta}}_1 \\ \ddot{\boldsymbol{\theta}}_2 \end{bmatrix} = \begin{bmatrix} F \\ g\boldsymbol{\theta}_1 \\ g\boldsymbol{\theta}_2 \end{bmatrix} \Rightarrow \begin{bmatrix} \ddot{x} \\ \ddot{\boldsymbol{\theta}}_1 \\ \ddot{\boldsymbol{\theta}}_2 \end{bmatrix} = \begin{bmatrix} M+2m & ml_1 & ml_2 \\ 1 & l_1 & 0 \\ 1 & 0 & l_2 \end{bmatrix}^{-1}\begin{bmatrix} F \\ g\boldsymbol{\theta}_1 \\ g\boldsymbol{\theta}_2 \end{bmatrix} = \begin{bmatrix} \ddot{x}_2 \\ \ddot{x}_4 \\ \ddot{x}_6 \end{bmatrix} \tag{۱۱}$$

برای این سیستم می‌توان شش متغیر حالت تعریف کرد:

$$\dot{x} = Ax + Bu \Rightarrow \begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{\boldsymbol{\theta}}_1 \\ \ddot{\boldsymbol{\theta}}_1 \\ \dot{\boldsymbol{\theta}}_2 \\ \ddot{\boldsymbol{\theta}}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{-gm}{M} & 0 & \frac{-gm}{M} & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{g(m+M)}{Ml_1} & 0 & \frac{mg}{Ml_1} & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & \frac{mg}{l_2M} & 0 & \frac{g(m+M)}{Ml_2} & 0 \end{bmatrix}\cdot\begin{bmatrix} x \\ \dot{x} \\ \boldsymbol{\theta}_1 \\ \dot{\boldsymbol{\theta}}_1 \\ \boldsymbol{\theta}_2 \\ \dot{\boldsymbol{\theta}}_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \\ 0 \\ \frac{-1}{l_1M} \\ 0 \\ \frac{-1}{l_2M} \end{bmatrix}.F \tag{۱۲}$$

برای مسأله دو پاندول معکوس فرض کنید $m\ell_1 = 1.5$ و $m\ell_2 = 1$. با توجه به اینکه خروجی سیستم چه متغیری باشد، می‌توان توابع تبدیل x/F و θ_1/F و θ_2/F را تعریف کرد:

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -9.81 & 0 & -9.81 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 13.08 & 0 & 6.54 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 9.81 & 0 & 19.62 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ 1 \\ 0 \\ -0.67 \\ 0 \\ -1 \end{bmatrix}$$

$$For : \frac{x}{F} \rightarrow C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$For : \frac{\boldsymbol{\theta}_1}{F} \rightarrow C = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

$$For : \frac{\boldsymbol{\theta}_2}{F} \rightarrow C = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

$$Transfer\ Function = H(s) = C(sI - A)^{-1}.B$$

- تابع تبدیل $H_1(s)$

$$for\frac{x}{F}\rightarrow H_1(s)=\begin{bmatrix}1&0&0&0&0&0\end{bmatrix}.\begin{bmatrix}s&-1&0&0&0&0\\0&s&9.81&0&9.81&0\\0&0&s&-1&0&0\\0&0&-13.08&s&-6.54&0\\0&0&0&0&s&-1\\0&0&-9.81&0&-19.62&s\end{bmatrix}^{-1}.\begin{bmatrix}0\\1\\0\\-0.67\\0\\-1\end{bmatrix}$$

$$T.F.for\frac{x}{F}\rightarrow H_1(s)=\frac{s^4-16.32s^2+63.84}{s^6-32.70s^4+192.47s^2} \quad (۱۳)$$

- تابع تبدیل $H_2(s)$

$$for\frac{\theta_1}{F}\rightarrow H_2(s)=\begin{bmatrix}0&0&1&0&0&0\end{bmatrix}.\begin{bmatrix}s&-1&0&0&0&0\\0&s&9.81&0&9.81&0\\0&0&s&-1&0&0\\0&0&-13.08&s&-6.54&0\\0&0&0&0&s&-1\\0&0&-9.81&0&-19.62&s\end{bmatrix}^{-1}.\begin{bmatrix}0\\1\\0\\-0.67\\0\\-1\end{bmatrix}$$

$$T.F.for\frac{\theta_1}{F}\rightarrow H_2(s)=\frac{-0.67s^4+6.61s^2}{s^6-32.70s^4+192.47s^2} \quad (۱۴)$$

- تابع تبدیل $H_3(s)$

$$for\frac{\theta_2}{F}\rightarrow H_3(s)=\begin{bmatrix}0&0&0&0&1&0\end{bmatrix}.\begin{bmatrix}s&-1&0&0&0&0\\0&s&9.81&0&9.81&0\\0&0&s&-1&0&0\\0&0&-13.08&s&-6.54&0\\0&0&0&0&s&-1\\0&0&-9.81&0&-19.62&s\end{bmatrix}^{-1}.\begin{bmatrix}0\\1\\0\\-0.67\\0\\-1\end{bmatrix}$$

$$T.F.for\frac{\theta_2}{F}\rightarrow H_3(s)=\frac{-s^4+6.51s^2}{s^6-32.70s^4+192.47s^2} \quad (۱۵)$$

۱– ۲ تعاریف معیارهای عملکردی سیستم‌های کنترلی در حوزه زمان

۱-۲-۱ حداکثر جهش[1]

یکی از مشخصه‌های عملکردی یک سیستم کنترلی در حوزه زمان حداکثر جهش آن می‌باشد. در واقع جهش بسیار زیاد مطلوب نیست، زیرا باعث کاهش میزان پایداری سیستم می‌گردد. اگر حداکثر مقدار خروجی سیستم

[1] Maximum Overshoot

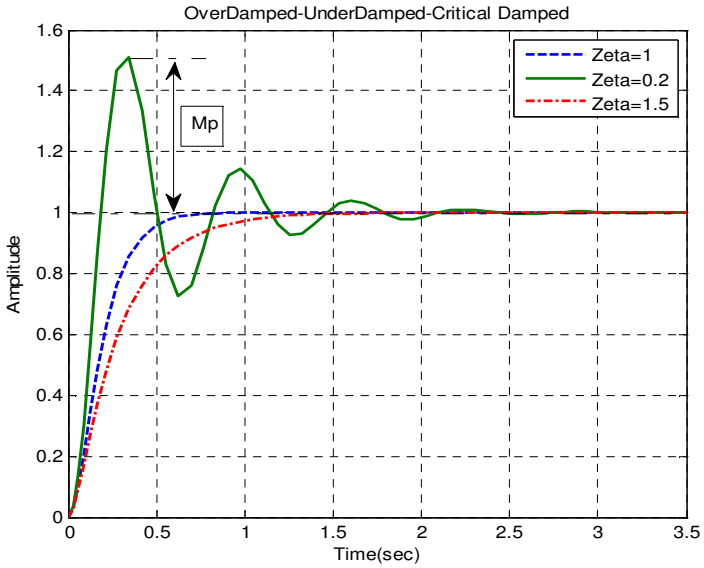
$C(t)$ پس از اعمال ورودی پله را با C_{Max} نشان دهیم و C_{ss} نیز مقدار حالت ماندگار آن بوده و $C_{Max} \geq C_{ss}$ باشد، آنگاه حداکثر جهش را به صورت زیر تعریف می‌کنیم:

$$\text{حداکثر جهش} = M_p = C_{Max} - C_{ss}$$

در بسیاری موارد حداکثر جهش را به صورت درصدی از مقدار حالت ماندگار پاسخ پله سیستم نمایش می‌دهند.

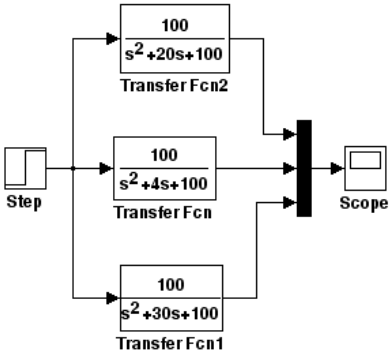
$$\text{درصد حداکثر جهش} = \frac{C_{Max} - C_{ss}}{C_{ss}} \times 100\%$$

شکل زیر پاسخ یک سیستم مرتبه دوم $G(s) = \dfrac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}$ ، به ورودی پله را نمایش می‌دهد.



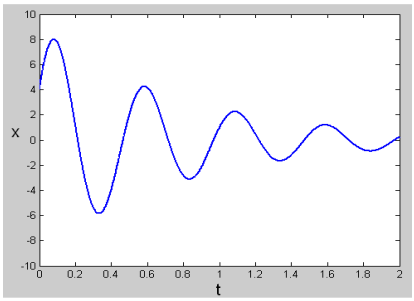
شکل (۱-۵) پاسخ پله سیستم مرتبه دوم

برای گرفتن خروجی بالا، شبیه سازی در محیط سیمولینک به صورت زیر انجام می‌شود.

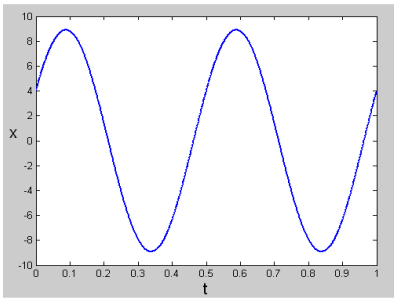


در شکل بالا M_p حداکثر جهش است که با رابطه $M_p = e^{-\frac{\pi\zeta}{\sqrt{1-\zeta^2}}}$ به‌دست می‌آید . در این رابطه ζ نسبت میرایی[1] است. با توجه به مقدار ζ داریم :

- اگر $\zeta = 0$، در آن صورت سیستم ، بدون میرایی[2] خواهد بود و پاسخ به صورت شکل (۱-۶) می‌شود.
- اگر $0 < \zeta < 1$، در آن صورت سیستم با میرایی زیر بحرانی[3] خواهد بود و پاسخ به صورت شکل(۱-۷) می‌شود.

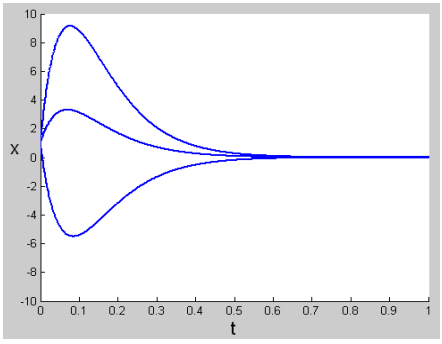


شکل(۱-۶) سیستم بدون میرایی

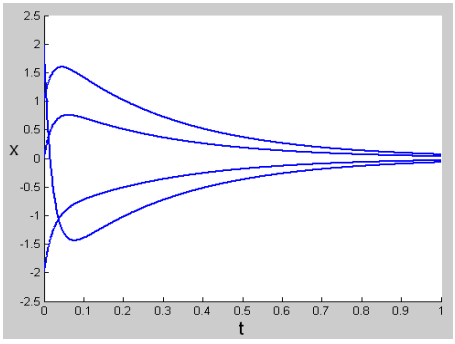


شکل(۱-۷) سیستم با میرایی زیر بحرانی

- اگر $\zeta = 1$، درآن صورت سیستم با میرایی بحرانی[4] خواهد بود و پاسخ به صورت شکل (۱-۸) می‌شود.
- اگر $\zeta > 1$، در آن صورت سیستم با میرایی فوق بحرانی[5] خواهد بود و پاسخ به صورت شکل (۱-۹) می‌شود



شکل(۱-۸) سیستم با میرایی بحرانی



شکل(۱-۹) سیستم با میرایی فوق بحرانی یا شدید

۱-۲-۲ زمان خیز[6]

زمان خیز به زمانی گفته می‌شود که لازم است تا پاسخ از 10 درصد به 90 درصد (یا از 5 به 95 درصد یا از 0 به 100 درصد) مقدار نهایی‌اش برسد. برای سیستم‌هایی با میرایی ضعیف معمولاً از 0 تا 100 درصد و برای

[1] Damping Ratio
[2] Undamped System
[3] Underdamped System
[4] Critically damped System
[5] Overdamped System
[6] Rise Time

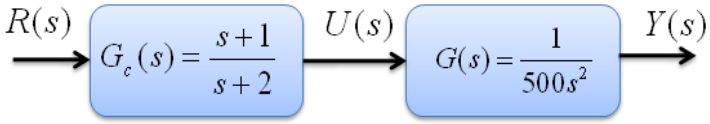
سیستم‌های با میرایی شدید اغلب از 10 تا 90 درصد استفاده می‌شود . زمان خیز را می‌توان با رابطه $T_r = \frac{1.8}{\omega_n}$ به‌دست آورد.

۱-۲-۳ زمان نشست[1]

زمان نشست به مدت زمانی گفته می‌شود که طی آن، پاسخ سیستم به یک محدوده مفروض در نزدیکی مقدار نهایی اش برسد و پس از آن نیز در آن محدوده باقی بماند. این محدوده، عموماً بر حسب درصدی از مقدار نهایی پاسخ مانند 2٪ یا 5٪ آن بیان می‌شود. برای رسیدن به محدوده 2٪، زمان نشست برابر 4τ و برای محدوده 5٪ برابر 3τ است. در اغلب کتابها این مقدار را با رابطه $T_s = \frac{4.6}{\zeta\omega_n}$ به‌دست می‌آورند. در این کتاب نیز ما از این رابطه استفاده می‌کنیم.

۱– ۳ ساده سازی بلوک نمودارها

فرض کنید تابع تبدیل یک سیستم تحت کنترل $G(s) = \frac{1}{500s^2}$ و تابع تبدیل کنترل کننده به صورت $G_c(s) = \frac{s+1}{s+2}$ باشد، که به صورت سری با هم قرار گرفته اند. با استفاده از دستور Series می‌توان این دو تابع تبدیل را در هم ضرب نموده و یک سیستم معادل به دست آورد.



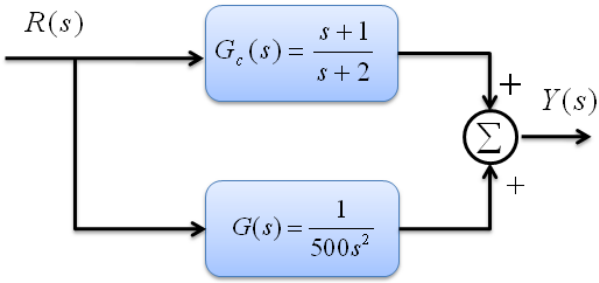
$$T(s) = G_c(s).G(s) = \frac{s+1}{s+2} \times \frac{1}{500s^2} = \frac{s+1}{500s^3 + 1000s^2}$$

```
numg=[1];
deng=[500 0 0];
numh=[1 1];
denh=[1 2];
[num,den]=series(numg,deng,numh,denh);
printsys(num,den)
```

```
>> num/den =
          s + 1
   ------------------
   500 s^3 + 1000 s^2
```

حال فرض کنید دو تابع تبدیل $G_1(s), G_2(s)$ به صورت موازی با هم قرار گرفته اند. در آن صورت تابع تبدیل سیستم معادل به صورت زیر به‌دست می‌آید:

[1] Settling Time

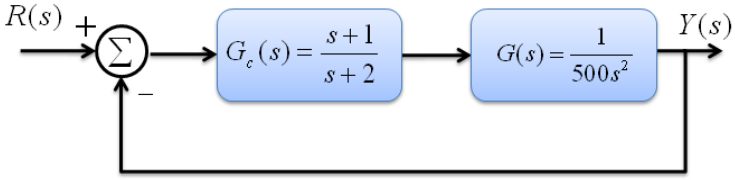


$$T(s)=G_c(s)+G(s)=\frac{s+1}{s+2}+\frac{1}{500s^2}=\frac{500s^3+500s^2+s+2}{500s^3+1000s^2}$$

```
numg=[1];
deng=[500 0 0];
numh=[1 1];
denh=[1 2];
[num,den]=parallel(numg,deng,numh,denh);
printsys(num,den)
```

```
>> num/den =

    500 s^3 + 500 s^2 +  s + 2
    --------------------------
        500 3 + 1000 s^2
```

سیستم با فیدبک واحد مثبت یا منفی زیر را در نظر بگیرید. تابع تبدیل سیستم حلقه بسته زیر برابر است با:

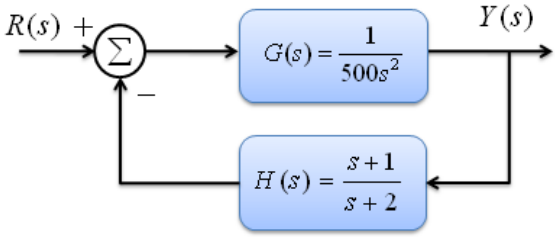


$$T(s)=\frac{G_cG(s)}{1+G_cG(s)}=\frac{\frac{s+1}{s+2}\times\frac{1}{500s^2}}{1+\frac{s+1}{s+2}\times\frac{1}{500s^2}}=\frac{s+1}{500s^3+1000s^2+s+1}$$

```
numg=[1];
deng=[500 0 0];
numh=[1 1];
denh=[1 2];
[num1,den1]=series(numg,deng,numh,denh);
[num,den]=cloop(num1,den1,-1)
printsys(num,den)
```

```
>> num/den =
                s + 1
    --------------------------
    500 s^3 + 1000 s^2 +  s + 1
```

حال به جای فیدبک واحد، تابع تبدیل H(s) را قرار دهید. در این حالت تابع تبدیل سیستم حلقه بسته برابر خواهد بود با:



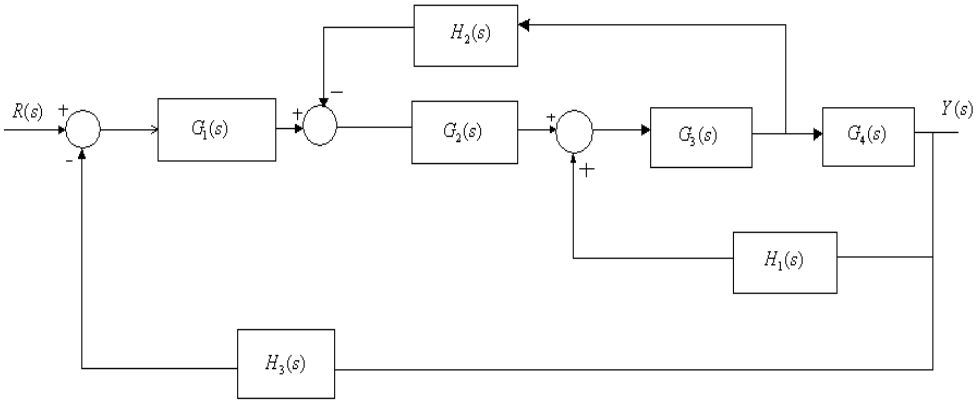
$$T(s)=\frac{G(s)}{1+H(s).G(s)}=\frac{\frac{1}{500s^2}}{1+\frac{s+1}{s+2}\times\frac{1}{500s^2}}=\frac{s+1}{500s^3+1000s^2+s+1}$$

```
numg=[1];
deng=[500 0 0];
numh=[1 1];
denh=[1 2];
[num,den]=feedback(numg,deng,numh,denh);
printsys(num,den)
```

مثال: در سیستم چند حلقه‌ای زیر، تابع تبدیل سیستم حلقه بسته $T(s)=\frac{Y(s)}{R(s)}$ را به‌دست آورید. در این سیستم هر یک از توابع تبدیل به صورت زیر می‌باشند:

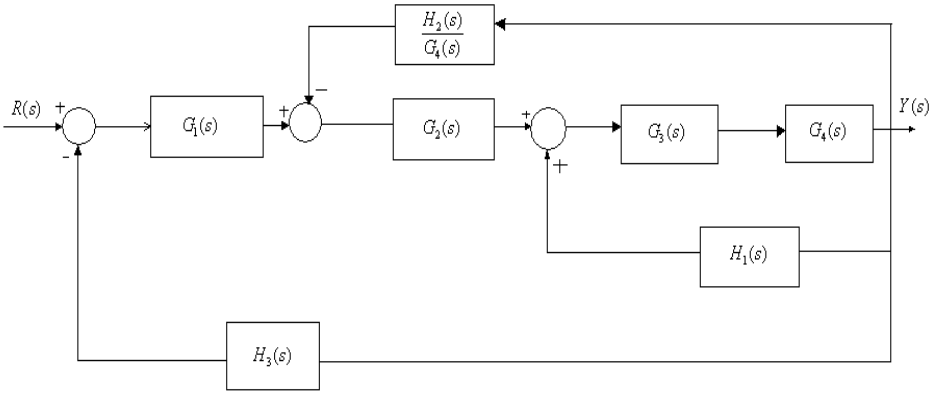
$$G_1(s)=\frac{1}{s+10}\quad G_2(s)=\frac{1}{s+1}\quad G_3(s)=\frac{s^2+1}{s^2+4s+4}\quad G_4(s)=\frac{s+1}{s+6}$$

$$H_1(s)=\frac{s+1}{s+2}\quad H_2(s)=2\quad H_3(s)=1$$



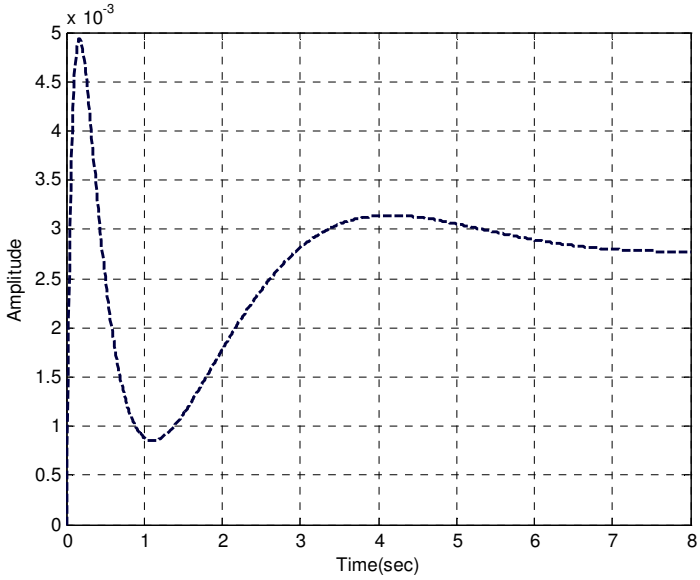
شکل(۱-۱۰) بلوک نمودار سیستم حلقه بسته

ابتدا $H_2(s)$ را به پشت $G_4(s)$ انتقال می‌دهیم تا به بلوک نمودار زیر تبدیل شود. سپس با استفاده از دستورهای Feedback، Cloop، وSeries تابع تبدیل سیستم حلقه بسته را به‌دست می‌آوریم.




```
clear all
clc
ng1=[1];dg1=[1 10];
ng2=[1];dg2=[1 1];
ng3=[1 0 1];dg3=[1 4 4];
ng4=[1 1];dg4=[1 6];
nh1=[1 1];dh1=[1 2];
nh2=[2];dh2=[1];
nh3=[1];dh3=[1];

n1=conv(nh2,dg4);d1=conv(dh2,ng4);
[n2a,d2a]=series(ng3,dg3,ng4,dg4);
[n2,d2]=feedback(n2a,d2a,nh1,dh1,+1);
[n3a,d3a]=series(ng2,dg2,n2,d2);
[n3,d3]=feedback(n3a,d3a,n1,d1,-1);
[n4,d4]=series(ng1,dg1,n3,d3);
[num,den]=cloop(n4,d4,-1)

printsys (num,den)
t=0:0.01:8;
[y,x,t]=step(num,den,t);
Plot(t,y)
xlabel('Time(sec)')
ylabel('Amplitude')
```

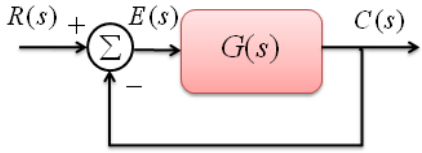
```
>> num/den =

                  s^5 + 4 s^4 + 6 s^3 + 6 s^2 + 5 s + 2
  ---------------------------------------------------------------
  12 s^6 + 205 s^5 + 1066 s^4 + 2517 s^3 + 3128 s^2 + 2196 s + 712
```



شکل(۱-۱۱) پاسخ سیستم حلقه بسته به ورودی پله

۱ – ۴ مفهوم خطای حالت ماندگار

قبل از بررسی خطای حالت ماندگار به بررسی مفهوم *نوع سیستم* می‌پردازیم. برای بیان مفهوم نوع سیستم، ساختار فیدبک واحد نشان داده شده در شکل (۱-۱۲) مورد استفاده قرار گرفته و نوع سیستم را با توجه به تابع تبدیل حلقه باز آن تعریف می‌کنند.



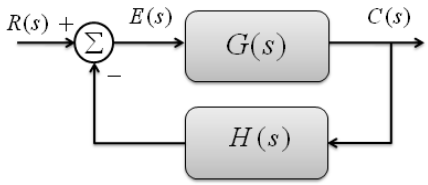
شکل(۱-۱۲) ساختار فیدبک واحد

در حالت کلی می‌توان $G(s)$ را به شکل زیر بیان کرد:

$$G(s)=\frac{K(s+z_1)(s+z_2)\ldots}{s^N(s+p_1)(s+p_2)\ldots} \qquad (۱-۱)$$

ریشه‌های صورت و مخرج را به ترتیب صفرها و قطب‌های سیستم می‌نامیم. K بهره سیستم و N=0, 1, 2,… نوع سیستم را مشخص می‌کند. بنابراین N=0 یک سیستم نوع صفر، N=1 یک سیستم نوع یک و... را نشان می‌دهد.

مثال: اگر در بلوک نمودار زیر $G(s)=\frac{1}{s(s+1)}$ و $H(s)=\frac{3}{(s+3)}$ باشد، نوع و مرتبه سیستم را تعیین کنید.



برای تعیین نوع سیستم، نخست لازم است که آن را به صورت فیدبک واحد تبدیل کنیم. بنابراین:

$$\frac{C(s)}{R(s)}=\frac{G(s)}{1+G(s)H(s)}=\frac{s+3}{s^3+4s^2+3s+3}=\frac{A(s)}{B(s)}$$

$$\frac{A(s)}{B(s)}=\frac{G_{eq}(s)}{1+G_{eq}(s)}\Rightarrow \quad G_{eq}(s)=\frac{A(s)}{B(s)-A(s)}=\frac{s+3}{s(s^2+4s+2)}$$

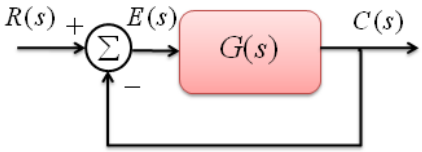
از این رو سیستم نوع یک و از مرتبه سوم است.

نکته: با فرمول زیر به راحتی می‌توانیم تابع تبدیل حلقه باز معادل را از روی سیستم فیدبک غیرواحد بالا به‌دست آوریم.

$$\text{تابع تبدیل حلقه باز معادل}=\frac{G(s)}{1+G(s)[H(s)-1]}$$

۱-۷-۱ سیستم‌های نوع صفر

فرض کنید که یک سیستم مرتبه ۲ نوع صفر به صورت زیر داریم که در آن $G(s)$ برابر است با :



$$G(s)=\frac{1}{(s+2)(s+3)}$$

در این قسمت با به‌کارگیری سیگنال‌های استاندارد و اعمال آنها به سیستم‌های کنترلی به بررسی پاسخ سیستم می‌پردازیم. خروجی سیستم از دو بخش تشکیل می‌شود: قسمت اول عکس العمل سیستم به ورودی اعمال شده، در زمان‌های اولیه را پاسخ گذرای[۱] سیستم می‌نامند. قسمت دوم، عکس العمل ماندگار سیستم است. عکس العمل ماندگار سیستم را پاسخ حالت ماندگار [۲] سیستم می‌نامند. پاسخ حالت ماندگار سیستم کنترل بسیار مهم است زیرا نشان می‌دهد که با گذشت زمان پاسخ سیستم به چه سمتی میل می‌کند. اگر پاسخ حالت ماندگار با ورودی فرمان داده شده به سیستم دقیقاً مطابقت نداشته باشد سیستم دارای خطای حالت ماندگار است.

توجه: اغتشاش وارد شده به سیستم می‌تواند در عملکرد حالت گذرا و ماندگار سیستم تاثیر زیادی داشته باشد. ورودی یک سیستم صنعتی در حین کار نوعاً غیر قابل پیش بینی است. مثلاً در یک سیستم دنبال کننده رادار موقعیت و سرعت هدف به طور کاملاً تصادفی تغییر می‌کند. حال با استفاده از سیگنال‌های فرمان استاندارد تا حد زیادی می‌توان رفتار سیستم را به ورودی‌های ناشناس دیگر حدس زد.

حال به پاسخ سیستم، وقتی ورودی‌های مختلف به آن اعمال می‌شود، توجه کنید.

۱-۱-۴-۱ ورودی پله

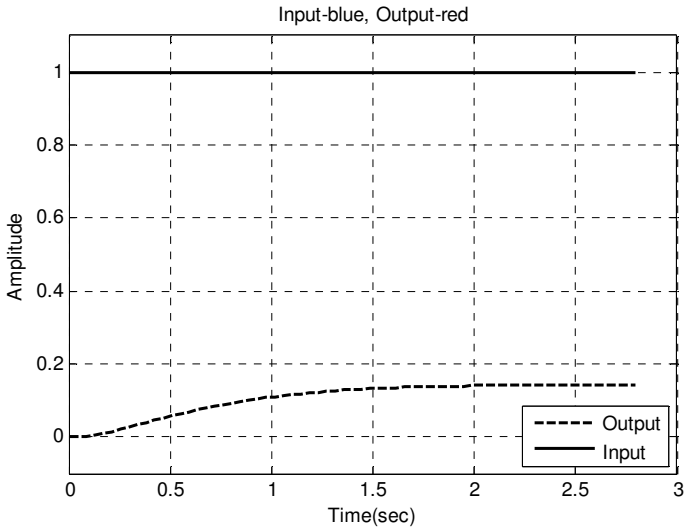
این ورودی ساده ترین نوع ورودی است. این ورودی را با اعمال یک ولتاژ ثابت در یک مدار الکتریکی و یا یک چرخش ناگهانی و ثابت در یک محور مکانیکی می‌توان قیاس نمود. اگر اندازه دامنه ورودی واحد باشد آن را پله واحد می‌نامند. پله غیر واحد با دامنه R (یک ثابت حقیقی) به این صورت است:

$$u(t)=\begin{cases}R & t\geq 0\\ 0 & t<0\end{cases}$$

```
num = 1;den = conv([1 2],[1 3]);
sys = tf(num,den);sys_cl = feedback(sys,1);
[y,t] = step(sys_cl);
u = ones(size(t));
plot(t,y,'y',t,u,'g')
```

[۱] Transient Response
[۲] Steady-state Response

```
axis([0,3,0,1.1])
xlabel('Time(secs)');ylabel('Amplitude')
title('Input-blue, Output-red')
```



شکل(۱-۱۳) پاسخ سیستم نوع صفر به ورودی پله

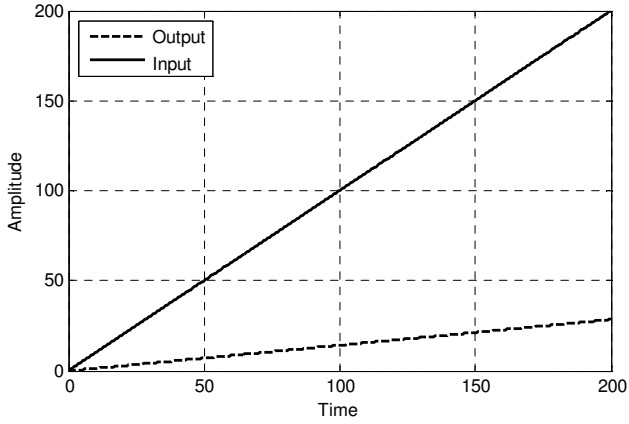
در این مثال، مقدار خطای حالت ماندگار ثابت است.

۱-۴-۱-۲ ورودی شیب دار یا مرتبه یک

اگر از ورودی پله‌ای انتگرال گرفته شود سیگنالی به‌دست خواهد آمد که دامنه آن یک تغییر ثابت نسبت به زمان خواهد داشت. ورودی شیب دار را ورودی سرعت نیز می‌نامند.

$$u(t)=\begin{cases} Rt & t\geq 0 \\ 0 & t<0 \end{cases}$$

```
num = 1;
den = conv([1 2],[1 3]);
sys = tf(num,den);
sys_cl = feedback(sys,1);
t = 0:0.1:200;
u = t;
[y,t,x] = lsim(sys_cl,u,t);
plot(t,y,'r',t,u,'b')
```



شکل(۱-۱۴) پاسخ سیستم نوع صفر به ورودی شیب دار

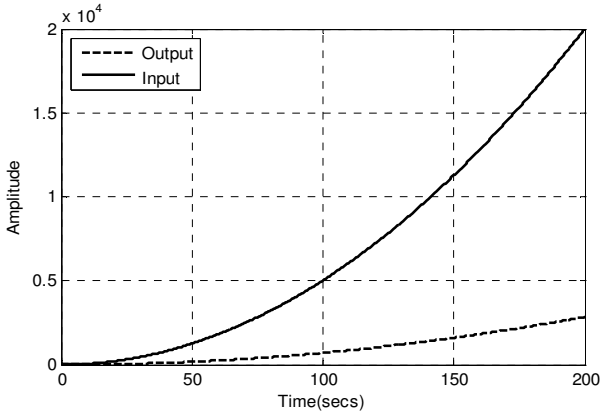
در این مثال، مقدار خطای حالت ماندگار برابر بینهایت است.

۱-۴-۱-۳ ورودی سهموی یا مرتبه دو

ورودی سهموی با انتگرال از تابع شیب به‌دست می‌آید. ورودی سهموی را ورودی شتاب نیز می‌نامند. نمایش ریاضی آن به صورت زیر است:

$$u(t)=\begin{cases}\dfrac{Rt^2}{2} & t\geq 0\\ 0 & t<0\end{cases}$$

```
num = 1;den = conv([1 2],[1 3])
sys = tf(num,den);
sys_cl = feedback(sys,1);
t = 0:0.1:200;
u = 0.5*t.*t;
[y,t,x] = lsim(sys_cl,u,t);
plot(t,y,'r',t,u,'b')
xlabel('Time(secs)');ylabel('Amplitude')
title('Input-blue, Output-red')
```

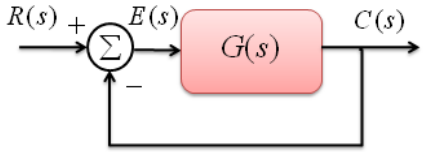


شکل(۱-۱۵) پاسخ سیستم نوع صفر به ورودی سهموی

در این مثال، مقدار خطای حالت ماندگار برابر بینهایت است.

۱-۴-۲ سیستم‌های نوع یک

فرض کنید که یک سیستم مرتبه ۳ نوع یک به صورت زیر داریم که در آن $G(s)$ برابر است با :

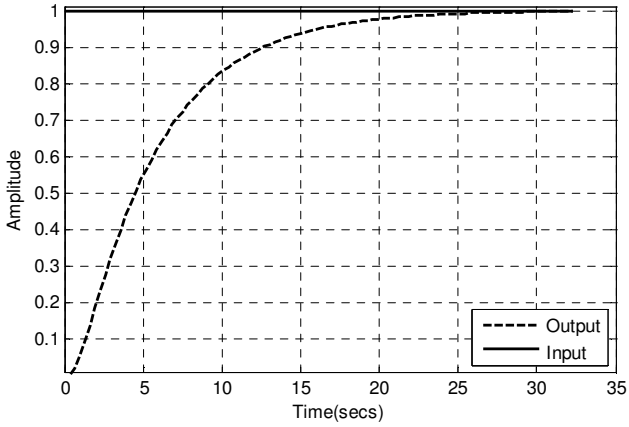


$$G(s)=\frac{1}{s(s+2)(s+3)}$$

حال به پاسخ سیستم وقتی که ورودی‌های مختلف به آن اعمال می‌شود توجه کنید:

۱-۴-۲-۱ ورودی پله

```
num = 1;den = conv([1 2],[1 3]);
den = conv(den,[1 0]);
sys = tf(num,den); sys_cl = feedback(sys,1);
[y,t] = step(sys_cl);
u = ones(size(t));
plot(t,y,'r',t,u,'b')
```



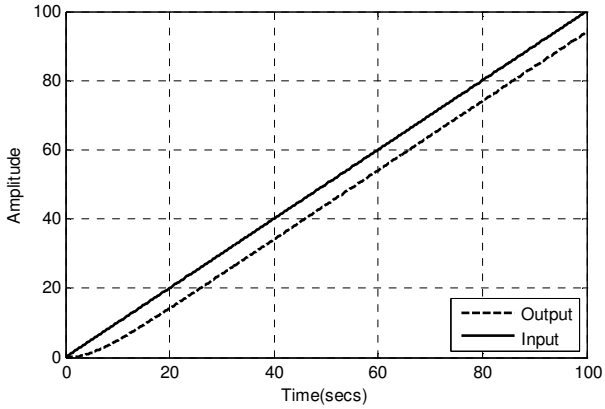
شکل(۱-۱۶) پاسخ سیستم نوع یک به ورودی پله

یعنی مقدار خطای حالت ماندگار صفر است.

۱-۴-۲-۲ ورودی شیب دار یا مرتبه یک

```
num = 1;den = conv([1 2],[1 3]);den = conv(den,[1 0]);
sys = tf(num,den);sys_cl = feedback(sys,1);
t = 0:0.1:100;
u = t;
[y,t,x] = lsim(sys_cl,u,t);
```

```
plot(t,y,'b',t,u,'r')
xlabel('Time(secs)'); ylabel('Amplitude')
```

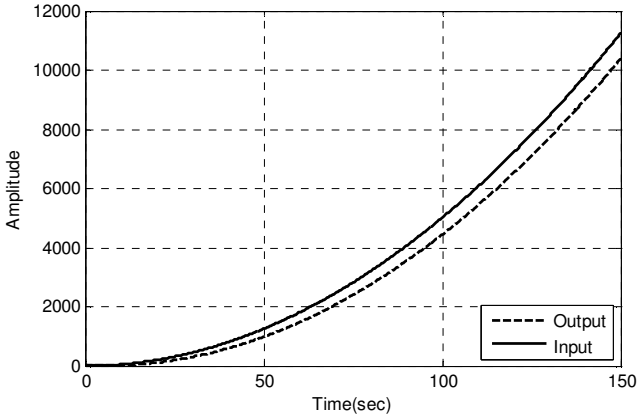


شکل(۱-۱۷) پاسخ سیستم نوع یک به ورودی شیب دار

در این جا، مقدار خطای حالت ماندگار ثابت است.

۱-۴-۲-۳ ورودی سهموی یا مرتبه دو

```
num = 1;den = conv([1 2],[1 3]);den = conv(den,[1 0]);
sys = tf(num,den);
sys_cl = feedback(sys,1);
t = 0:0.1:150;
u = 0.5*t.*t;
[y,t,x] = lsim(sys_cl,u,t);
plot(t,y,'b',t,u,'m')
xlabel('Time(secs)');ylabel('Amplitude')
title('Input-purple, Output-blue')
```

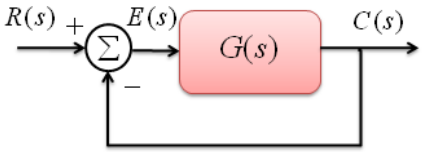


شکل(۱-۱۸) پاسخ سیستم نوع یک به ورودی سهموی

در این حالت، مقدار خطای حالت ماندگار برابر بینهایت است.

۱-۴-۳ سیستم‌های نوع دو

فرض کنید که یک سیستم مرتبه چهار نوع دو به صورت زیر داریم که در آن G(s) برابر است با :

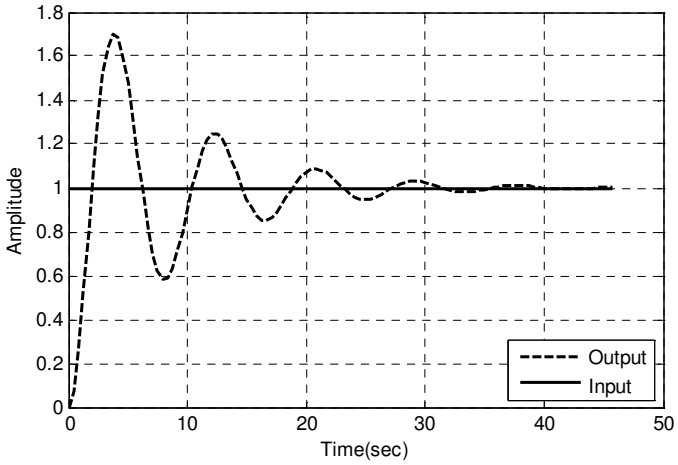


$$G(s)=\frac{(s+1)(s+3)}{s^2(s+2)(s+3)}$$

حال به پاسخ سیستم وقتی که ورودی‌های مختلف به ان اعمال می‌شود توجه کنید:

۱-۴-۳-۱ ورودی پله

```
num = conv([1 1],[1 3]);
den = conv([1 2],[1 3]);
den = conv(den,[1 0]);
den = conv(den,[1 0]);
sys = tf(num,den);sys_cl = feedback(sys,1);
[y,t] = step(sys_cl);
u = ones(size(t));
plot(t,y,'b',t,u,'r')
```



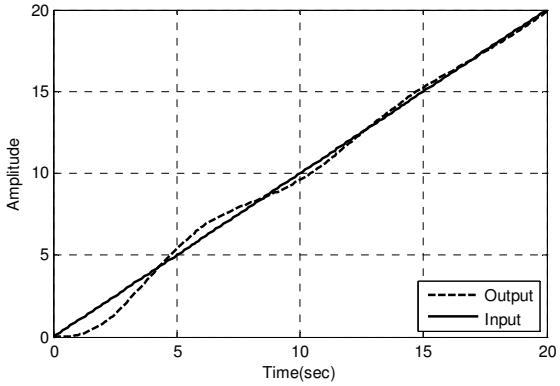
شکل(۱-۱۹) پاسخ سیستم نوع دو به ورودی پله

مقدار خطای حالت ماندگار صفر است.

۱-۴-۳-۲ ورودی شیب دار یا مرتبه یک

```
num = conv([1 1],[1 3]);
den = conv([1 2],[1 3]);
```

```
den = conv(den,[1 0]);
den = conv(den,[1 0]);
sys = tf(num,den);sys_cl = feedback(sys,1);
t = 0:0.1:50;u = t;
[y,t,x] = lsim(sys_cl,u,t);
plot(t,y,'b',t,u,'m')
xlabel('Time(secs)')
ylabel('Amplitude')
```

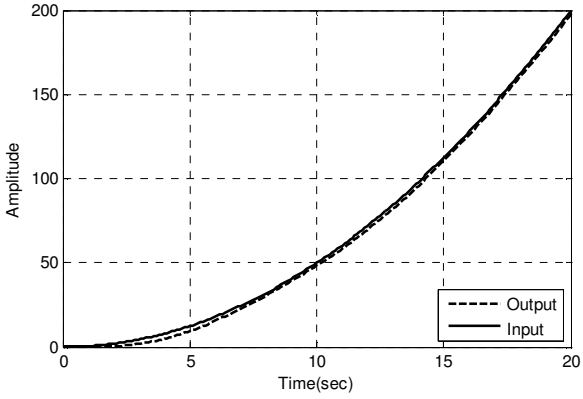


شکل(۱-۲۰) پاسخ سیستم نوع دو به ورودی شیب دار

مقدار خطای حالت ماندگار صفر است.

۱-۴-۳-۳ ورودی سهموی یا مرتبه دو

```
num = conv([1 1],[1 3]);den = conv([1 2],[1 3]);
den = conv(den,[1 0]);den = conv(den,[1 0]);
sys = tf(num,den);sys_cl = feedback(sys,1);
t = 0:0.1:20;u = 0.5*t.*t;
[y,x] = lsim(sys_cl,u,t);
plot(t,y,'b',t,u,'m')
```



شکل(۱-۲۱) پاسخ سیستم نوع دو به ورودی سهموی

مقدار خطای حالت ماندگار ثابت است.

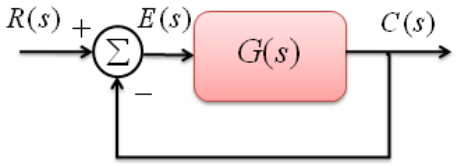
۱ – ۵ تعریف خطای حالت ماندگار

خطای حالت ماندگار عبارت است از تفاوت بین ورودی و خروجی سیستم، وقتی که زمان به سمت بینهایت میل می‌کند. مقدار خطای حالت ماندگار به ورودی سیستم و نوع سیستم (نوع صفر، یک و دو) بستگی دارد.

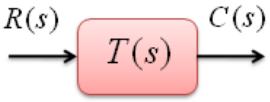
توجه کنید که خطای حالت ماندگار برای سیستم پایدار تعریف می‌شود. یعنی قبل از اینکه به بررسی مقدار خطای حالت ماندگار سیستم بپردازیم، باید پایداری سیستم را بررسی کنیم. یعنی اگر سیستم ناپایدار باشد بحث خطای حالت ماندگار بی معنی است.

محاسبه خطای حالت ماندگار: قبل از صحبت در مورد رابطه بین خطای حالت ماندگار و نوع سیستم، باید ببینیم خطای حالت ماندگار بدون در نظر گرفتن نوع سیستم و ورودی چگونه محاسبه می‌شود.

خطای حالت ماندگار را می‌توان از روی تابع تبدیل حلقه باز یا حلقه بسته با فیدبک واحد به‌دست آورد. سیستم زیر را در نظر بگیرید:



این شکل معادل سیستم زیر است:



$$E(s) = R(s) - C(s) = R(s)[1 - \frac{C(s)}{R(s)}] = \frac{R(s)}{1 + G(s)} \qquad (۲-۱)$$

مقدار خطای حالت ماندگار را برای این سیستم می‌توان از روی تابع تبدیل حلقه بسته یا حلقه باز با استفاده از قضیه مقدار نهایی به‌دست آورد. توجه کنید که زمانی می‌توانید از این تئوری استفاده کنید که مخرج تابع هیچ قطبی در سمت راست محور موهومی نداشته باشد.

$$e(\infty) = \lim_{s \to 0} \frac{s.R(s)}{1 + G(s)} \qquad (۳-۱)$$

$$e(\infty) = \lim_{s \to 0} s.R(s) \times [1 - T(s)] \qquad (۴-۱)$$

حال ورودی‌های مختلف به سیستم اعمال کرده و مقدار خطای حالت ماندگار را به‌دست می‌آوریم:

ورودی پله:

$$R(s) = \frac{1}{s} \Rightarrow e(\infty) = \frac{1}{1 + \lim_{s \to 0} G(s)} = \frac{1}{1 + K_p} \quad , K_p = \lim_{s \to 0} G(s) \qquad (۵-۱)$$

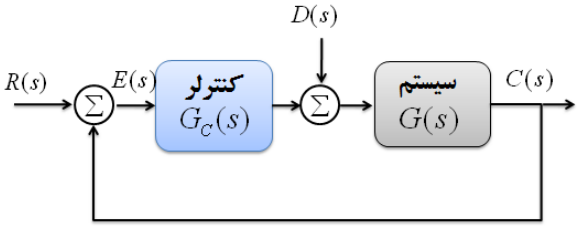
ورودی شیب دار:

$$R(s)=\frac{1}{s^2} \Rightarrow e(\infty)=\frac{1}{\lim\limits_{s\to 0} s.G(s)}=\frac{1}{K_v} \quad ,K_v=\lim_{s\to 0} s.G(s) \qquad (۶-۱)$$

ورودی سهموی:

$$R(s)=\frac{1}{s^3} \Rightarrow e(\infty)=\frac{1}{\lim\limits_{s\to 0} s^2\times G(s)}=\frac{1}{K_a} \quad ,K_a=\lim_{s\to 0} s^2\times G(s) \qquad (۷-۱)$$

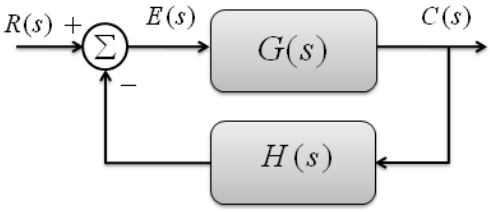
نکته: در حین طراحی یک کنترلر، معمولاً می‌خواهیم اغتشاش وارده به سیستم را نیز جبران کنیم. فرض کنید که سیستم زیر موجود است:



می‌توانیم خطای حالت ماندگار در اثر ورودی اغتشاش پله را از معادله زیر پیدا کنیم:

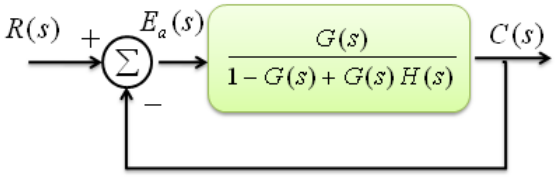
$$R(s)=\frac{1}{s} \Rightarrow e(\infty)=\frac{1}{\lim\limits_{s\to 0} G_c(s)+\lim\limits_{s\to 0}\frac{1}{G(s)}} \qquad (۸-۱)$$

نکته: برای به دست آوردن خطای حالت ماندگار تحت فیدبک غیر واحد که در شکل زیر آمده است، نیز به ترتیب زیر عمل می‌کنیم:



$$\text{تابع تبدیل حلقه باز معادل} = \frac{G(s)}{1+G(s)[H(s)-1]}$$

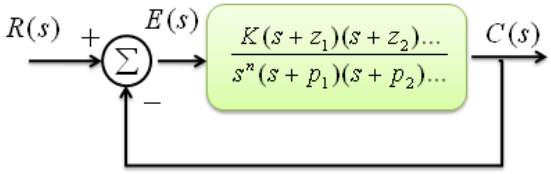
که معادل بلوک زیر است:



$$e(\infty)=\lim_{s\to 0}\frac{s.R(s)}{1+\frac{G(s)}{1+G(s).H(s)-G(s)}}=\lim_{s\to 0}\frac{s.R(s)\times(1+G(s).H(s)-G(s))}{1+G(s).H(s)} \qquad (۹-۱)$$

۱ – ۶ نوع سیستم و خطای حالت ماندگار

در قسمت‌های قبل درباره یکسری ثابت‌ها صحبت کردیم. Kp ثابت موقعیت، Kv ثابت سرعت، و Ka ثابت شتاب است. اگر مقدار این ثابت ها مشخص باشد در آن صورت می‌توانیم خطای حالت ماندگار را به‌دست آوریم. همانطور که قبلاً گفتیم، نوع سیستم یعنی اینکه چند انتگرال‌گیر خالص در سیستم وجود دارد. مثلاً در سیستم زیر n نشان دهنده نوع سیستم است. پس یک سیستم می‌تواند مرتبه صفر، یک، دو و ... باشد. در جدولهای زیر رابطه خطای حالت ماندگار و نوع سیستم نمایش داده شده است.

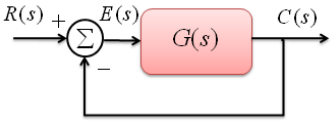


سیستم ها ی نوع صفر	ورودی پله	وردی شیب دار	ورودی سهمی
فرمول خطای حالت ماندگار	1/(1+Kp)	1/Kv	1/Ka
ثابت خطای استاتیکی	Kp = ثابت	Kv = 0	Ka = 0
خطا	1/(1+Kp)	بینهایت	بینهایت

سیستم ها ی نوع یک	ورودی پله	وردی شیب دار	ورودی سهمی
فرمول خطای حالت ماندگار	1/(1+Kp)	1/Kv	1/Ka
ثابت خطای استاتیکی	Kp = بینهایت	Kv = ثابت	Ka = 0
خطا	0	1/Kv	بینهایت

سیستم ها ی نوع دو	ورودی پله	وردی شیب دار	ورودی سهمی
فرمول خطای حالت ماندگار	1/(1+Kp)	1/Kv	1/Ka
ثابت خطای استاتیکی	Kp = بینهایت	Kv = بینهایت	Ka = ثابت
خطا	0	0	1/Ka

مثال: سیستم زیر را در نظر بگیرید :

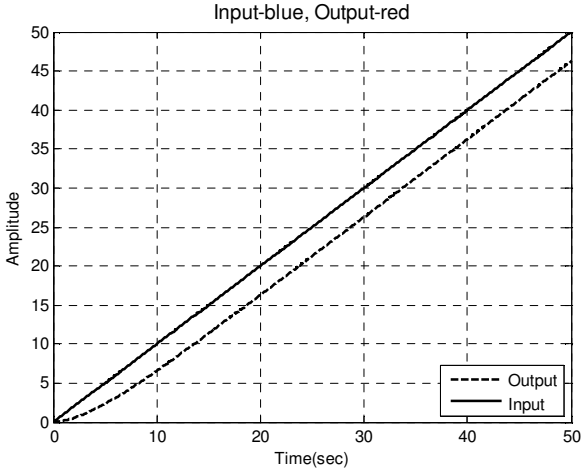


که در آن G(s) برابر است با:

$$G(s)=\frac{K.(s+5)(s+3)}{s(s+7)(s+8)}$$

مقدار K را طوری به‌دست آورید که خطای حالت ماندگار در حالت حلقه باز برابر 10% شود. از آنجایی که سیستم نوع اول است، پس مقدار خطای حالت ماندگار به ورودی پله برابر صفر، به ورودی شیب دار مقدار ثابت، و به ورودی مرتبه دوم برابر بینهایت است. پس به ورودی شیب دار با بهره 1 توجه کنید.

```
num = conv( [1 5], [1 3]);
den = conv(conv([1,7],[1 8]),[1,0]);
plant = tf(num,den);sysC = feedback(plant,1);
t = 0:0.1:50;
u = t;
[y,t,x] = lsim(sysC,u,t);
plot(t,y,'r',t,u,'b')
xlabel('Time(secs)')
ylabel('Amplitude')
title('Input-blue, Output-red')
```



شکل(۱-۲۲) پاسخ سیستم نوع یک به ورودی شیب دار

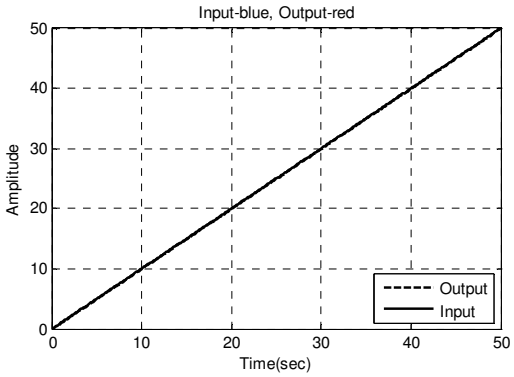
مقدار خطای حالت ماندگار بسیار زیاد است، مثلاً در زمان t =20 ثانیه مقدار خروجی برابر 16 است که نسبتاً زیاد است. می‌دانیم که خطا برابر 0.1 است، پس مساله را به صورت زیر حل می‌کنیم:

$$e(\infty)=\frac{1}{K_v}=0.1\Rightarrow K_v=\lim_{s\to 0}\ s.G(s)=10$$

$$K_v=37.33$$

```
k =37.33 ;
num = conv( [1 5], [1 3]);
den = conv(conv([1,7],[1 8]),[1,0]);
```

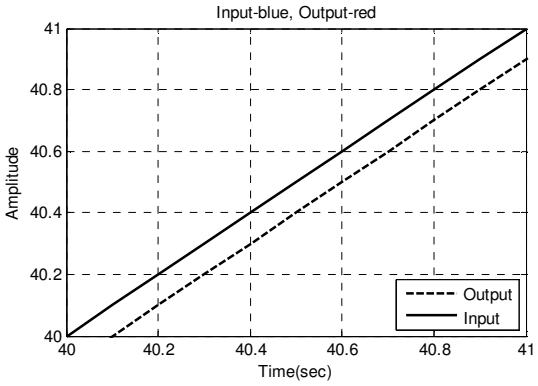
```
plant = tf(num,den);
sysC = feedback(k*plant,1);
t = 0:0.1:50;
u = t;
[y,t,x] = lsim(sysC,u,t);
plot(t,y,'r',t,u,'b')
```



شکل(۱-۲۳) پاسخ سیستم نوع یک به ورودی شیب دار

برای رسیدن به یک نمای بهتر باید بیشتر در نمودار دقت کنیم. پس محور را به صورت زیر تغییر می‌دهیم:

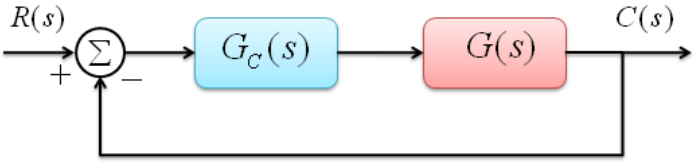
```
axis([40,41,40,41])
```



شکل(۱-۲۴) پاسخ سیستم نوع یک به ورودی شیب دار

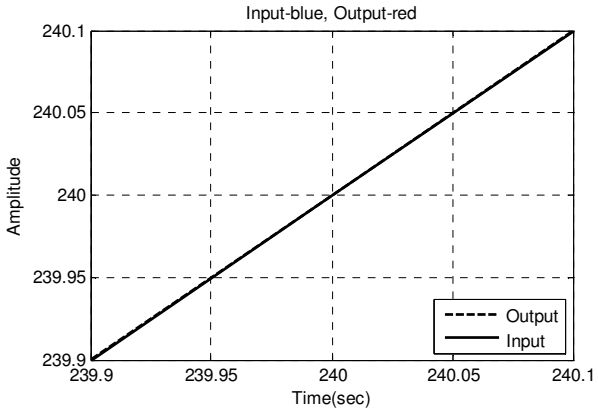
همانطور که در شکل مشاهده می‌شود فاصلهٔ بین دو خط برابر 0.1 است.

حال مساله را تا حدودی تغییر می‌دهیم و فرض می‌کنیم سیستم به صورت زیر است:



ودر آن، G(s) مشابه حالت بالا است و ما می‌خواهیم خطای حالت ماندگار برابر صفر باشد. از طرفی می‌دانیم خطای حالت ماندگار سیستم مرتبه دو به ورودی شیب دار برابر صفر است. پس برای صفر کردن خطای حالت ماندگار، کافی است که یک انتگرال‌گیر به سیستم اضافه کنیم. (یک قطب در مبداء)

```
num = conv( [1 5], [1 3]);
den = conv(conv([1,7],[1 8]),[1,0]);
plant = tf(num,den);
numc = 1; denc = [1 0];
contr = tf(numc,denc);
sysC = feedback(contr*plant,1);
t = 0:0.1:250;
u = t;
[y,t,x] = lsim(sysC,u,t);
plot(t,y,'r',t,u,'b')
axis([239.9,240.1,239.9,240.1])
```

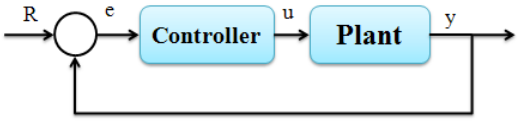


شکل(۱-۲۵) پاسخ سیستم نوع یک به ورودی شیب دار

همانطور که می‌بینید خطای حالت ماندگار برابر صفر است.

۱-۷ طراحی کنترلر PID

در این قسمت سیستم فیدبک واحد زیر را در نظر بگیرید:



شکل(۱-۲۵) سیستم به همراه کنترلر با فیدبک واحد

Plant: سیستمی که قرار است کنترل شود.

Controller: این قسمت سیستم مورد نظر ما را تحریک می‌کند و طوری عمل می‌کند که رفتار کلی سیستم را کنترل نماید.

کنترلر با سه ترم: تابع تبدیل کنترلر PID به صورت زیر می‌باشد:

$$k_p + k_D s + \frac{k_I}{s} = \frac{k_D s^2 + k_p s + k_I}{s} \qquad (۱۰-۱)$$

که در آن K_p ،گین تناسبی، K_I ،گین انتگرالی، K_d ،گین مشتق‌گیر می‌باشد. در ابتدا اجازه بدهید ببینیم سیستم حلقه بسته با کنترلر PID به چه صورت عمل می‌کند. متغیر e نشان دهنده خطای ردگیری است که از اختلاف بین ورودی مطلوب R و خروجی واقعی سیستم یعنی Y ناشی می‌شود. این سیگنال خطا وارد کنترلر PID می‌شود و کنترلر، انتگرال و مشتق خطای ورودی را می‌گیرد و در نهایت خروجی U حاصل می‌شود.

$$U = k_p e + k_D \frac{de}{dt} + k_I \int e dt \qquad (۱۱-۱)$$

حال باید به سیستم تحت کنترل خروجی U وارد شود و یک خروجی جدید Y گرفته شده، و خطای e دوباره محاسبه شود و این روند به همین صورت ادامه پیدا کند.

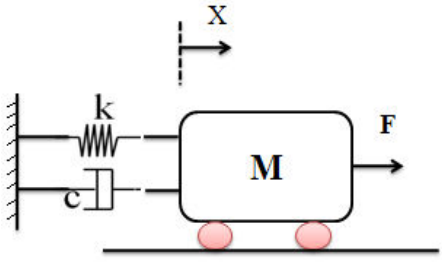
۱-۷-۲ خصوصیات کنترلرهای P,I,D

کنترلر تناسبی K_p زمان خیز را کاهش داده، و خطای حالت ماندگار را کاهش می‌دهد ولی آن را حذف نمی‌کند. کنترلر انتگرالی K_I خطای حالت ماندگار را حذف می‌کند ولی پاسخ حالت گذرا را بدتر می‌کند.کنترلر مشتق گیر K_d پایداری سیستم را افزایش داده، مقدار جهش را کاهش می‌دهد و پاسخ حالت گذرا را بهتر می‌کند.
تاثیر هر یک از کنترلرهای تناسبی، مشتق‌گیر، و انتگرال‌گیر را در جدول زیر مشاهده می‌کنید:

نوع کنترلر	زمان خیز	جهش	زمان نشست	خطای حالت ماندگار
تناسبی	کاهش	افزایش	تغییر اندک	کاهش
انتگرال گیر	کاهش	افزایش	افزایش	حذف
مشتق گیر	تغییر اندک	کاهش	کاهش	تغییر اندک

توجه کنید که این روابط کاملاً دقیق نیستند. از آنجا که مقادیر k_p, k_D, k_I به یکدیگر وابسته هستند، تغییر یکی از این متغیرها روی دو تای دیگر تاثیر می‌گذارد. به همین دلیل این جدول به عنوان یک مرجع زمانی استفاده می‌شود که بخواهیم مقادیر k_p, k_D, k_I را تعیین کنیم.

مثال: یک جرم و فنر و دمپر ساده را در نظر بگیرید.



شکل(۱-۲۶) سیستم جرم و فنر و دمپر

پاسخ: بعد از مدل‌سازی معادله زیر حاصل می‌شود:

$$M\ddot{x}+b\dot{x}+kx=F \qquad (۱)$$

بعد از لاپلاس‌گیری از معادله بالا داریم:

$$Ms^2.x(s)+bs.x(s)+k.x(s)=F(s) \qquad (۲)$$

تابع تبدیل بین ورودی $F(s)$ و جابه جایی $x(s)$ برابر است با:

$$\frac{x(s)}{F(s)}=\frac{1}{Ms^2+bs+k} \qquad (۳)$$

اجازه بدهید مقادیر زیر را برای پارامترها در نظر بگیریم:

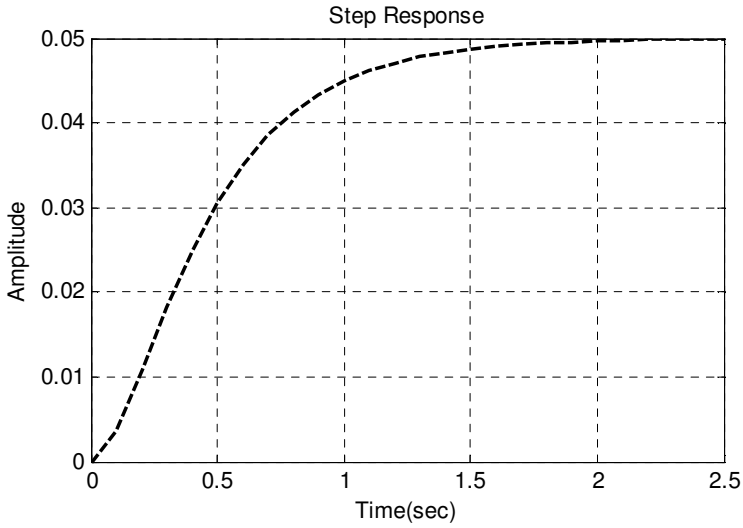
$$F=1N, M=1, b=10\ N.s/m,\ k=20N/m$$

هدف از این مساله این است که تاثیر هر سه متغیررا بررسی کنیم تا زمان خیز سریع، حداقل جهش، و خطای حالت ماندگار صفر را داشته باشیم.

پاسخ پله سیستم حلقه باز: در ابتدا پاسخ پله سیستم حلقه باز را بررسی می‌کنیم.پس دستورات زیر را در یک m-file وارد کنید:

```
num=1;
den=[1 10 20];
step(num,den)
```

بعد از اجرای برنامه نمودار شکل (۱-۲۶) حاصل می‌شود.



شکل(۱-۲۶) پاسخ پله سیستم حلقه باز

مقدار نهایی خروجی به ورودی پله واحد برابر 0.05 است یعنی مقدار خطای حالت ماندگار برابر 95% بوده و این مقدار بسیار زیاد است. زمان خیز حدود 1 ثانیه و زمان نشست حدود 1.5 ثانیه است. حال اجازه بدهید کنترلری طراحی‌کنیم که مقدار زمان خیز و زمان نشست را کاهش داده، و خطای حالت ماندگار را حذف کند.

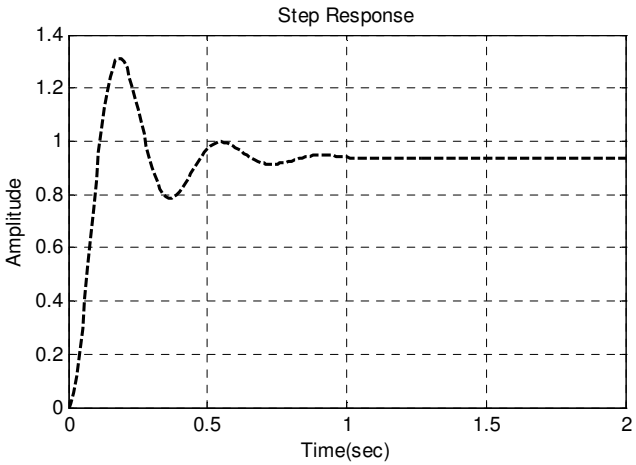
کنترلر تناسبی: از جدول بالا، می‌بینیم که کنترلر تناسبی مقدار زمان خیز و زمان نشست را کاهش داده و مقدار جهش را افزایش می‌دهد.تابع تبدیل سیستم با کنترلر تناسبی به صورت زیر می‌باشد:

$$\frac{x(s)}{F(s)}=\frac{k_p}{s^2+10s+(k_p+20)}$$

مقدار بهره تناسبی را برابر 300 قرار می‌دهیم.

```
Kp=300;
num=[Kp];
den=[1 10 20+Kp];
t=0:0.01:2;
step(num,den,t)
```

با اجرای برنامه زیر نمودار زیر حاصل می‌شود:



شکل(۱-۲۶) پاسخ پله سیستم با کنترلر تناسبی

از روی نمودار مشخص است که بهره تناسبی باعث کاهش زمان خیز و خطای حالت ماندگار شده و مقدار جهش نیز افزایش پیدا کرده است. با استفاده از دستور cloop می‌توانید به جای اینکه تابع تبدیل سیستم را با دست محاسبه کنید، تابع تبدیل سیستم حلقه بسته را به طور مستقیم از تابع تبدیل حلقه باز به‌دست آورید.

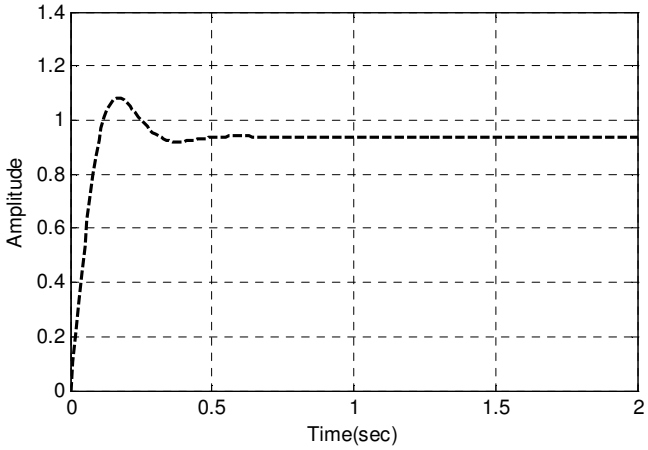
```
num=1; den=[1 10 20];
Kp=300;
[numCL,denCL]=cloop(Kp*num,den);
t=0:0.01:2;
step(numCL, denCL,t)
```

کنترلر مشتق‌گیر و تناسبی: حال اجازه بدهید به بررسی کنترلرPD بپردازیم. از جدول بالا می‌بینیم که کنترلر مشتق‌گیر k_D مقدار جهش و زمان نشست را کاهش می‌دهد. تابع تبدیل سیستم حلقه بسته به کنترلر PD به صورت زیر می‌باشد:

$$\frac{x(s)}{F(s)}=\frac{k_D s+k_p}{s^2+(10+k_D)s+(k_p+20)}$$

اجازه بدهید که قرار دهیم $k_p=300, k_D=10$ و خروجی کنترل شده را مشاهده کنیم. پس دستورات زیر را وارد کنید:

```
Kp=300;Kd=10;
num=[Kd Kp]; den=[1 10+Kd 20+Kp];
t=0:0.01:2;
step(num,den,t)
```



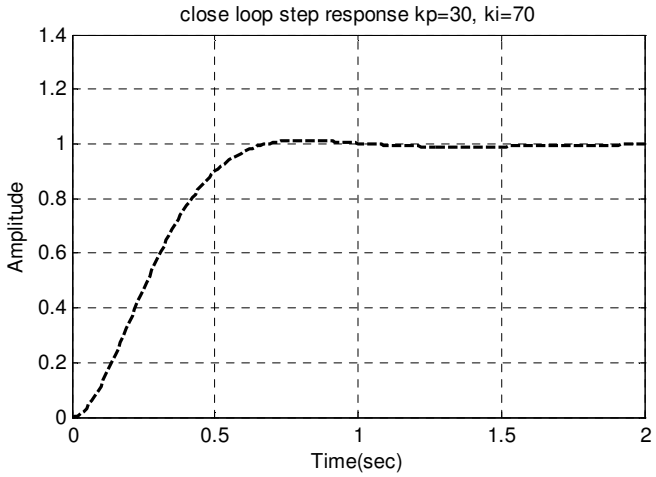
شکل(۱-۲۷) پاسخ پله سیستم با کنترلر مشتق گیر و تناسبی

کنترلر انتگرال‌گیر و تناسبی: قبل از اینکه درباره کنترلر PID بحث کنیم بهتر است که ابتدا کنترلر PI را بررسی کنیم. از جدول بالا، می‌بینیم که k_I مقدار زمان جهش را کاهش می‌دهد ولی مقدار جهش و زمان نشست را افزایش داده و خطای حالت ماندگار را حذف می‌کند. برای سیستم داده شده تابع تبدیل حلقه بسته به صورت زیر می‌باشد:

$$\frac{x(s)}{F(s)}=\frac{k_p s+k_I}{s^3+10s^2+(k_p+20)s+k_I}$$

اجازه بدهید که قرار دهیم $k_p=30, k_I=70$ و خروجی کنترل شده را مشاهده کنیم. پس دستورات زیر را وارد کنید:

```
Kp=30;Ki=70;
num=[Kp Ki];den=[1 10 20+Kp Ki];
t=0:0.01:2;
step(num,den,t)
```



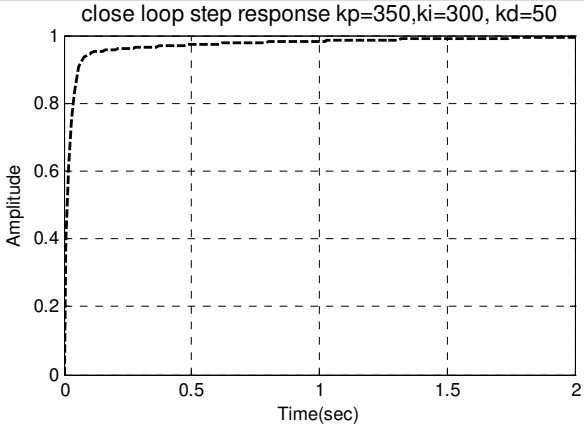
شکل(۱-۲۸) پاسخ پله سیستم با کنترلر تناسبی و انتگرالی

کنترلر انتگرال‌گیر و تناسبی و مشتق‌گیر: حال اجازه دهید به بررسی کنترلر PID بپردازیم. تابع تبدیل سیستم حلقه بسته با کنترلر PID به صورت زیر می‌باشد:

$$\frac{x(s)}{F(s)}=\frac{k_D s^2+k_p s+k_I}{s^3+(10+k_D)s^2+(k_p+20)s+k_I}$$

بعد از چند بار سعی و خطا کردن و انتخاب مقادیر $k_p=350, k_D=50, k_I=300$، به پاسخ دلخواه می‌رسیم. پس دستورات زیر را در یک m-file وارد کنید:

```
Kp=350;Ki=300;Kd=50;
num=[Kd Kp Ki];den=[1 10+Kd 20+Kp Ki];
t=0:0.01:2;
step(num,den,t)
```



چند نکته برای طراحی کنترلر PID

وقتی می‌خواهیم یک کنترل PID برای یک سیستم طراحی کنیم، مراحل زیر را دنبال نمائید تا به پاسخ دلخواه برسید:

۱. ابتدا پاسخ سیستم حلقه باز را به‌دست آورید و ببینید برای به‌دست آوردن یک جواب مطلوب به چه چیزی نیاز داریم.

۲. یک کنترلر تناسبی به سیستم اضافه کنید تا زمان خیز بهبود پیدا کند.

۳. یک کنترلر مشتق‌گیر به سیستم اضافه کنید تا جهش بهبود پیدا کند.

۴. یک کنترلر انتگرال گیر به سیستم اضافه کنید تا خطای حالت ماندگار حذف شود.

۵. مقادیر بهره‌های k_p, k_D, k_I را تنظیم کنید تا به پاسخ دلخواه برسید.

این موضوع را در ذهن خود داشته باشید که حتماً نباید از سه کنترلر k_p, k_D, k_I استفاده کنید. به‌طور مثال وقتی با کنترلر PI جواب مناسب گرفتید دیگر نیازی نیست که کنترلر مشتق‌گیر را نیز به آن اضافه کنید.

فصل دوم

تحلیل سیستم‌های کنترلی با استفاده از تاثیر مکان هندســی ریشه‌ها و روش پاسخ فرکانسی

۲ – ۱ طراحی جبران سازها

۲-۱-۱ طراحی جبران سازهای پیش فاز و پس فاز برای سیستم‌های پیوسته

مراحل اصلی گفته شده در بخش‌های قبلی این کتاب را می‌توان به صورت زیر خلاصه‌سازی کرد:

۱. مدل ریاضی سیستم تحت کنترل را به دست آورده و آن را به شکل مجموعه‌ای از معادلات دیفرانسیل خطی و یا غیرخطی نمایش می‌دهیم.

۲. در صورت لزوم، مدل غیرخطی را با فرض انحرافات کوچک و یا با روشهای دیگر، همانند روش خطی‌سازی فیدبکی[1]، به صورت معادلات خطی در می‌آوریم.

۳. معادلات حاصل را به صورت تابع تبدیل و یا فضای حالت در می‌آوریم.

۴. برای تحلیل سیستم از مدل بیان شده، و یا در صورت امکان با تحت آزمایش قرار دادن سیستم واقعی به صورت حلقه باز استفاده می‌نمائیم.

۵. برای بررسی چگونگی رفتار سیستم در حلقه بسته، کنترل تناسبی ساده k را در نظر می‌گیریم.

۶. مساله پایداری حلقه بسته به ازای مقادیر مختلف k را از طریق روش ترسیم مسیر ریشه‌های حلقه بسته مطالعه نموده، مقادیرخطای ماندگار، جهش، زمان خیز، زمان نشست، محل صفرها و قطبهای حلقه بسته، مقادیر حد بهره و حد فاز، و امثال آنها را مورد مطالعه قرار می‌دهیم. برای این کار باید انواع ورودی‌های فرمان استاندارد (مانند پله، شیب و غیره) قابل انتظار را نیز به سیستم حلقه بسته اعمال نمود.

حال اگر یک یا چند پارامتر از مشخصه‌های سیستم مانند پایداری، سرعت پاسخ، حد بهره و فاز مناسب نبوده و از طریق تغییر مقدار k نیز قابل حصول نباشد، طراح باید به گونه‌های دیگر به مشخصه‌های مطلوب برای عملکرد سیستم دست یابد. از این رو طراح باید با به‌کارگیری روش‌های طراحی و جبران‌سازی پیچیده‌تر، کنترل کننده مناسبی را جهت دست یابی به مشخصه‌های مورد نظر طراحی نماید.

[1] Feedback Linearization

متداول ترین جبران سازها که در صنعت کاربرد فراوانی دارند عبارتند از : **جبران سازهای پیش فاز**[1]**، پس فاز**[2]**، و پس فاز- پیش فاز** [3]

نکته۱: جبران‌ساز پیش‌فاز عموماً ممکن است باعث افزایش پایداری، افزایش سرعت پاسخ یا کاهش زمان نشست سیستم گردد. از طرفی این جبران‌کننده معمولاً موجب تشدید اثر پذیری رفتار سیستم از نویزهای فرکانس بالا می‌گردد. این نویزها معمولاً به دلایلی همچون منابع تغذیه الکتریکی، منابع خارجی نویز مثل لامپهای مهتابی، و یا ارتعاشات سریع سازه‌های مکانیکی ایجاد می‌گردد. این جبران کننده درجه سیستم دینامیکی سیستم حلقه بسته را در صورتی‌که حذف صفر و قطب صورت نگیرد، یک درجه بالا می‌برد.

نکته۲: جبران ساز پس فاز باعث کاهش خطای حالت ماندگار می‌شود؛ در عوض این کنترل کننده زمان پاسخ گذرای سیستم را افزایش داده، پایداری سیستم را کاهش می‌دهد و در صورت عدم دقت طراح، ممکن است سیستم حلقه بسته دچار ناپایداری کامل گردد. جبران ساز پس فاز اثرات نویزهای فرکانس بالا را نیز کاهش می دهد، این جبران کننده نیز درجه سیستم دینامیکی را در صورتی‌که حذف صفر و قطب صورت نگیرد یک درجه بالا می‌برد.

نکته۳: جبران ساز پس فاز- پیش فاز اثرات هر دو جبران سازی پس و پیش فاز را با هم ترکیب کرده، درجه سیستم دینامیکی را در صورتی‌که حذف صفر و قطب صورت نگیرد دو درجه بالا می‌برد. بنابراین استفاده از این کنترل کننده موجب پیچیده‌تر شدن سیستم حلقه بسته شده و امکانات بالقوه‌ای را برای طراحی کنترل کننده بهتر فراهم می‌آورد. بدیهی است که در این حالت، طراحی کنترل کننده مناسب مستلزم مهارت بیشتری است.

چنانکه بیان شد، بعد از به‌دست آوردن مدل ریاضی مناسب برای سیستم تحت کنترل، مراحل شبیه سازی و تحلیل سیستم مورد نظر صورت می‌پذیرد و نقاط قوت و ضعف سیستم به طور کامل تعیین می‌گردد. مشخصه‌های عملکردی بسیاری بسته به نیاز صنعت و کاربرد خاص سیستم کنترل برای تعیین رفتار مطلوب سیستم کنترل وجود دارند، که طراح با روش سعی و خطا به طراحی جبران ساز می‌پردازد تا رفتار مطلوب حلقه بسته را فراهم نماید. با توجه به در نظر گرفتن مسائل عملی از قبیل عناصر غیرخطی در سیستم، محرک‌ها، سنسورها، برخی دینامیک‌های مدل نشده و نامعینی در مدل‌سازی و شبیه سازی سیستم حلقه باز و جبران ساز، بدیهی است که رفتار سیستم حلقه بسته واقعی با رفتار مدل واقعی آن مقداری تفاوت دارد و طراح باید با سعی و خطا به بهینه سازی مدل طراحی شده بپردازد تا اینکه در شرایط کاری واقعی به سیستم مطلوب حلقه بسته دست پیدا کند.

حال به طراحی جبران سازها با استفاده از روش‌های مکان هندسی ریشه ها و پاسخ فرکانسی پرداخته، موارد زیر را مورد بحث قرار می‌دهیم:

[1] Phase Lead
[2] Phase lag
[3] Phase Lead-Lag

۱. طراحی جبران ساز پیش فاز با استفاده از مکان هندسی ریشه ها
۲. طراحی جبران ساز پیش فاز با استفاده از روش پاسخ فرکانسی
۳. طراحی جبران ساز پس فاز با استفاده از مکان هندسی ریشه ها
۴. طراحی جبران ساز پس فاز با استفاده از روش پاسخ فرکانسی
۵. جبران سازهای lead-lag

توجه: اگر سیستم کنترلی با کنترل تناسبی ساده و به ازای کلیه مقادیر بهره K ناپایدار بوده و یا پاسخ آن مشخصه‌های مطلوبی نداشته باشد، بدیهی است که با تغییر K به تنهایی نمی‌توان به مشخصه‌های مطلوب عملکرد دست پیدا کرد. دراین صورت می‌توان با تغییر شکل نمودار مکان هندسی ریشه‌های سیستم توسط جبران سازهای مناسب، به مشخصه‌های عملکردی مطلوب تعیین شده رسید.

۲-۱-۱-۱ طراحی جبران ساز پیش فاز با استفاده از مکان هندسی ریشه‌ها

جبران ساز پیش فاز مرتبه اول با استفاده از مکان هندسی ریشه ها نیز طراحی می‌شود. این جبران کننده به فرم مکان هندسی ریشه ها به صورت زیر می‌باشد:

$$G_{lead}(s) = K_c \frac{(s - z_0)}{(s - p_0)} \qquad (۱-۲)$$

که در آن اندازهٔ z_0 کوچکتر از p_0 است. جبران کننده پیش فاز باعث کشیده شدن مکان هندسی ریشه ها به سمت نیم صفحه چپ می‌گردد. این نتیجه باعث افزایش و بهبود پایداری سیستم و افزایش پاسخ سرعت سیستم می‌گردد. حال این کار چگونه انجام می‌شود؟

زمانی که یک جبران کننده پیش فاز به سیستم اضافه می‌شود، محل تلاقی مجانب‌ها نسبت به قبل، در محلی با مقدار منفی بزرگتر ایجاد می‌شود. علت این امر آن است که، اگرچه اختلاف تعداد قطب ها و صفر های سیستم حلقه باز نسبت به قبل تغییری نکرده است، اما قطب اضافه شده توسط کنترل کننده از صفر آن منفی‌تر است. بنابراین جبران کننده باعث می‌شود که محل تلاقی مجانب‌ها با محور حقیقی به سمت چپ انتقال یابد،که این خود باعث افزایش ناحیه پایداری و افزایش سرعت پاسخ‌گویی سیستم حلقه بسته می‌گردد. در محیط MATLAB جبران‌کننده پیش فاز به روش مکان هندسی ریشه‌ها با استفاده از تابع تبدیل پیاده‌سازی می‌شود.

```
numlead = kc*[1 z];
denlead = [1 p];
lead = tf(numlead,denlead);
```

سپس با استفاده از تابع conv صورت و مخرج این تابع تبدیل را در صورت و مخرج سیستم ضرب می‌کنیم.

۲-۱-۱-۲ طراحی جبران ساز پیش فاز با استفاده از روش پاسخ فرکانسی

تابع تبدیل جبران کننده پیش فاز مرتبه اول به فرم پاسخ فرکانسی به صورت زیر می‌باشد:

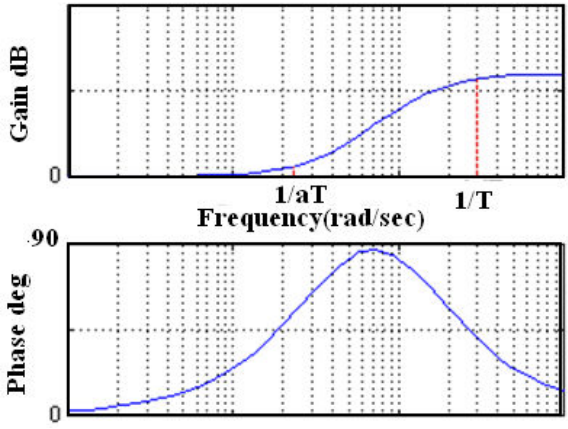
$$G_{lead}(s)=\frac{1+aTs}{1+Ts} \qquad a>1 \tag{۲-۲}$$

این رابطه معادل فرم مکان هندسی ریشه‌های $G(s)=K_c\frac{(s-z_0)}{(s-p_0)}$ است که در آن،

$$p=\frac{1}{T}\ ,\ z=\frac{1}{aT}\ \ ,\ K_c=a \tag{۲-۳}$$

جبران کننده پیش فاز یک فاز مثبت در رنج فرکانسی $\frac{1}{T}$ تا $\frac{1}{aT}$ به سیستم اضافه می‌کند.

نمودار بود این جبران کننده به صورت زیر می‌باشد:



شکل(۲-۱) نمودار جبران ساز پیش فاز

$\frac{1}{T}$ و $\frac{1}{aT}$ فرکانس‌های گوشه می‌باشند. توجه کنید که فاز مثبت در بین این دو فرکانس به سیستم اضافه می شود که بسته به مقدار a تا حداکثر 90 درجه می‌توان آن‌را به فاز سیستم اضافه کرد. برای اضافه کردن فاز بیشتر از 90 درجه باید از دو جبران کننده پیش فاز استفاده کنید. حداکثر مقدار فاز اضافه شده، در مرکز فرکانس یا میانگین هندسی فرکانس‌های گوشه یعنی در نقطه $\omega_n=\frac{1}{T\sqrt{a}}$ می‌باشد. معادله ای که حداکثر فاز را تعیین می‌کند به صورت زیر می‌باشد:

$$Sin\varphi=\frac{a-1}{a+1} \tag{۲-۴}$$

اضافه کردن فاز مثبت باعث افزایش حد فاز سیستم حلقه بسته شده و در نتیجه پایداری سیستم بیشتر می-شود. تاثیر دیگر جبران کننده پیش فاز در نمودار دامنه دیده می‌شود. این جبران کننده بهره سیستم را در

فرکانس‌های بالا افزایش می‌دهد. مقدار این افزایش بهره برابر با a است. این مقدار باعث افزایش فرکانس گذر (w_g) می‌شود و همین عامل باعث کاهش زمان خیز و کاهش زمان نشست سیستم می‌گردد.
در MATLAB پیاده سازی جبران کننده پیش فاز به فرم پاسخ فرکانس با استفاده ار تابع تبدیل به صورت زیر است:

```
numlead = kc*[1 z];
denlead = [1 p];
lead = tf(numlead,denlead);
```

سپس با استفاده از تابع conv صورت و مخرج این تابع تبدیل را در صورت و مخرج سیستم ضرب می‌کنیم.

۲-۱-۱-۳ طراحی جبران ساز پس فاز با استفاده از مکان هندسی ریشه ها

جبران کننده پس فاز مرتبه اول با استفاده از مکان هندسی ریشه ها نیز طراحی می‌شود. این جبران کننده به فرم مکان هندسی ریشه ها به صورت زیر می‌باشد:

$$G_{lag}(s) = K_c \frac{(s - z_0)}{(s - p_0)} \qquad (۵-۲)$$

که اندازهٔ z_0 بزرگتر از p_0 است. جبران کننده پس فاز باعث کشیده شدن مکان هندسی ریشه ها به سمت نیم صفحه راست می‌گردد که نتیجه‌ای نامطلوب است. بنابراین صفر و قطب باید نزدیک هم باشند که معمولاً نزدیک مبدا در نظر گرفته می‌شود. حال ببینیم این کنترل کننده چگونه مکان هندسی را به سمت راست می‌کشاند؟

زمانی که یک جبران کننده پس فاز به سیستم اضافه می‌شود محل تلاقی مجانب‌ها نسبت به قبل، در محلی با مقدار منفی کوچکتر ایجاد می‌شود. علت این امر آن است که، اگرچه اختلاف تعداد قطب ها و صفر های سیستم حلقه باز نسبت به قبل تغییری نکرده است، اما صفر اضافه شده توسط کنترل کننده از قطب آن منفی تر است. بنابراین جبران کننده باعث می‌شود که محل تلاقی مجانب ها با محور حقیقی به سمت راست انتقال یابد، که این خود باعث کاهش ناحیه پایداری و کاهش سرعت پاسخ گویی سیستم حلقه بسته می‌گردد.
این کنترل کننده فقط باعث کاهش خطای حالت ماندگار می‌گردد و اصولاً برای اصلاح پاسخ حالت گذرا و پایداری سیستم به کار نمی‌رود. در فرکانس‌های بالا کنترلر پس فاز بهره واحد خواهد داشت و در فرکانس‌های پائین برابر $\frac{z_o}{p_o}$ خواهد بود که بزرگتر از 1 است. این فاکتور در ثابت موقعیت، سرعت، و شتاب k_a, k_v, k_p ضرب می‌شود و خطای حالت ماندگار به نسبت $\frac{Z_o}{P_o}$ کاهش پیدا می‌کند.

در محیط MATLAB جبران کننده پیش فاز به فرم مکان هندسی ریشه ها با استفاده از تابع تبدیل پیاده سازی می‌شود.

```
numlag = [1 z];
denlag = [1 p];
```

```
lag = tf(numlag,denlag);
```

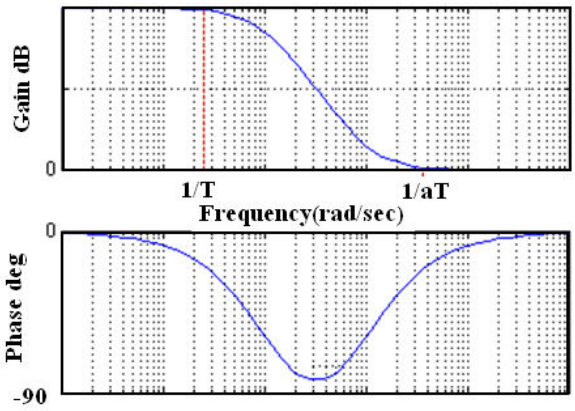
سپس با استفاده از تابع conv صورت و مخرج این تابع تبدیل را در صورت و مخرج سیستم ضرب می‌کنیم.

۴-۱-۱-۲ طراحی جبران ساز پس فاز با استفاده از روش پاسخ فرکانسی

تابع تبدیل جبران کننده پس فاز مرتبه اول به فرم پاسخ فرکانسی به صورت زیر می‌باشد:

$$G_{lag}(s)=\frac{1}{a}\left(\frac{1+aTs}{1+Ts}\right) \qquad a<1 \tag{۶-۲}$$

شکل کلی این جبران کننده مشابه جبران کننده پیش فاز است با این تفاوت که در اینجا a<1 است. تفاوت اصلی این جبران کننده با پیش فاز در این است که یک فاز منفی در رنج فرکانسی $\frac{1}{T}$ تا $\frac{1}{aT}$ به سیستم اضافه می‌کند، که این امر اصولاً نامطلوب است، اما مزیت این کنترل کننده در افزایش بهره سیستم حلقه باز در فرکانس‌های کوچک است که این امر به معنای بهتر شدن خطای ماندگار است. البته این مزیت زمانی مطرح است که سیستم حلقه بسته در کاهش حد فاز دچار کاهش غیر قابل قبول پایداری نشده باشد. نمودار بود این جبران کننده به صورت زیر می‌باشد:



شکل(۲-۲) نمودار جبران ساز پس فاز

$\frac{1}{T}$ و $\frac{1}{aT}$ فرکانس‌های گوشه می‌باشند. تاثیر اصلی کنترل پس فاز در نمودار دامنه قابل نمایش است. این کنترلر بهره را در فرکانس‌های پائین اضافه می‌کند. بزرگی این بهره برابر a است و تاثیر این بهره باعث کاهش خطای حالت ماندگار سیستم حلقه بسته با فاکتور a می‌شود، زیرا بهره کنترلر در فرکانس‌های بالا و وسط برابر 1 است. وابسته به مقدار a مقادیر مختلف فازی بزرگتر از 90- را می‌توان به سیستم اضافه کرد. در MATLAB پیاده سازی جبران کننده پیش فاز به فرم پاسخ فرکانس با استفاده از تابع تبدیل به صورت زیر می‌باشد:

```
numlag = [a*T 1];
denlag = a*[T 1];
lag = tf(numlag,denlag);
```

سپس با استفاده از تابع conv، صورت ومخرج این تابع تبدیل را در صورت و مخرج سیستم ضرب می‌کنیم.

۲-۱-۱-۵ جبران ساز پیش فاز- پس فاز

این جبران کننده اثر جبران کننده‌های پیش فاز و پس فاز را با هم ترکیب می‌کند ودر نتیجه پاسخ حالت گذرا، پایداری و خطای حالت ماندگار کاهش پیدا می‌کند. برای پیاده سازی کنترلر Lead-lag اول باید کنترلر پیش فاز را برای رسیدن به یک پاسخ حالت گذرای مطلوب و پایداری طراحی کرد. سپس یک کنترلر پس فاز برای کاهش خطای حالت ماندگار به سیستم کنترلی اضافه می‌شود.

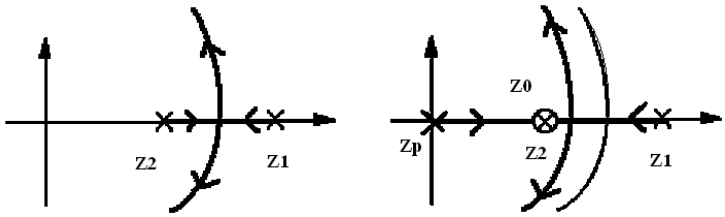
۲-۲-۱ طراحی جبران سازهای پیش فاز و پس فاز دیجیتالی

۲-۱-۲-۱ جبران ساز پیش فاز

جبران کننده پیش فاز مرتبه اول زیر را در نظر بگیرید:

$$G_{lead}(z) = K_d \frac{(z - z_0)}{(z - p_0)} \qquad (۷-۲)$$

که بهره K_d باید $K_d = \frac{1 - p_0}{1 - z_0}$ باشد تا تاثیری روی پاسخ حالت ماندگار نگذارد. قطب p_0 باید قطبی داخل دایره واحد باشد. برای تقدم فاز مقدار صفر باید بزرگتر از قطب باشد و ضریب K_d باید بزرگتر از 1 باشد. به‌طور عمومی برای طراحی جبران کننده با تقدم فاز، z_0 را نزدیک یکی از قطب‌های سیستم برای حذف صفر و قطب قرار می‌دهیم. قطب p_0 در سمت چپ صفر قرار می‌گیرد تا مکان هندسی ریشه ها را به سمت چپ بکشاند. شکل زیر نشان می‌دهد که چگونه با گذاشتن صفر و قطب، مکان هندسی به سمت چپ کشیده می‌شود.



شکل (۲-۳) کشیده شدن مکان هندسی به سمت چپ با گذاشتن صفر و قطب

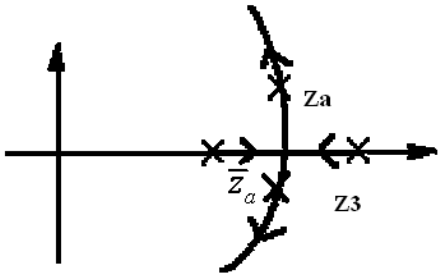
کشیده شدن مسیر ریشه‌ها به سمت چپ باعث سریع شدن پاسخ نیز می‌گردد. توجه کنید که با توجه به ملزومات طراحی مثل زمان نشست ومقدار جهش، می‌توانیم در یابیم که تا چه حد باید مکان هندسی ریشه‌ها را به سمت چپ بکشیم.

۲-۱-۲-۲ جبران ساز پس فاز

جبران کننده پس فاز مرتبه اول زیر را در نظر بگیرید:

$$G_{lag}(z) = K_d \frac{(z - z_0)}{(z - p_0)} \qquad (۸-۲)$$

که بهره K_d باید $K_d = \frac{1-p_0}{1-z_0}$ باشد. قطب p_0 باید قطبی داخل دایره واحد باشد. برای تأخیر فاز، مقدار قطب باید بزرگتر از صفر باشد و ضریب K_d باید کوچکتر از 1 باشد. به طور عمومی برای طراحی جبران کننده با تأخیر فاز، z_0 را نزدیک یکی از قطب‌های سیستم برای حذف صفر و قطب قرار می‌دهیم. فلسفه این طراحی در زیر آمده است.



فرض کنید که z_a و $\overline{z}_a$ پاسخ گذرای مطلوب (زمان خیز، زمان نشست، و مقدار جهش) را به ما می‌دهند. اما مقدار بهره K باید افزایش پیدا کند تا خطای حالت ماندگار کاهش یابد. ما قطب جبران کننده را نزدیک z=1 و صفر را سمت چپ قطب سیستم در نظر می‌گیریم.

یادآوری می‌کنیم که مقدار بهره K در صفر برابر بی‌نهایت و در قطب برابر صفر است. با اضافه کردن صفر به مکان هندسی ریشه ها مقدار بهره در z_a و $\overline{z}_a$ افزایش می‌یابد. همچنین صفر و قطب‌های جبران کننده باید در همان موقعیت مطلوب قرار گرفته باشند.

۲ – ۲ ترسیم مکان هندسی ریشه‌ها

۲-۲-۱ مقدمه

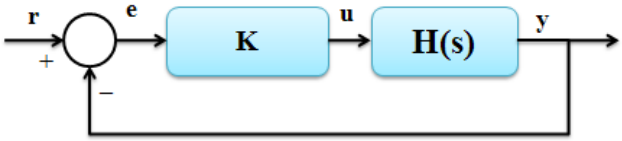
می‌دانیم که رفتار حالت گذرای یک سیستم حلقه بسته به موقعیت قطب‌های حلقه بسته آن بستگی دارد. اگر سیستم یک بهره متغیر و قابل تنظیم داشته باشد در آن صورت موقعیت قطب‌های حلقه بسته با تغییر و تنظیم این بهره تغییر خواهد کرد. بنابراین در طراحی سیستم کنترل باید از روند تغییرات قطب‌های حلقه بسته با تغییر بهره آن اطلاع داشته باشیم. برای پیدا کردن این روند و یا تعیین قطب‌های سیستم حلقه بسته می‌توان معادله مشخصه سیستم حلقه بسته را حل کرد؛ یعنی طوری بهره را تغییر دهیم که قطب‌های سیستم حلقه بسته در مکان‌های مطلوب قرار بگیرند. حال اگر با تغییر بهره نتوانیم به مشخصه‌های مطلوب عملکرد سیستم حلقه بسته دست پیدا کنیم، باید از جبران سازهای پیش فاز (lead) و پس فاز (lag) استفاده کنیم.

۲-۲-۲ تعریف مکان هندسی ریشه ها

به محل ریشه‌های معادله مشخصه یک سیستم کنترل، وقتی پارامتر k (بهره کنترلی) از صفر تا بی نهایت تغییر می کند، مکان هندسی ریشه‌های آن گفته می‌شود. در این روش، مکان هندسی ریشه‌های معادله مشخصه برای

کلیه مقادیر یکی از پارامترهای سیستم رسم می‌شود. ریشه‌های متناظر با یک مقدار خاص از این پارامتر را می توان بر روی نمودار حاصل تعیین کرد. در اغلب موارد این پارامتر همان بهره حلقه باز سیستم کنترل است، اما در حالت کلی ترسیمه مسیر ریشه‌ها را برای مطالعه هر پارامتر دیگری نیز می‌توان رسم نمود.

قطب‌های حلقه بسته: محل ریشه‌های معادله مشخصه سیستم کنترل نقش تعیین کننده‌ای در رفتار و مشخصات حالت گذرای سیستم کنترل دارد. بنابراین بسیار ارزشمند و مفید است که مکان ریشه‌های معادله مشخصه سیستم کنترل را، وقتی پارامتر سیستم (k) از صفر تا بی نهایت تغییر می‌کند، به دست آورده و مورد مطالعه قرار دهیم.



تابع تبدیل سیستم حلقه بسته بالا برابر است با:

$$\frac{Y(s)}{R(s)}=\frac{k.H(s)}{1+k.H(s)} \qquad (۹-۲)$$

بنابراین قطب‌های سیستم حلقه بسته مقادیری هستند که به ازای آنها، $1+K.H(s)=0$ شود. حال اگر $H(s)=\frac{b(s)}{a(s)}$، بنابراین معادله بالا را می‌توان به فرم زیر نوشت:

$$1+k\frac{b(s)}{a(s)}=0\Rightarrow a(s)+k.b(s)=0 \qquad (۱۰-۲)$$

فرض کنید که n مرتبه $a(s)$ و m مرتبه $b(s)$ باشد. تمام مقادیر مثبت k را در نظر می‌گیریم. اگر $k\to 0$ در آن صورت قطب‌های حلقه بسته برابر $a(s)=0$ یا قطب‌های $H(s)$ است و اگر $k\to\infty$ در آن صورت قطب‌های حلقه بسته برابر $b(s)=0$ یا صفرهای $H(s)$ می‌باشد. در این کتاب k مثبت در نظر گرفته می‌شود. برای حالتی که k منفی باشد باید به کتابهای استاندارد کنترل مراجعه کنید.

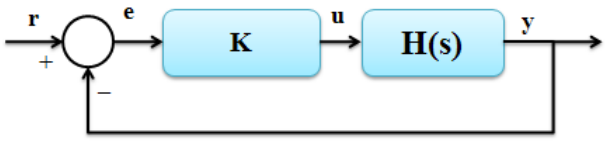
نکته۱: $k=0$ یا شروع مکان ریشه ها، قطب‌های $H(s)$ است و $k=\infty$ یا پایان مکان ریشه‌ها، صفرهای $H(s)$است. این سیستم حلقه بسته باید n قطب داشته باشد، که n تعداد قطب‌های $H(s)$ است و مکان هندسی ریشه ها نیز باید n شاخه داشته باشد. هر شاخه از یک قطب $H(s)$ شروع می‌شود و به یک صفر $H(s)$ ختم می‌شود.

نکته۲: اگر تعداد قطب‌های $H(s)$ بیشتر از صفرهای $H(s)$ باشد، در آن صورت می‌گوئیم سیستم صفر یا صفرهایی در بی‌نهایت دارد. تعداد صفرهای در بی‌نهایت برابر n-m است (تعداد قطب ها منهای تعداد صفر ها) و از طرفی این مقدار برابراست با تعداد شاخه‌های مکان هندسی ریشه ها که به سمت بی‌نهایت میل می‌کند.

با به‌دست آوردن این نمودار می‌توان ریشه‌های مطلوب معادله مشخصه را روی آن مشخص کرده، و بهره‌ای که این ریشه ها را ارائه می‌دهد محاسبه نمود. با تعیین ریشه‌های مطلوب می‌توان پاسخ حالت گذرای سیستم را پیش بینی کرد. با به‌دست آوردن مکان ریشه ها، می‌توان تغییر در عملکرد سیستم کنترل را نسبت به تغییر در بهره K تعیین کرد.

۲-۲-۳ رسم مکان هندسی ریشه ها از طریق تابع تبدیل

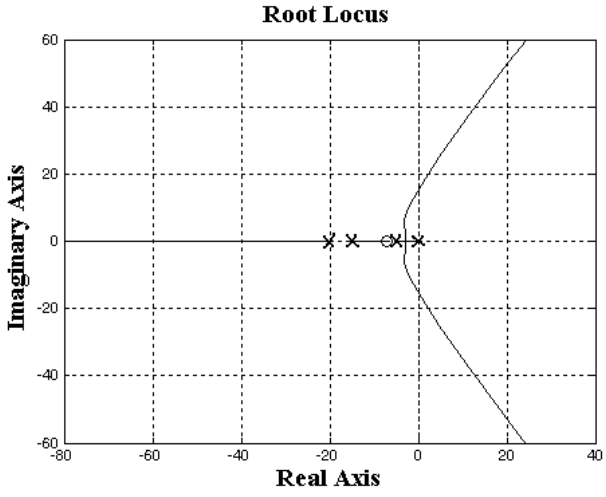
فرض کنید تابع تبدیل یک سیستم حلقه باز به صورت زیر باشد:



$$H(s)=\frac{Y(s)}{R(s)}=\frac{s+7}{s(s+5)(s+15)(s+20)}$$

هدف ما از این طراحی این است که جهش،کمتر از 5% و زمان خیز کمتر از 1 ثانیه باشد. پس در یک m-file دستورات زیر را وارد نموده، و مکان هندسی ریشه ها را رسم کنید:

```
num=[1 7];
den=conv(conv([1 0],[1 5]),conv([1 15],[1 20]));
sys=tf(num,den);
rlocus(sys)
axis([-22 3 -15 15])
```



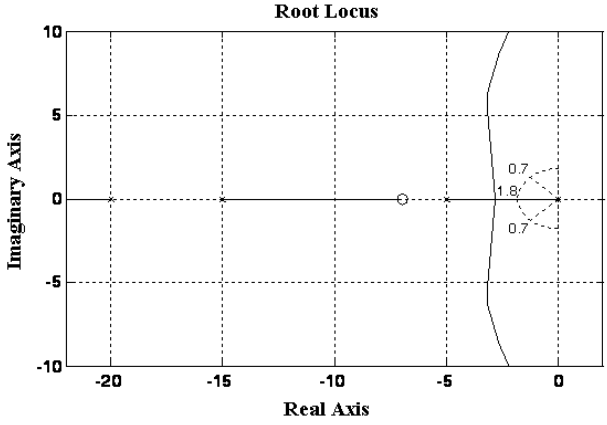
شکل (۲-۴) مکان هندسی ریشه‌ها

۲-۲-۴ انتخاب k از روی مکان هندسی ریشه ها

نمودار بالا موقعیت‌های ممکن برای قطب‌های حلقه بسته با یک بهره تناسبی K را نشان می‌دهد. روشن است که این قطب ها تمام معیارهای طراحی ما را برآورده نمی‌کنند. بنابراین باید حتماً تعیین کنیم که کدام قسمت مکان ریشه ها قابل پذیرش است. ما می‌توانیم با استفاده از دستور sgrid که به دو آرگومان ζ, ω_n نیاز دارد موقعیت قطب‌های مطلوب را تعیین کنیم. در این مساله با توجه به زمان خیز 1 ثانیه‌ای و مقدار جهش 5% ،مقادیر فرکانس طبیعی میرا نشده و نسبت میرایی را به‌دست می‌آوریم: $\zeta = 0.707$, $\omega_n = 1.8$

در ادامه دستورات زیر را وارد کنید:

```
zeta=0.7;
Wn=1.8;
sgrid(zeta, Wn)
```



شکل (۲-۵) مکان هندسی ریشه‌ها

در شکل بالا دو خط چین صاف دیده می‌شود که با زاویه 45 درجه رسم شده اند. این دو خط‌چین نشان دهنده قطب‌هایی هستند که نسبت میرایی آنها برابر $\zeta = 0.707$ می‌باشد. در بین این دو خط‌چین قطب‌هایی با $\zeta < 0.7$ ودر خارج این دو خط‌چین $\zeta > 0.7$ است. پس برای جهش کمتر از 5% باید $\zeta > 0.7$ یعنی بین دو خط باشیم.

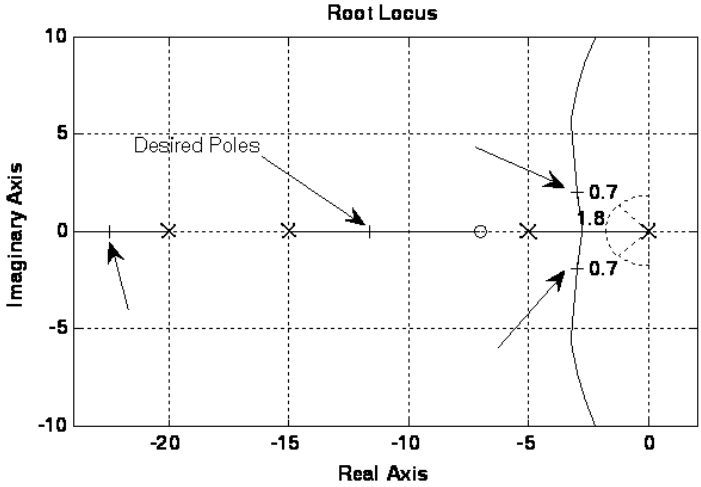
نیم دایره رسم شده نیز موقعیت قطب‌هایی با فرکانس طبیعی میرا نشده $\omega_n = 1.8$ را نشان می‌دهد. داخل دایره $\omega_n < 1.8$ و خارج دایره $\omega_n > 1.8$. برای زمان خیز کمتر از 1 ثانیه باید قطب‌های خارج دایره را انتخاب کنیم پس $\omega_n > 1.8$.

توجه: تا اینجا می‌دانیم که قطب‌های بین دو خط‌چین و خارج دایره پذیرفتنی هستند از طرفی تمام قطب ها در سمت چپ محور موهومی قرار گرفته اند، پس سیستم **حلقه بسته پایدار** می‌باشد.

دراین نمونه انتظار داریم که کنترلر تناسبی قطب‌ها را به منطقه دلخواه بکشاند. شما می‌توانید با استفاده از دستور rlocfind قطب دلخواهتان را انتخاب کنید. در نمودار حاصل در جایی کلیک کنید که می‌خواهید قطب‌ها در آن نقطه قرار بگیرند، سپس مقدار k متناظر با این نقطه را در پنجره MATLAB بخوانید.

```
[k,poles] = rlocfind(sys)
```

```
k =
   189.9850
poles =
 -22.4685
 -11.6406
  -2.9455 + 1.9606i
  -2.9455 - 1.9606i
```



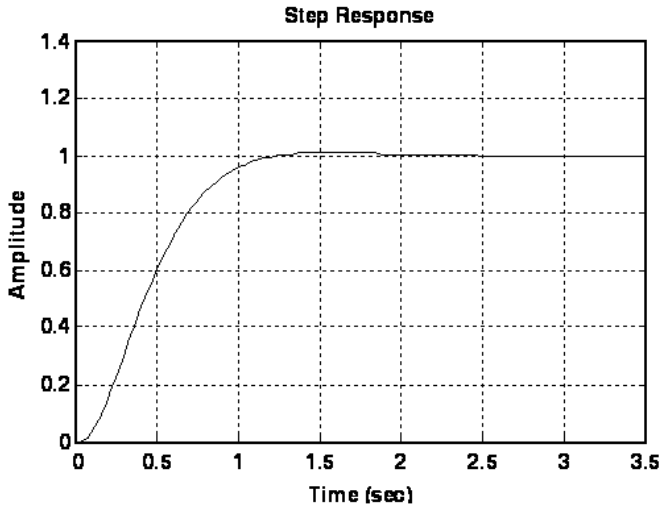
شکل (۲-۶) مکان هندسی ریشه‌ها

پاسخ حلقه بسته: برای پیدا کردن پاسخ سیستم حلقه بسته نیاز داریم که تابع تبدیل سیستم حلقه بسته را در اختیارداشته باشیم. برای این کار از دستور feedback استفاده می‌کنیم. در صورتی که فیدبکی غیر واحد داشته باشید باز هم می‌توانید با این دستور تابع تبدیل حلقه بسته را حساب نموده، و سپس پاسخ سیستم حلقه بسته به ورودی پله را حساب کنید.

```
sys_cl= feedback(k*sys,1);
step(sys_cl);
```

Transfer function:

```
                467.8 s + 3274
-------------------------------------------
s^4 + 40 s^3 + 475 s^2 + 1968 s + 3274
```



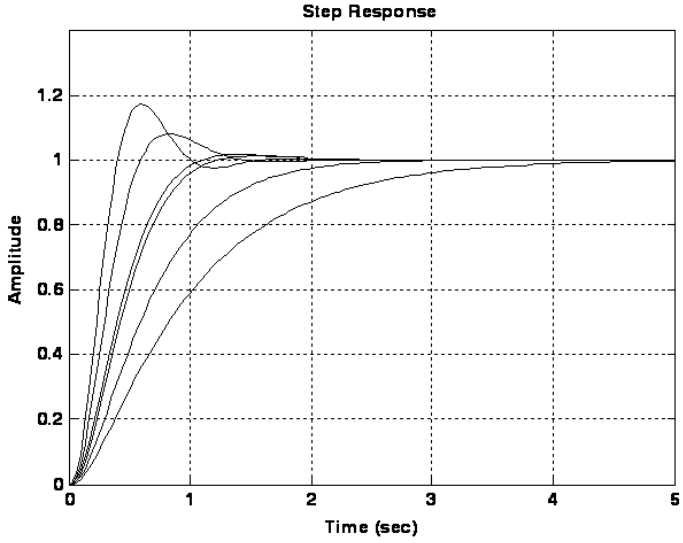
شکل (۲-۷) پاسخ سیستم حلقه بسته

با تحلیل مکان هندسی ریشه ها نتایج زیر را برای این سیستم به ازای افزایش بهره آن می‌توان به دست آورد:

۱. افزایش بهره موجب کاهش نسبت میرایی و در نتیجه افزایش جهش می‌شود.

۲. افزایش بهره باعث افزایش فرکانس طبیعی میرا نشده ω_n می‌شود و زمان خیز کاهش پیدا می‌کند.

۳. افزایش بهره باعث افزایش فرکانس طبیعی میرا شده ω_d می‌شود.



شکل (۲-۸) مکان هندسی ریشه‌ها

۵-۲-۲ تحلیل پایداری

یکی از مهمترین مشخصه‌های سیستم، پایداری[۱] آن است. این ویژگی اولین ویژگی‌ای است که برای هر سیستم کنترل فیدبک مورد توجه قرار می‌گیرد. به سیستمی پایدار می‌گویند که، اگر ورودی یا اغتشاش با دامنه محدود به آن اعمال گردد، پاسخ به‌دست آمده دامنه‌ای محدود داشته باشد.

نکته: توجه شود که مفهوم پایداری یک مفهوم کاملاً وسیع بوده و تعاریف گوناگونی برای آن ارائه شده است، مثلاً لیاپانوف یک تعریف برای پایداری ارائه کرده است. البته تعاریف صورت گرفته بی پایه و اساس نبوده و به دنبال هر تعریف چند تئوری آمده است که آن تعریف را حمایت می‌کند. در این جا ما پایداری به روش BIBO[۲] را تعریف می‌کنیم.

یک سیستم دینامیکی یا کنترلی را پایدار[۳] می‌گویند اگر به ازاء هر ورودی با دامنه محدود (کراندار) یا هر شرایط اولیه غیر صفر، خروجی سیستم کراندار باشد. سیستم کنترل ناپایدار[۴] ،سیستمی است که پایدار نباشد. پاسخ حالت گذرا یا حالت آزاد نیز خود همیشه به صورت مجموع تمام مودهای سیستم[۵] می باشد . بنابراین چنانچه مجموع تمام مودهای یک سیستم به سمت صفر میل کند آن سیستم پایدار خواهد بود زیرا پاسخ حالت گذرای آن تحت هر شرط اولیه به سمت صفر میل می کند. اما اگر حتی یکی از مودهای سیستم به سمت بینهایت میل کند پاسخ حالت گذرای آن به سمت بینهایت میل نموده و در نتیجه سیستم ناپایدار خواهد بود.

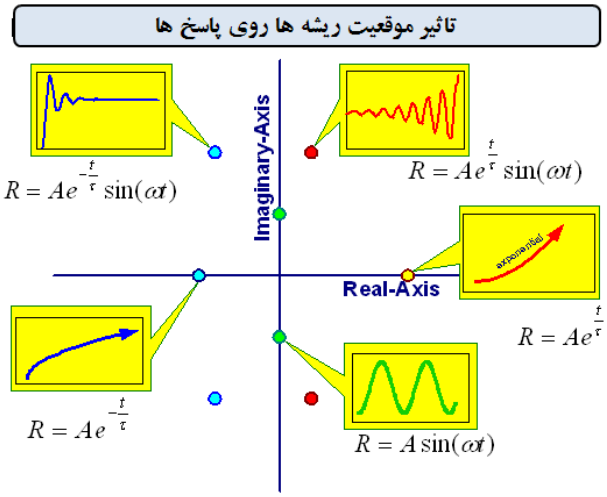
بررسی چند نکته

۱. اگر قطب‌های تابع تبدیل در حوزه s در سمت چپ محور موهومی قرار گیرند پاسخ گذرای سیستم میرا شونده است و با گذشت زمان به سمت صفر میل می‌کند و سیستم پایدار می‌باشد.
۲. اگر یک یا چند زوج قطب غیر تکراری تابع تبدیل بر روی محور موهومی قرار گرفته و سایر قطب ها در سمت چپ محور موهومی قرار بگیرند پاسخ سیستم به صورت نوسانات سینوسی خواهد بود.
۳. اگر تنها یکی از قطب ها در سمت راست محور موهومی باشد، دامنه پاسخ سیستم با گذشت زمان افزایش می‌یابد. در آن صورت سیستم حلقه بسته ناپایدار خواهد بود.
۴. توجه داشته باشید که قطب‌های نزدیک محور موهومی بیشترین تاثیر را روی پاسخ سیستم دارند.

---

[۱] Stability
[۲] Bounded Input Bounded Output
[۳] stable
[۴] unstable

[۵] منظور از مودهای سیستم توابع نمایی به شکل $t^{k_i}e^{p_i t}$ است که در آن p_i قطب i ام سیستم و k_i تعداد تکرار آن قطب است. پاسخ هر سیستم خطی به شرایط اولیه غیر صفر در واقع مجموعه‌ای وزن دار از مودهاست. وزن مودها در پاسخ سیستم از جمله به محل صفرهای سیستم بستگی دارد.

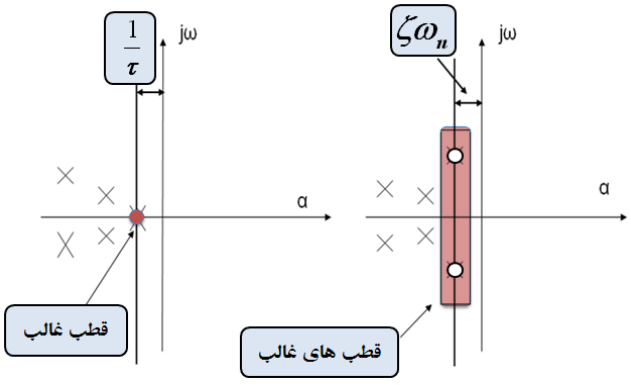


شکل (۲-۹) تاثیر موقعیت ریشه ها روی پاسخ‌ها

برای تعیین پایداری سیستم از روش راث - هرویتز[1] استفاده می‌شود.

۲-۲-۶ سرعت پاسخ

دومین پارامتر مهم در طراحی سیستم های کنترل فیدبک پارامتر سرعت است. سرعت پاسخ سیستم به مقدار ثابت زمانی سیستم بستگی دارد به‌طوری‌که هر چه ثابت زمانی سیستم کمتر باشد، سرعت پاسخ سیستم بیشتر خواهد بود. ثابت زمانی برابر عکس فاصله ریشه غالب تا محور قائم است از این‌رو هر چه ریشه های غالب دورتر از خط قائم باشند سرعت پاسخ سیستم بیشترخواهد شد. توجه داشته باشید که قطب‌های نزدیک محور موهومی بیشترین تاثیر را روی پاسخ سیستم دارند. پس برای افزایش سرعت سیستم باید تا حد امکان این قطب ها را از محور موهومی دور کنیم.

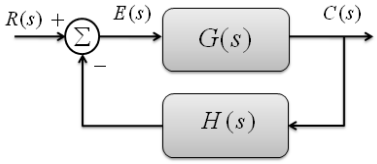


شکل (۲-۱۰) موقعیت قطب‌های غالب

[1] Routh-**Hurwitz**

۷-۲-۲ دقت

سومین پارامتر مهم در طراحی سیستم های کنترل فیدبک پارامتر دقت است. خطـای حالـت مانـدگار سیسـتم کنترل فیدبک به صورت زیر تعریف می‌شود:



$$E(s)=R(s)-H(s).C(s)=R(s)[1-H(s).\frac{C(s)}{R(s)}]$$
$$\Rightarrow E(s)=R(s)[1-H(s).\frac{G(s)}{1+G(s).H(s)}]=\frac{R(s)}{1+G(s).H(s)} \qquad (۱۱-۲)$$
$$e_{ss}=\lim_{s\to 0}\frac{s.R(s)}{1+G(s).H(s)}$$

توجه کنید که بررسی سرعت و دقت سیستم کنترل، زمانی صورت می گیرد که سیستم پایدار باشد.

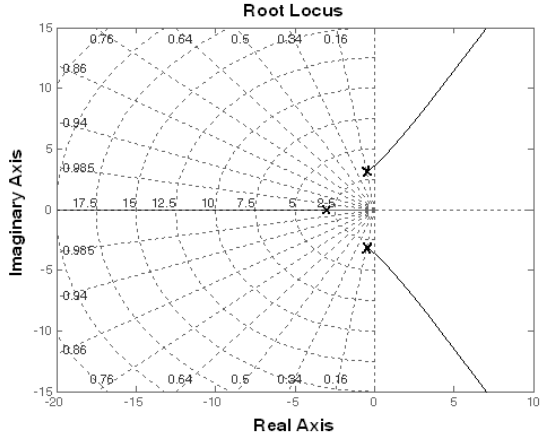
۸-۲-۲ Notch Filter

در بسیاری موارد تابع تبدیل سیستم کنترل شده به گونه‌ای است که یک یا چند قطب مزدوج مختلط در صفحه s نزدیک محور موهومی دارد. این گونه سیستم‌ها تحت کنترل حلقه بسته تناسبی منجر به سیستم حلقه بسته با پایداری ناچیز و یا در اصل ناپایدار خواهد شد. به طور مثال تابع تبدیل زیر را در نظر بگیرید:

$$G_P(s)=\frac{K}{(s+3)(s^2+s+10)}$$

برای رسم مکان هندسی ریشه ها دستورات زیر را وارد کنید و در ابتدا فرض کنید بهره سیستم برابر 1 می‌باشد.

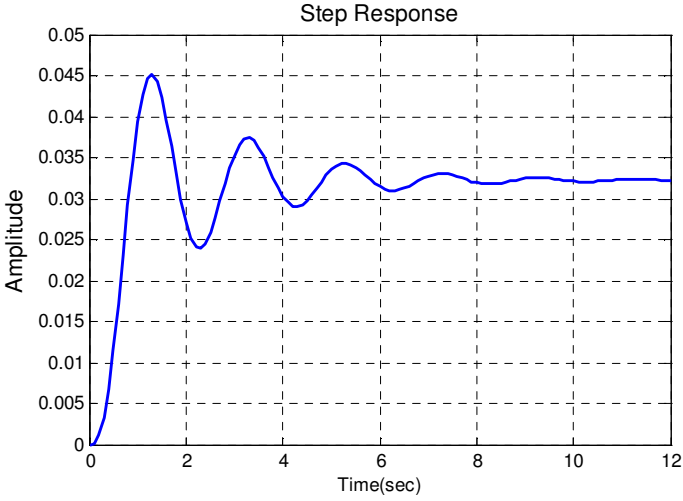
```
num = 1;den = conv([1 3],[1 1 10]);
rlocus(num,den)
```



شکل (۲-۱۱) مکان هندسی ریشه ها

همانطور که از روی شکل(۲-۱۱)مشخص است، سیستم فقط در یک ناحیه کوچک از مکان هندسی ریشه‌ها پایدار است و در قسمتی که پایدار است میرایی بسیار ناچیزی دارد چون مقدار ζ کم است. با بستن حلقه و رسم پاسخ پله برای این ناحیه کوچک پایداری در مکان هندسی ریشه ها و با فرض بهره کنترلی واحد نمودار شکل(۲-۱۲)حاصل می‌گردد که بیانگر عملکردی بسیار ضعیف است:

```
[numc,denc]=cloop(num,den,-1);
step(numc,denc)
```



شکل (۲-۱۲) پاسخ پله سیستم حلقه بسته به ورودی پله

با توجه به نمودار حاصل می‌بینیم که مقدار جهش زیاد، زمان نشست طولانی، و خطای حالت ماندگار نیز زیاد است. اگر مقدار بهره افزایش پیدا کند پاسخ تا حدودی بهبود پیدا می‌کند ولی قبل از رسیدن به جواب مطلوب سیستم ناپایدار می‌شود. پس یک کنترلر تناسبی به تنهایی برای کنترل این سیستم کافی نیست.

یک راه حل برای کنترل این سیستم این است که کنترلری طراحی کنیم که چند تا صفر نزدیک قطب‌های نامطلوب قرار دهیم به‌طوری‌که این صفرها اثر این قطب ها را ضعیف کنند. سپس قطب‌های مطلوب را در موقعیت مطلوب قرار می‌دهیم. چنین کنترلری **Notch filter** نامیده می‌شود. با استفاده از روش حذف تقریبی نیز می‌توان یکسری مشخصات مطلوب به سیستم داد. به طور مثال سعی کنید صفر ها را نزدیک قطب ها و در سمت چپ آنها قرار دهید چون در این صورت قطب ها به سمت چپ کشیده می‌شوند.

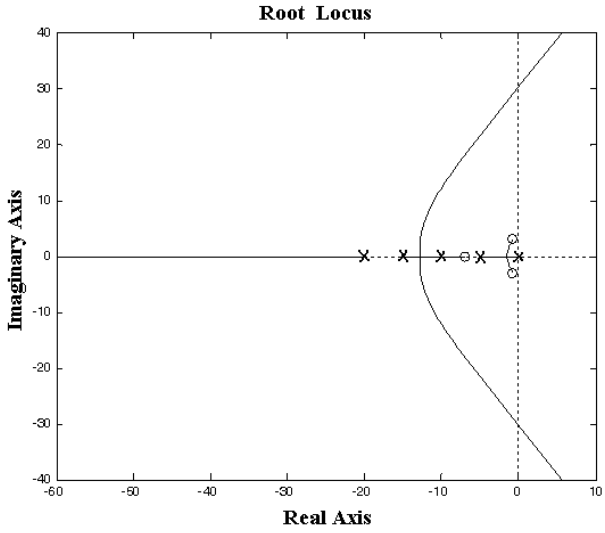
$$G_{Comp}(s)=\frac{s^2+1.5s+10}{s^2+20s+100}$$

همان‌طور که می‌بینید ریشه‌های صورت تقریباً مشابه قطب‌های مختلط مخرج سیستم می‌باشد. مخرج کنترل کننده دو عدد قطب در -10 دارد. حال با نوشتن دستورات زیر اثر این کنترل کننده را مشاهده می‌کنید:

```
numnotch = [1 1.5 10];
dennotch = [1 20 100];
newnum = conv(num,numnotch);
```

```
newden = conv(den,dennotch);
rlocus(newnum, newden);
```

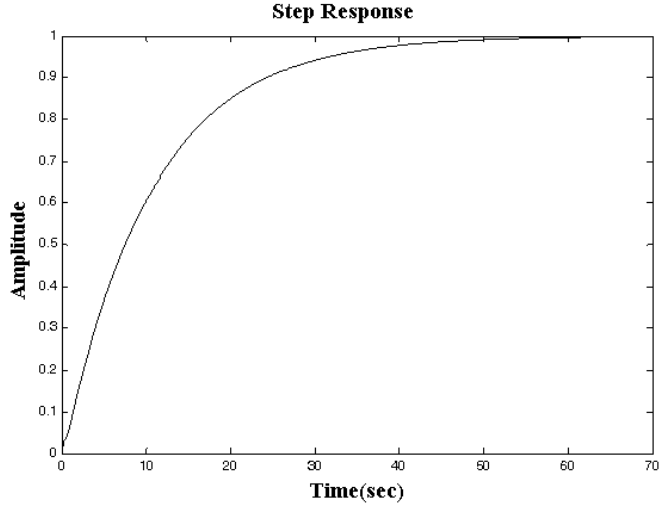
همان‌طور که از نمودار بر می‌آید می‌بینید که قطب‌های مختلط نزدیک محور موهومی تقریباً حذف می‌شوند و مکان هندسی ریشه ها کاملاً به سمت چپ کشیده می‌شود و این بدان معناست که **بهره**های **بزرگترK** را با حفظ پایداری می‌توان استفاده کرد ، حتی میرایی‌های بیشتر همراه با بهره‌های کوچکتر نیز قابل دسترسی است.



شکل (۲-۱۳) مکان هندسی ریشه‌ها

```
[k,poles]=rlocfind(newnum,newden)
[numc,denc] = cloop(k*newnum,newden,-1);
step(numc,denc)
```

نقطه‌ای را انتخاب کنید که بهره آن تقریباً برابر 200 باشد.



شکل (۲-۱۴) پاسخ پله سیستم حلقه بسته به ورودی پله

همانطور که می‌بینید پاسخ پله بهتری به‌دست می‌آید و زمان نشست و جهش کاهش پیدا می‌کند.

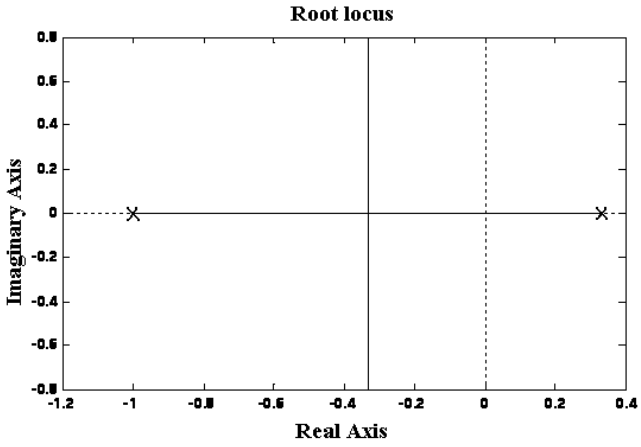
۲-۲-۹ حذف قطب و صفر[1]

وقتی سیستم حلقه باز یکسری قطب، در سمت راست محور موهومی دارد که باعث ناپایدار شدن سیستم می شود، یک ایده برای حل این مشکل این است که یکسری صفر درست در همان محل قطب ها قرار دهیم تا قطب‌های ناپایدار را حذف کند. ولی متاسفانه این روش اساساً اشتباه است. مساله این‌جاست که وقتی صفر به سیستم اضافه می‌شود دقیقاً قطب‌ها را حذف نمی‌کند. یک قسمت از مکان هندسی ریشه ها در نیم صفحه راست به دام می‌افتد. بنابراین باعث می‌شود که سیستم حلقه بسته ناپایدار گردد. البته به صورت دقیق‌تر باید گفت که چنین سیستمی در برابر شرایط اولیه واقعی غیر صفر دارای یک یا چند مود ناپایدار خواهد بود و در اصل قابل استفاده نیست. مثال زیر این مفهوم را بهتر نشان می‌دهد. فرض کنید که تابع تبدیل سیستم حلقه باز به صورت زیر باشد:

$$G(s)=\frac{1}{3s^2+2s-1}$$

```
num=1; den=[3 2 -1];
rlocus(num,den)
```

مکان هندسی ریشه‌های این سیستم حلقه باز به صورت زیر خواهد بود. از روی شکل مشخص است که یک قطب در سمت راست محور موهومی قرار گرفته است و با انتخاب هر بهره‌ای که قطب هنوز در سمت راست باشد، سیستم حلقه بسته ناپایدار خواهد بود. ولی با انتخاب درست بهره می‌توانیم سیستم را پایدار کنیم. اما ببینیم که سیستم چگونه با حذف صفر و قطب پایدار می‌گردد.

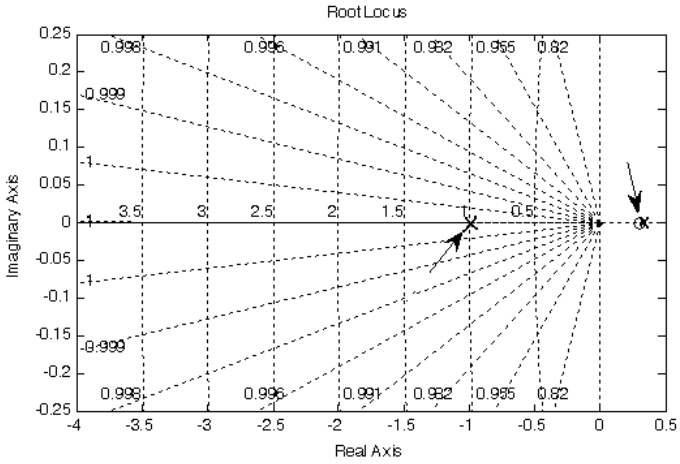


شکل (۲-۱۵) مکان هندسی ریشه‌ها

ازروی نمودار بالا مشخص است که قطب ناپایدار در حوالی 0.3 است. پس دستورات زیر را به ادامه M-file اضافه کنید:

[1] Pole-zero Cancellation

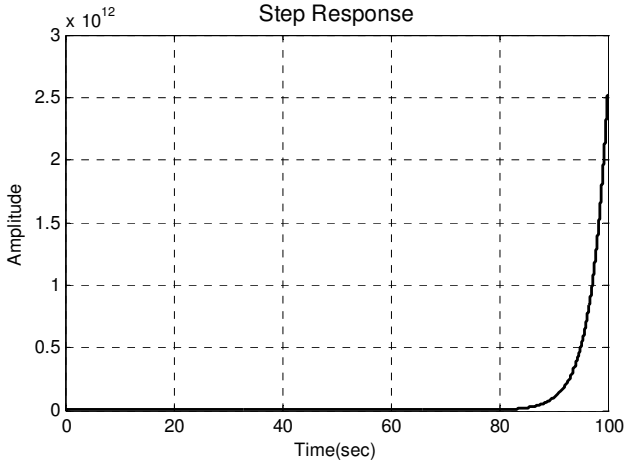
```
Zero=-0.3;
numOL=conv(num,[1 Zero]);
denOL=den;
[numCL,denCL]=cloop(numOL,denOL);
rlocus(numOL,denOL)
```



شکل (۲-۱۶) مکان هندسی ریشه‌ها

از مکان هندسی بالا مشخص است که قطب ناپایدار تقریباً حذف شده است. حال ببینیم پاسخ حلقه بسته به چه صورت می‌باشد. پس دستورات زیر را به ادامه M-file اضافه کنید:

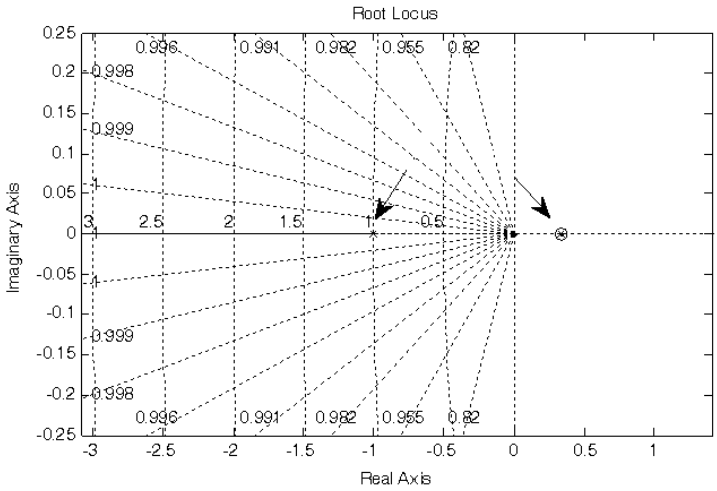
```
t=0:0.01:100;
step(numCL,denCL,t)
```



شکل (۲-۱۷) پاسخ پله سیستم حلقه بسته به ورودی پله

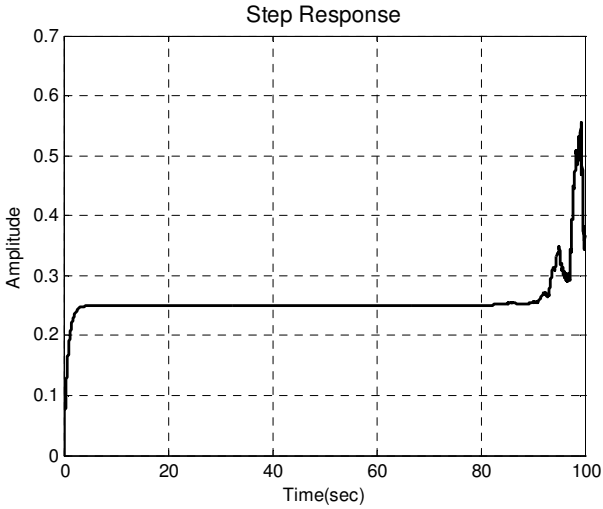
سیستم هنوز ناپایدار است. دلیل این امر این است که صفر اضافه شده دقیقاً قطب ناپایدار را حذف نکرده است. آیا ما می‌توانیم صفر را دقیقاً روی قطب قرار دهیم و سیستم را به این طریق پایدار کنیم؟ از روی تابع تبدیل

سیستم حلقه باز می‌بینیم که قطب دقیقاً در 1/3 است. حال اگر دقیقاً صفر را در 1/3 قرا دهیم چه اتفاقی می افتد؟ پس در M-file نوشته شده تغییر زیر را اعمال کنید: Zero=-1/3



شکل (۲–۱۸) مکان هندسی ریشه‌ها

با اجرای برنامه و مشاهده مکان هندسی ریشه می‌بینیم که صفر دقیقاً روی قطب قرار گرفته است. حال باید سیستم حلقه بسته را بررسی کنیم که آیا پایدار است یا خیر. پاسخ پله سیستم به صورت زیر است:



شکل (۲-۱۹) پاسخ پله سیستم حلقه بسته به ورودی پله

همان طور که مشاهده می‌شود سیستم هنوز ناپایدار بوده و به سمت بینهایت می‌رود با اینکه معادله ما ظاهراً پایدار است. بر اساس توضیحات قبلی در این متن، برای این پدیده توضیح مناسبی را ارائه کنید.

۲ – ۳ روش پاسخ فرکانسی

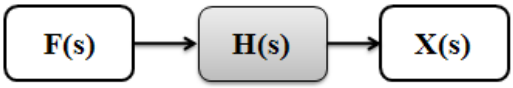
در بررسی سیستم‌های کنترل خطی به روش کلاسیک، دو شیوه اساسی برای تحلیل و بهبود عملکرد سیستم وجود دارد که بدون حل معادلات دیفرانسیل حاکم بر سیستم عمل می‌کنند. یکی از این روش ها، روش مکان هندسی ریشه ها و روش دیگر روش تحلیل پاسخ فرکانسی است. روش پاسخ فرکانسی نسبت به روش‌های دیگری که قبلاً مطالعه کردیم، تا حدودی نیاز به درک شهودی دارد و یکسری فواید دیگر نسبت به بقیه روش ها دارد که در ادامه توضیح داده می‌شود.

در تحلیل پاسخ فرکانسی برخلاف مکان ریشه، بهره سیستم و سایر پارامترهای آن ثابت فرض شده، تغییرات دامنه[1] و فاز[2] پاسخ سیستم نسبت به یک ورودی سینوسی بررسی می‌شود. تابع تبدیل $G(s)$ در پاسخ به تغییرات قطب های تابع تبدیل در نظر گرفته می‌شود.

[1] Magnitude
[2] phase

پاسخ فرکانسی روشی است که به بررسی پاسخ سیستم ها به ورودی نوسانی با فرکانس های مختلف می‌پردازد یا به بیان دیگر، *پاسخ فرکانسی*[1]*، پاسخ حالت پایا*[2] *به ورودی سینوسی (هارمونیکی) است، وقتی که فرکانس از صفر تا بینهایت تغییر می کند.*

اگر ورودی سینوسی به صورت $f(t)=F_0\sin\omega t$ باشد در آن صورت f(t) در حوزه لاپلاس به صورت $F(s)=\frac{F_0\omega}{s^2+\omega^2}$ است. از تئوری سیستم‌های خطی می‌دانیم پاسخ یک سیستم خطی پایدار به ورودی سینوسی، خود نیز سینوسی است. تفصیل بیشتر این موضوع در زیر آمده است:



$$X(s)=G(s).F(s) \Rightarrow X(s)=\frac{F_0\omega}{s^2+\omega^2}G(s)=\frac{F_0\omega}{s^2+\omega^2}\frac{B(s)}{A(s)}$$
$$X(s)=\frac{F_0\omega}{s^2+\omega^2}\times\frac{B(s)}{(s-p_1)(s-p_2)(s-p_3)...(s-p_n)} \quad (۱۲-۲)$$

اگر $p_1, p_2, ...p_n$ قطب های سیستم باشند با بسط کسر های جزئی داریم:

$$X(s)=\frac{a_1}{s-j\omega}+\frac{a_2}{s+j\omega}+\frac{b_1}{s-p_1}+\frac{b_2}{s-p_2}+...+\frac{b_n}{s-p_n} \quad (۱۳-۲)$$

$$L^{-1}\left[X(s)\right]=x(t)=a_1e^{j\omega t}+a_2e^{-j\omega t}+\sum_{i=1}^{n}b_ie^{p_it} \quad (۱۴-۲)$$

اگر سیستم پایدار باشد، در آن‌صورت $\lim_{t\to\infty}e^{p_it}\to 0$ و پاسخ حالت ماندگار سیستم به ورودی سینوسی برابر است با:

$$X_{ss}(t)=\lim_{t\to\infty}x(t)=a_1e^{j\omega t}+a_2e^{-j\omega t} \quad (۱۵-۲)$$

$$a_1=-a_2=\frac{F_0}{2j}G(j\omega)=\frac{F_0}{2j}\left|G(j\omega)\right|e^{j\phi} \quad (۱۶-۲)$$

$$X_{ss}(t)=\frac{F_0}{2j}\left|G(j\omega)\right|\left[e^{j(\omega t+\phi)}-e^{-j(\omega t+\phi)}\right]$$
$$X_{ss}=X_o\sin(\omega t+\phi)$$
$$\begin{cases}X_0=F_0\left|G(j\omega)\right|\\ \phi=\angle G(j\omega)\end{cases} \quad (۱۷-۲)$$

[1] Frequency Response
[2] Steady-State Response

بررسی چند نکته

۱-پاسخ حالت ماندگار یک سیستم دینامیکی خطی به ورودی سینوسی با فرکانس ω_0 یک پاسخ سینوسی با همان فرکانس تحریک می باشد.

۲- اختلاف فاز پاسخ، نسبت به ورودی سینوسی مستقیماً از رابطه $\phi = \angle G(j\omega_0)$ به‌دست می آید.

۳- دامنه پاسخ مستقیماً از رابطه $X_0 = F_0 \left|G(j\omega_0)\right|$، که در واقع حاصلضرب دامنه تحریک در $\left|G(j\omega_0)\right|$ است ، به‌دست می آید.

در اینجا ما می‌خواهیم بدانیم که چطور می‌توانیم از پاسخ فرکانسی حلقه باز سیستم برای پیش گویی رفتار حلقه بسته همان سیستم استفاده کنیم. برای رسم پاسخ فرکانسی، یک بردار فرکانسی که از صفر (موج DC) تا بینهایت تغییر می‌کند ایجاد می‌کنیم و خروجی تابع تبدیل در برابر این فرکانس ها را به‌دست می‌آوریم. اگر $G(s)$ تابع تبدیل حلقه باز سیستم و ω بردار فرکانسی باشد، نمودار $G(jw)$ و ω را رسم می‌کنیم.

با توجه به اینکه تابع پاسخ فرکانسی $G(jw)$، یک تابع مختلط از فرکانس می باشد، فرمتهای مختلفی برای تحلیل حوزه فرکانسی سیستم های خطی ارائه شده است:

دیاگرام بود[1]

در این روش دو منحنی رسم می‌گردد، یکی منحنی دامنه بر حسب فرکانس و دیگری منحنی زاویه فاز بر حسب فرکانس. این منحنی‌ها در مقیاس لگاریتمی رسم می شوند. در واقع محور عمودی به صورت خطی مقیاس بندی می‌شود و محور افقی به صورت غیرخطی با مقیاس لگاریتم فرکانس رسم می‌شود. بدین ترتیب می‌توان از گستره فرکانسی بسیار وسیعتری نسبت به مقیاس خطی برای نمایش پاسخ فرکانسی استفاده کرد و از طرفی فرکانس های پائین تر نیز دقیق تر و واضح تر نمایش داده می‌شوند.

دیاگرام های قطبی و نایکوئیست[2]

در دیاگرام قطبی، منحنی قسمت موهومی $G(j\omega)$ برحسب قسمت حقیقی آن در مختصات کارتزین و یا منحنی $\left|G(j\omega)\right|$ بر حسب زاویه فاز $G(j\omega)$ در مختصات قطبی ترسیم می‌شود. به عبارت دیگر، دیاگرام قطبی نگاشت همدیس[3] نیم خط مثبت محور $j\omega$ از صفحه s به صفحه $G(j\omega)$ است. از سوی دیگر، دیاگرام نایکوئیست در واقع نگاشت تمام محور $j\omega$ است.

چارت نیکولز

این روش نیز شامل رسم یک منحنی است. آن منحنی دامنه $G(j\omega)$ برحسب زاویه فاز آن در مقیاس لگاریتمی می باشد.

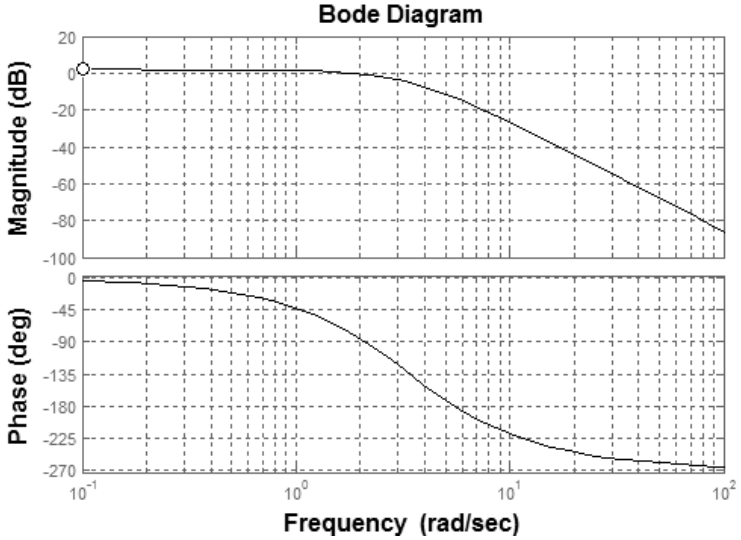
[1] Bode
[2] Nyquist
[3] Conformal Mapping

یک مزیت مشترک هر سه روش این است که با رسم پاسخ فرکانسی حلقه باز نتایج مهمی از رفتار سیستم حلقه بسته مانند پایداری نسبی آن، استنتاج می‌شود.

رسم نمودار بود

در تحلیل پاسخ فرکانسی، واحدی را که عموماً برای لگاریتم دامنه به‌کار می‌گیریم دسی بل است. توجه کنید که تابع تبدبل نسبت بین خروجی و ورودی است و متغیر های ورودی و خروجی لزوماً از یک واحد نمی‌باشند. برای مثال ورودی ممکن است ولتاژ باشد و خروجی رادیان بر ثانیه باشد. در نمودار بود، محور طولی بر حسب $rad/\sec$ و زاویه فاز بر حسب درجه و دامنه برحسب دسی بل(dB) است، که روی محور عمودی نمایش داده می‌شوند. برای رسم نمودار بود دستورات زیر را در یک m-file وارد کنید:

```
num = 50;
den = [1 9 30 40];
sys = tf(num,den);
bode(sys)
```



شکل (۲-۲۰) نمودار بود

لگاریتم دامنه یک تابع تبدیل به صورت زیر تعریف می‌شود:

$$v(db) = 20\log|G(j\omega)|$$

چند مثال از نمودار بود

اگر ما تابع تبدیل کنترلر و سیستم را داشته باشیم می‌توانیم نمودار بود هرکدام را به طور جداگانه رسم کرده، و سپس آنها را به‌طور گرافیکی با هم ترکیب کنیم. زیرا دیاگرام بود هرتابعی که به صورت حاصل‌ضرب چند تابع

پایه باشد جمع دیاگرام بود آنها می‌باشد. پس اگر ما تابع تبدیل توابع ساده را بدانیم درآن صورت می‌توانیم از آنها برای طراحی کنترلر استفاده کنیم.

$G(s) = G_1 G_2 G_3$

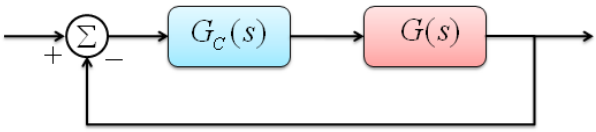
$G_1\{\nu_1, \Phi_1\}$

$G_2\{\nu_2, \Phi_2\}$

$G_3\{\nu_3, \Phi_3\}$

$\nu = \nu_1 + \nu_2 + \nu_3$

$\Phi = \Phi_1 + \Phi_2 + \Phi_3$

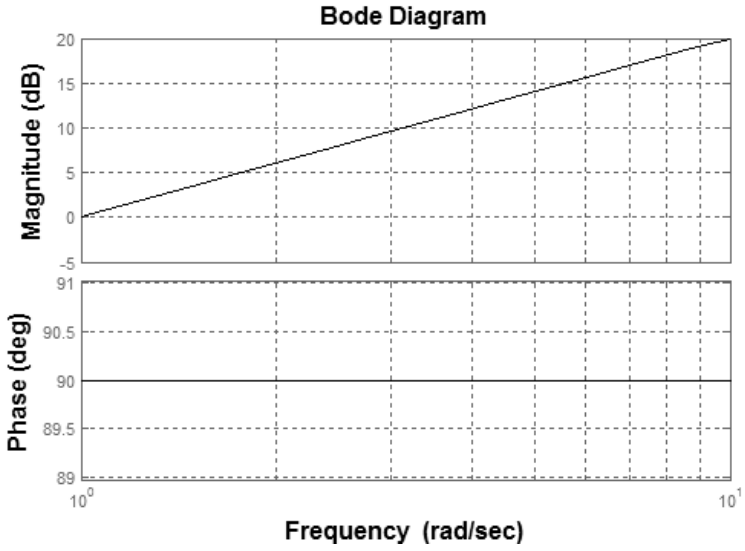



```
bode([1 0], 1)% G(s) = s
```

$G(s) = \tau s$

$$G(j\omega) = \tau j\omega \begin{cases} |G(j\omega)| = \tau\omega \\ \Phi = 90^0 \end{cases}$$

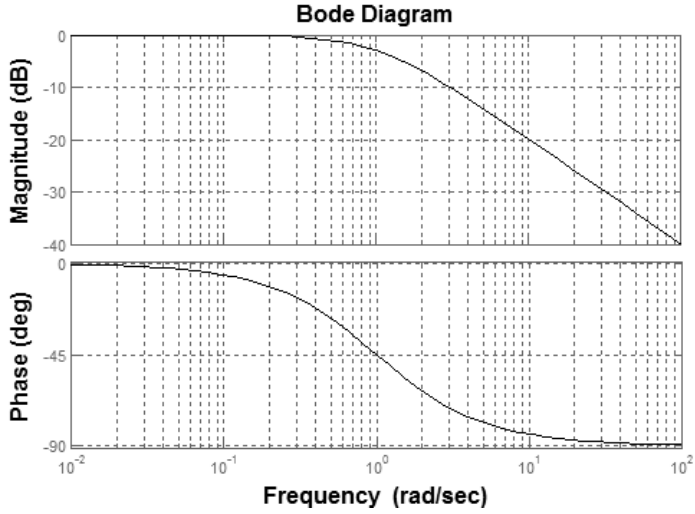
$\nu = 20Log(\tau\omega) = 20Log\omega + 20Log\tau$



شکل (۲-۲۱) نمودار بود

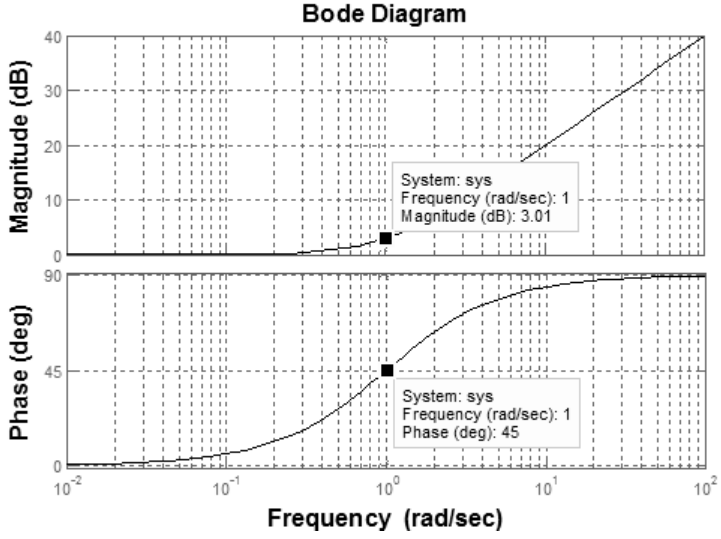
توجه: وقتی تابع تبدیل معکوس می شود، مقدار زاویه و مقدار شیب خط دریک علامت منفی ضرب می شود.

```
bode(1, [1 1]) % G(s) = 1/(s+1)
```



شکل (۲-۲۲) نمودار بود

```
bode([1 1], 1) % G(s) = s+1
```



شکل (۲-۲۳) نمودار بود

در توابع تبدیل زیر موقعیت صفرها و قطب‌ها را تغییر دهید تا پاسخ های فرکانسی مختلف را ببینید، تا متوجه شوید که چگونه با جابه‌جا کردن صفر ها و قطب ها پاسخ تغییر می‌کند.

```
bode(1, [1 0])
bode(1,[ 1, 0.1])
bode([1, 0.1],1)
bode(1,[ 1, 10])
bode([1, 10],1)
```

توجه

۱. برای یک عدد داده شده، مقدار دسی بل معکوس آن در یک منفی ضرب می‌شود.

۲. اگر یک عدد دو برابر شود، مقدار دسی بل آن 6.02dB افزایش پیدا می‌کند.

۳. اگر یک عدد در فاکتور 10 ضرب شود، مقدار دسی بل آن 20dB افزایش پیدا می‌کند.

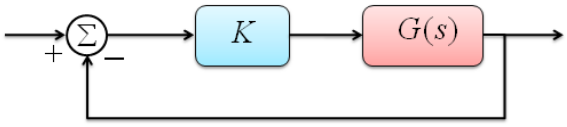
حد تقویت و حد فاز

در روشهای پاسخ فرکانسی برای بررسی مشخصات گذرا، سرعت و اصطلاحاً پایداری نسبی سیستمهای کنترل دو پارامتر زیر تعریف شده است:

۱- حد تقویت (بهره) [1]

۲- حد فاز [2]

سیستم زیر را در نظر بگیرید:



k بهره ثابت یا متغیر و $G(s)$ تابع تبدیل سیستم تحت بررسی می‌باشد.

حد تقویت به صورت زیر تعریف می‌شود:

چقدر می‌توانیم بهره حلقه باز سیستم را تغییر دهیم پیش از آنکه سیستم حلقه باز ناپایدار گردد. سیستم هایی با حد تقویت بالا قادرند قبل از اینکه سیستم در حالت حلقه بسته ناپایدار شود در برابر تغییرات بزرگ در پارامترهای سیستم مقاومت کنند. پس می‌توان گفت هرچه حد تقویت بزرگتر بوده و از 1 فاصله بگیرد، سیستم پایدارتر می‌شود.

توجه: بهره واحد برابر صفر dB می‌باشد.

حد فاز به صورت زیر تعریف می‌شود:

چقدر می‌توانیم تاخیر فاز سیستم حلقه باز را افزایش دهیم قبل از آنکه سیستم حلقه بسته ناپایدار گردد.

حال سوالی که پیش می‌آید این است که چگونه از روی نمودار بود حد فاز و حد بهره را اندازه گیری کنیم؟

حد فاز را چگونه اندازه گیری کنیم؟ برای اندازه گیری حد فاز باید اختلاف بین منحنی فاز و فاز 180^0 را در نقطه ای متناظر با فرکانسی که منحنی حد تقویت محور افقی را قطع می‌کند (ω_g یا فرکانسی که بهره $0dB$ دارد) حساب کنیم.

[1] *Gain Margin*

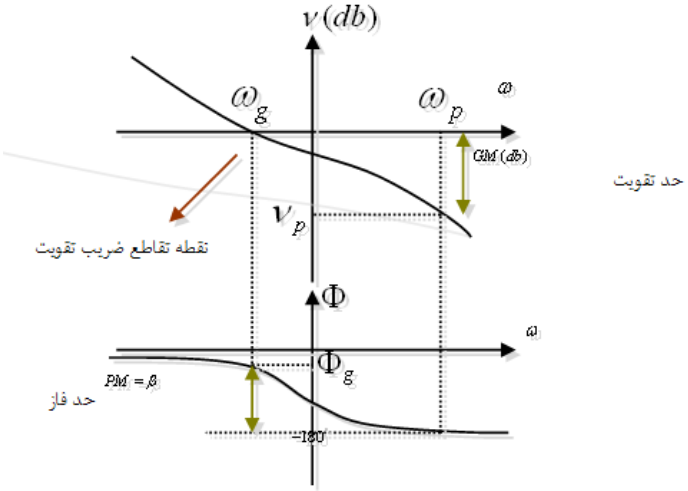
[2] *Phase Margin*

$$\left|G(j\omega_g)\right|=1\Rightarrow \omega_g$$
$$\angle G(j\omega_g)=\Phi_g$$
$$PM=\beta=180^\circ+\Phi_g \qquad (۱۸-۲)$$

۲- **حد تقویت را چگونه اندازه گیری کنیم؟** مشابه حالت بالا برای اندازه‌گیری حد تقویت باید اختلاف بین منحنی دامنه و بهره $0\,dB$ را در نقطه‌ای متناظر با فرکانسی که منحنی حد فاز، فاز -180 درجه را قطع می کند حساب کنیم.

(ω_p)

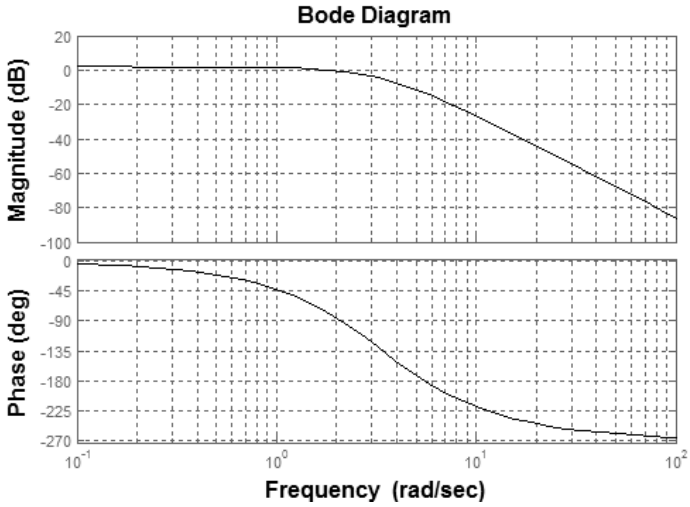
$$\angle G(j\omega)=-180^\circ\Rightarrow \omega=\omega_p$$
$$GM=\frac{1}{\left|G(j\omega_p)\right|} \qquad (۱۹-۲)$$



نکته: حد فاز منفی و حد تقویت کوچکتر از 1 به معنای این است که سیستم ناپایدار است. معمولاً حد تقویت 2 و حد فاز 60~45 درجه برای سیستم های کنترلی پیشنهاد می‌شود.

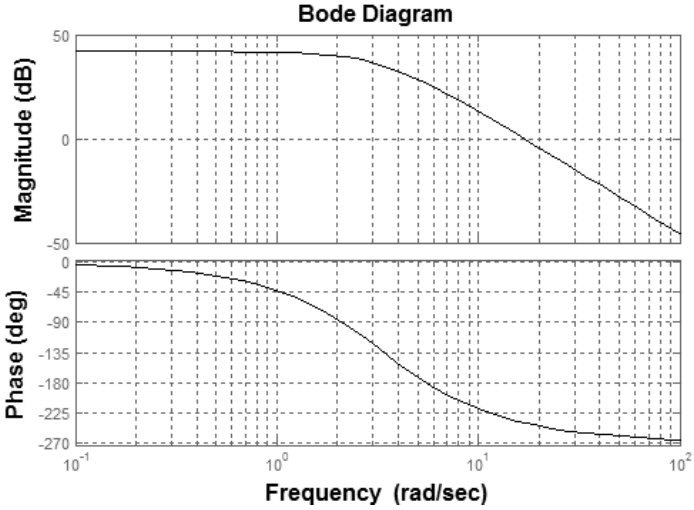
اضافه کردن بهره فقط نمودار دامنه را به سمت بالا شیفت می‌دهد که این معادل تغییرات در راستای محور y است، پس محل تقاطع منحنی دامنه با محور افقی تغییر می‌کند. از روی این محل تقاطع یعنی فرکانس ω_g می‌توانیم مقدار حد فاز را به‌دست آوریم. به طور مثال نمودار بود زیر را رسم کنید:

```
num = 50;
den = [1 9 30 40];
sys = tf(num,den);
bode(sys)
```



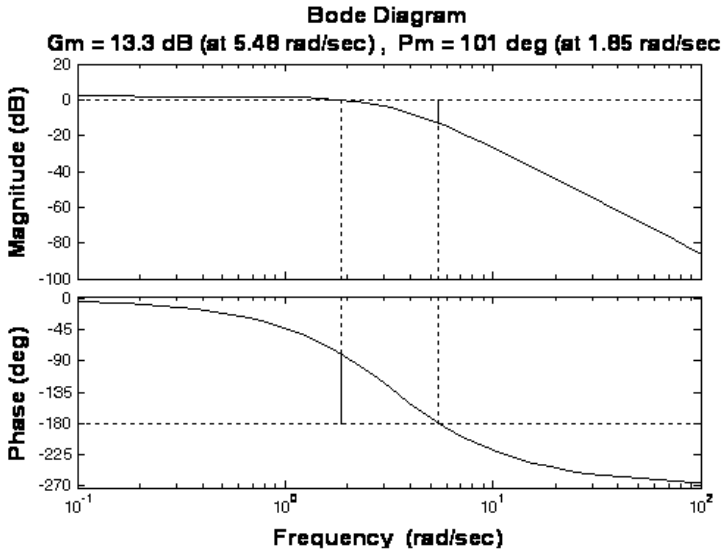
شکل (۲-۲۴) نمودار بود

شما می‌بینید که حد فاز حدود 100 درجه است. حال بهره 100 را به آن اضافه کنید. سپس دستور زیر را وارد کنید. نمودار بود به صورت زیر خواهد بود: bode(100*sys)



شکل (۲-۲۵) نمودار بود

همان‌طورکه در شکل دیده می‌شود، نمودار فاز همانند حالت قبل است ولی نمودار دامنه به اندازه 40dB به سمت بالا شیفت پیدا کرده است. حاشیه فاز دراین حالت حدود 60deg- است. مقدار حد تقویت و فاز را با استفاده از دستورات MATLAB زیر می‌توان به‌دست آورد. دستور Margin به صورت گرافیکی این مقادیر را نشان می‌دهد.



شکل (۲-۲۶) نمودار بود برای نمایش مقدار حد تقویت و فاز

نکته: در روشهای پاسخ فرکانسی همیشه منحنی پاسخ فرکانسی برای تابع تبدیل حلقه باز سیستم کنترل (حلقه بسته) رسم می‌گردد و آنگاه مشخصات سیستم کنترل مدار بسته (مانند پایداری، سرعت، و دقت...) از روی منحنی پاسخ فرکانسی تابع تبدیل مدار باز به‌دست آورده می‌شود.

کاربرد روشهای پاسخ فرکانسی در طراحی و تحلیل سیستم های کنترل فیدبک:

- بررسی مشخصات حالت گذرا مانند سرعت (پایداری نسبی) سیستمهای کنترل در دیاگرامهای نایکوئیست و بود
- به‌دست آوردن مشخصات پاسخ تابع تبدیل مدار فرکانسی بسته از روی مشخصات پاسخ فرکانسی مدار باز

پهنای باند فرکانسی[1]

از کمیت های حوزه فرکانسی می‌توان فرکانس قطع و پهنای باند را نام برد. فرکانس قطع، فرکانسی است که به ازای آن دامنه تابع تبدیل فرکانسی حلقه بسته برابر 3dB- باشد. بنابراین سیستم حلقه بسته با این مشخصه مولفه سیگنالی که فرکانس هایی بزرگتر از فرکانس قطع دارند را فیلتر می‌کند و مولفه هایی که فرکانس کمتر از فرکانس قطع دارند را انتقال می‌دهد. گستره بین فرکانس صفر تا فرکانس قطع که در آن اندازه تابع تبدیل حلقه بسته از 3dB- کمتر نمی‌شود را پهنای باند فرکانسی سیستم می‌گویند. در بسیاری از سیستم های کنترلی سیگنال ورودی ممکن است مقدار قابل توجهی نویز داشته باشد یا منبع تولید نویز در داخل سیستم کنترلی باشد، بنابراین برای تضعیف نویز، سیستم کنترلی را طوری طراحی می‌کنند که پهنای باند معین به همین منظور

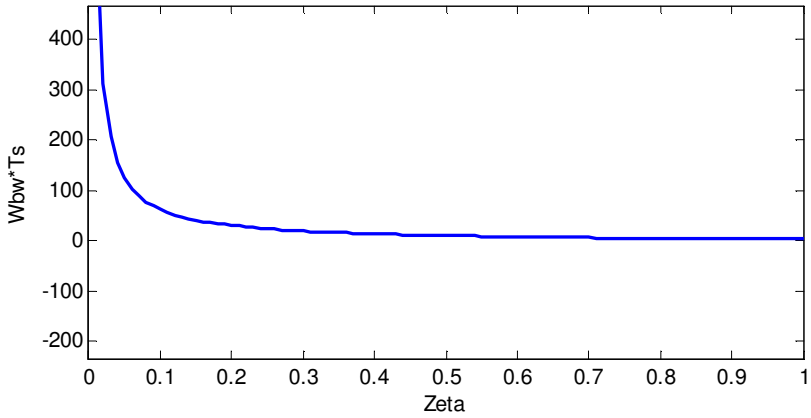
[1] Bandwidth frequency

داشته باشند. برای سیستم های مرتبه دوم پهنای باند فرکانسی سیستم به صورت ریاضی از ضریب میرایی و فرکانس طبیعی سیستم به صورت زیر به‌دست می‌آید.

استفاده از زمان نشست T_s برای به‌دست آوردن پهنای باند فرکانسی

$$w_{BW} = w_n\sqrt{(1-2\zeta^2)+\sqrt{4\zeta^4-4\zeta^2+2}}$$
$$w_n = \frac{4}{T_s.\zeta}$$
$$w_{BW} = \frac{4}{T_s.\zeta}\sqrt{(1-2\zeta^2)+\sqrt{4\zeta^4-4\zeta^2+2}} \qquad (۲۰-۲)$$

در شکل زیر $W_{BW}.T_s$ بر حسب ζ رسم شده است.



شکل (۲-۲۷) تغییرات $W_{BW}.T_s$ بر حسب ζ

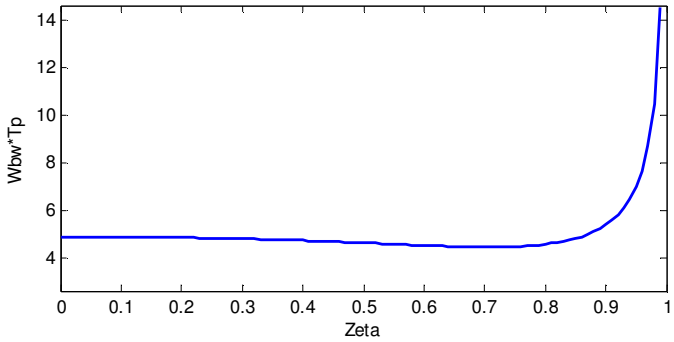
استفاده از زمان پیک T_p برای به‌دست آوردن پهنای باند فرکانسی

برای یک فرکانس طبیعی داده شده، زمان پیک[1] با افزایش نسبت میرایی افزایش پیدا می‌کند و همچنین پهنای باند با افزایش نسبت میرایی کاهش می‌یابد. پس افزایش پهنای باند باعث افزایش سرعت پاسخ می‌شود.

$$w_n = \frac{\pi}{T_P.\sqrt{1-\zeta^2}}$$
$$w_{BW} = \frac{\pi}{T_P.\sqrt{1-\zeta^2}}\sqrt{(1-2\zeta^2)+\sqrt{4\zeta^4-4\zeta^2+2}} \qquad (۲۱-۲)$$

در شکل زیر $W_{BW}.T_p$ بر حسب ζ رسم شده است.

[1] Peak Time



شکل (۲-۲۸) تغییرات $W_{BW}.T_p$ بر حسب ζ

```
function[] = wbw()
DR = input('Damping Ratio?    ');
n = 0;
n = input('Settling Time(0) or Peak Time(1)?   ');
if n == 0
        ts = input('Settling Time?   ');
        temp1 = sqrt((4*DR^4) - (4*DR^2) +2);
        temp2 =  1- (2*DR^2);
        temp3 = 4/(ts*DR);
        Wbw = temp3*sqrt(temp1 + temp2)
        end
if n ==1
        ts = input('Peak Time?   ');
        temp1 = sqrt((4*DR^4) - (4*DR^2) +2);
        temp2 =  1- (2*DR^2);
        temp3 = ts*sqrt( 1- DR^2);
        temp4 = pi/temp3;
        Wbw = temp4*sqrt(temp1 + temp2)
        end
```

به منظور روشن ساختن اهمیت پاسخ فرکانسی نشان می‌دهیم که چطور خروجی سیستم با ورودی‌های فرکانسی مختلف تغییر می‌کند. ما نشان خواهیم داد که ورودی های سینوسی با فرکانس های کمتر از W_{bw} (پهنای باند فرکانسی) به طور منطقی و خوب توسط سیستم دنبال می‌شوند و موج های سینوسی با فرکانس‌های بیشتر از W_{bw} با ضریب 0.707 یا بیشتر در دنبال کردن ضعیف‌تر می‌شود؛ یعنی در فاز شیفت پیدا می‌کنند.

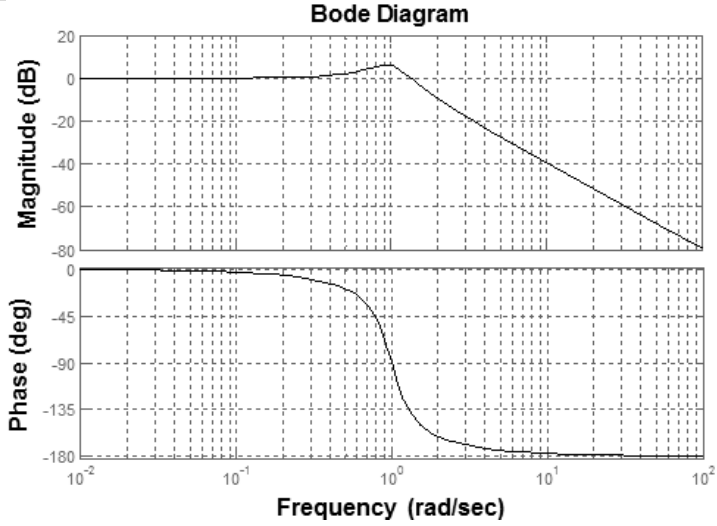
نکته: برای آن‌که سیستم ورودی‌های دلخواه را به طور دقیق دنبال کند باید دارای پهنای باند بزرگی باشد. البته توجه کنید که طراحی سیستم با پهنای باند بزرگ دارای دو مسأله مهم است. اول اینکه پهنای باند بزرگ از نظر نویز در سیستم موجب بروز مشکلاتی خواهد گردید (چرا که سیستم با پهنای باند بزرگ تمایل دارد نویز را به

عنوان ورودی دنبال کند) و دوم اینکه عناصری که در ساخت سیستم با پهنای باند بزرگ استفاده می‌شوند دارای عملکرد بالا بوده و در نتیجه گران می‌باشند. مثلاً سیستم $\frac{1}{s+1}$ که باند فرکانسی $0 \le w \le 1$ دارد نسبت به سیستم $\frac{1}{3s+1}$ که باند فرکانسی $0 \le w \le 0.33$ دارد ورودی دلخواه را بهتر ردگیری می‌کند. فرض کنید که سیستمی با تابع تبدیل حلقه بسته زیر را اختیار داریم:

$$G(s) = \frac{1}{s^2 + 0.5s + 1}$$

در ابتدا اجازه بدهید پهنای باند فرکانسی را از نمودار بود به‌دست آوریم. پس دستورات زیر را در یک m-file وارد کنید:

```
num = 1;
den = [1 0.5 1];
sys = tf(num,den);
bode (sys)
```



شکل (۲-۲۹) نمودار بود

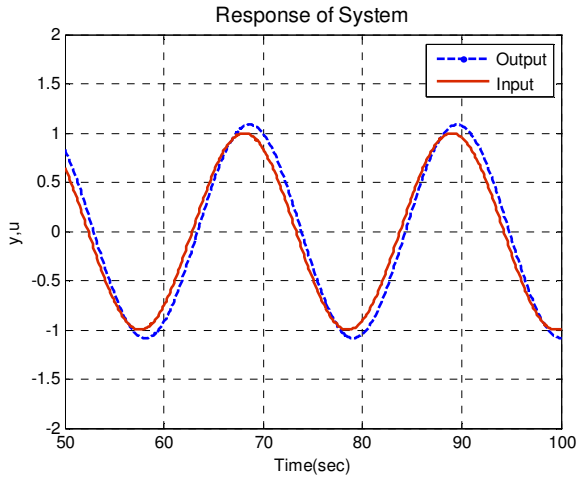
از آنجایی که این تابع تبدیل سیستم حلقه بسته است پس فرکانس قطع، فرکانس متناظر با بهره 3dB- می‌باشد که حدود 1.4rad/s است. پس پهنای باند فرکانسی حدود 1.4rad/s است. از روی نمودار مشخص است که برای فرکانس ورودی 3rad/s دامنه حدود 20dB- است و فاز نزدیک 180- می‌باشد.یعنی خارج از فاز[1] است و برای فرکانس 0.3rad/s مقدار فاز و بزرگی ناچیز می‌باشد. ما می‌توانیم از دستور lsim برای شبیه سازی خروجی به ورودی های سینوسی استفاده کنیم.

[1] Out of phase

در ابتدا فرض کنید که ورودی سینوسی با فرکانس کمتر از W_{bw} است و به خاطر داشته باشید که می‌خواهیم مقدار خطای حالت ماندگار رامشاهده کنیم.

```
w = 0.3;
num = 1;
den = [1 0.5 1];
sys = tf(num,den);
t = 0:0.1:100;
u = sin(w*t);
[y,t] = lsim(sys,u,t);
plot(t,y,t,u)
axis([50,100,-2,2])
```

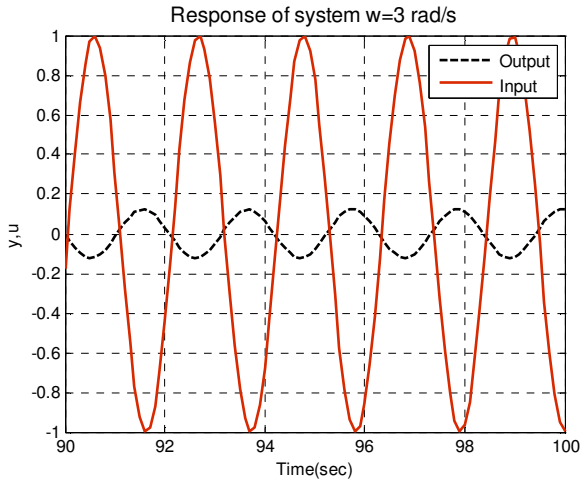
همانطور که انتظار داشتیم، خروجی سیستم به خوبی ورودی را دنبال می‌کند و خطای حالت ماندگار کم بوده و تا حدودی نسبت به ورودی تأخیر فاز دارد.



شکل (۲-۳۰) نمایش خروجی و ورودی سینوسی

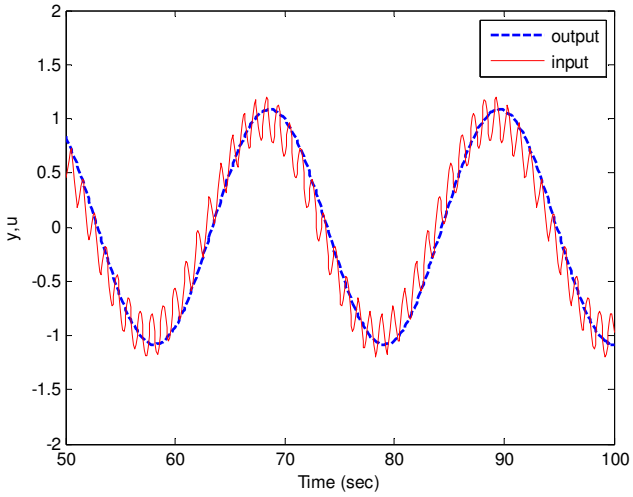
حال فرض کنید فرکانس ورودی ما بیشتر از فرکانس W_{bw} باشد، مثلاً 3rad/s. خروجی سیستم به صورت زیر به‌دست می‌آید:

```
w = 3;
num = 1;
den = [1 0.5 1];
sys = tf(num,den);
t = 0:0.1:100;
u = sin(w*t);
[y,t] = lsim(sys,u,t);
plot(t,y,t,u)
axis([90, 100, -1, 1])
```



شکل (۲-۳۱) خروجی سیستم به فرکانس های ورودی بیشتر از فرکانس W_{bw}

همانطور که انتظار داشتیم دامنه 0.1 ورودی است و خروجی نسبت به ورودی 180 درجه اختلاف فاز دارد، یعنی 180 درجه تأخیر فاز دارد. حال نتایج را برای حالتی که ورودی ما نویزی است تکرار می‌کنیم، انتظار داریم که در خروجی نویز ظاهر نشود.



شکل (۲-۳۲) خروجی سیستم به ورودی نویزی

عملکرد سیستم حلقه بسته

به منظور پیش بینی عملکرد سیستم حلقه بسته از پاسخ فرکانسی سیستم حلقه باز چند مفهوم باید کاملاً روشن شود:

۱- اگر قصدمان این باشد که طراحی را با نمودار بود انجام دهیم، سیستم باید در حالت حلقه باز پایدار باشد.

۲- اگر $w_g < w_p$، در آن صورت سیستم حلقه بسته پایدار خواهد بود.

۳- برای سیستم مرتبه دوم، رابطه بین ضریب میرایی، پهنای باند فرکانسی، و زمان نشست در صفحات قبل آورده شده است.

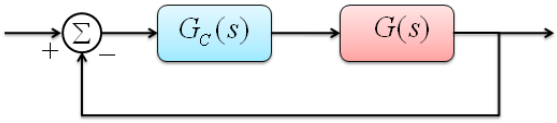
۴- برای سیستم های مرتبه دوم، اگر حد فاز بین 30 تا 60 درجه باشد، ضریب میرایی تقریباً بین $0.3 \le \zeta \le 0.6$ خواهد بود، یعنی رابطه بین حد فاز و ضریب میرایی تقریبا خطی است و در واقع داریم:

$$\zeta \approx \frac{PM}{100}$$

۵- با یک برآورد تقریبی می‌توانید بگویید که پهنای باند فرکانسی تقریباً برابر فرکانس طبیعی است.

۶- فرکانس رزونانس در سیستم های مرتبه دوم عبارت‌است از: $w_r = w_n\sqrt{1-2\zeta^2}$، که به ازای $\zeta \to 0$ فرکانس رزونانس به سمت w_n میل می‌کند و برای $0 \le \zeta \le 0.7$ فرکانس رزونانس کمتر از فرکانس طبیعی میرا شده $w_d = w_n\sqrt{1-\zeta^2}$ است.

اجازه بدهید که از این مفاهیم در طراحی یک کنترلر استفاده کنیم.



$G_c(s)$ کنترلر و $G(s) = \dfrac{10}{1.25s+1}$ تابع تبدیل سیستم می‌باشد.

طراحی باید طوری باشد که سیستم کنترل شده مشخصات زیر را داشته باشد:

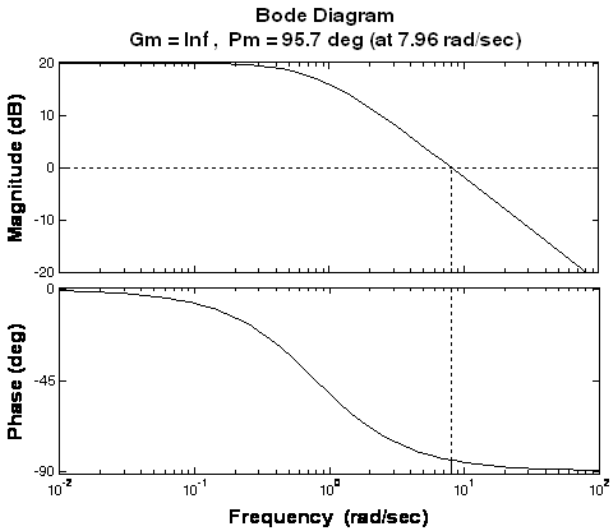
۱- خطای حالت ماندگار صفر باشد.

۲- حداکثر جهش کمتر از 40% باشد.

۳- زمان نشست کمتر از 2 ثانیه باشد.

دو راه حل برای حل این مساله وجود دارد یکی روش گرافیکی و دیگری روش عددی. ما از روش گرافیکی MATLAB استفاده می‌کنیم. در یک m-file جدید دستورات زیر را وارد نموده و به نمودار بود حاصل توجه کنید.

```
num = 10;
den = [1.25,1];
sys = tf(num,den);
bode(sys)
```



شکل (۲-۳۳) نمودار بود Pm ,Gm

نکته: نوع سیستم را می‌توان از شیب نمودار لگاریتم دامنه در فرکانس های پائین تعیین کرد به این صورت که:

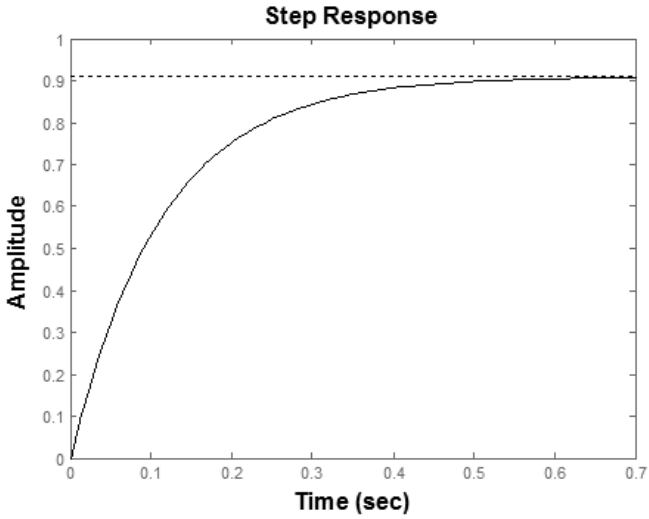
- سیستم نوع صفر شیب برابر 0dB بر دهه (هر واحد بر روی محور فرکانس که نسبت فرکانسی ۱۰ را نمایش می‌دهد را دهه یا decade نامند.)
- سیستم نوع یک -20dB بر دهه
- سیستم نوع دو -40dB بر دهه

بسیاری از مشخصات سیستم از روی نمودار بود قابل دستیابی است. پهنای باند حدود 10rad/sec است. از آنجا که پهنای باند تقریباً برابر فرکانس طبیعی است (برای سیستم مرتبه اول)، زمان خیز برابر است با:

$$T_r = \frac{1.8}{w_n} = 0.18 \approx 0.2\,\text{sec}$$

حاشیه فاز این سیستم تقریباً 95 درجه است و از آنجایی که سیستم مرتبه اول است پس جهش ندارد. با فرض واحد بودن فیدبک برای به‌دست آوردن پاسخ پله دستورات زیر را وارد کنید:

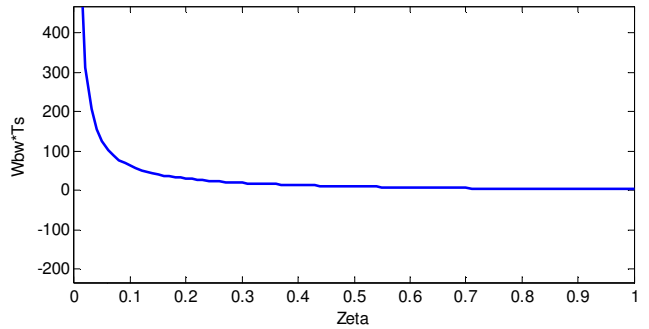
```
sys_cl = feedback(sys,1);
step(sys_cl)
```



شکل (۲-۳۴) پاسخ پله سیستم با فیدبک واحد

همان‌طور که از شکل بر می‌آید خطای حالت ماندگار برابر 10% و زمان خیز حدود 0.2 است و جهش نیز صفر است. اکنون نیاز داریم برای این سیستم یک کنترلر طراحی کنیم تا معیار های طراحی برآورده شود. کنترلر PI را انتخاب می‌کنیم تا خطای حالت ماندگار صفر شود: $\frac{K_p s + K_I}{s}$

درابتدا نیاز داریم ضریب میرایی متناسب با جهش 40% را به‌دست آوریم. در نتیجه‌ی استفاده از رابطه $M_p = e^{-\frac{\pi\zeta}{\sqrt{1-\zeta^2}}}$ مقدار $\zeta = 0.28$ به‌دست می‌آید. بنابراین حد فاز حداقل باید حدود 30deg باشد. از روی نمودار می‌بینیم که $T_s \times \omega_{bw} \simeq 21$ است.

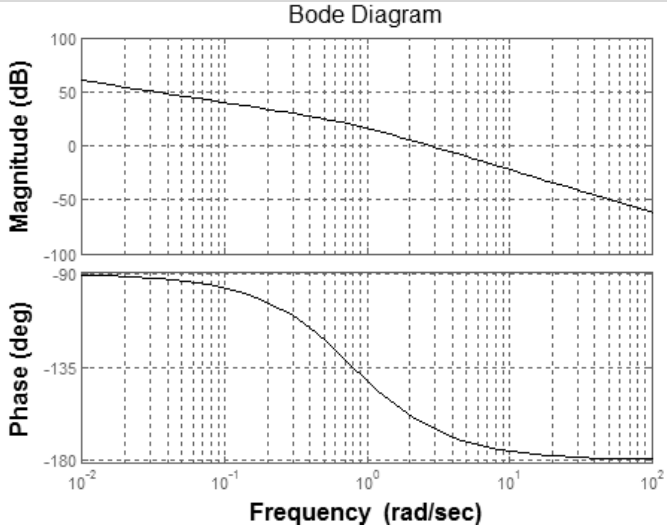


شکل (۲-۳۵) تغییرات $W_{BW}.T_s$ بر حسب ζ

اگر بخواهیم زمان نشستی حدود 2 ثانیه داشته باشیم باید پهنای باند بزرگتر یا مساوی 10 داشته باشیم. تا این مرحله حد فاز و پهنای باند فرکانس را می‌دانیم. پس اکنون می‌توانیم طراحی خود را شروع کنیم. توجه کنید که

ما داریم به نمودار بود حلقه باز نگاه می‌کنیم بنابراین پهنای باند متناظر با فرکانسی خواهد بود که بهره آن تقریباً 7dB- است. حال ببینیم قسمت انتگرالی چگونه روی پاسخ تاثیر می‌گذارد (فقط ترم انتگرالی را قرار دهید).

```
num = 10;
den = [1.25 1];
plant = tf(num,den);
numPI = 1;
denPI = [1 0];
numc=conv(num,numPI);
denc=conv(den,denPI);
sys=tf(numc,denc);
bode(sys)
```

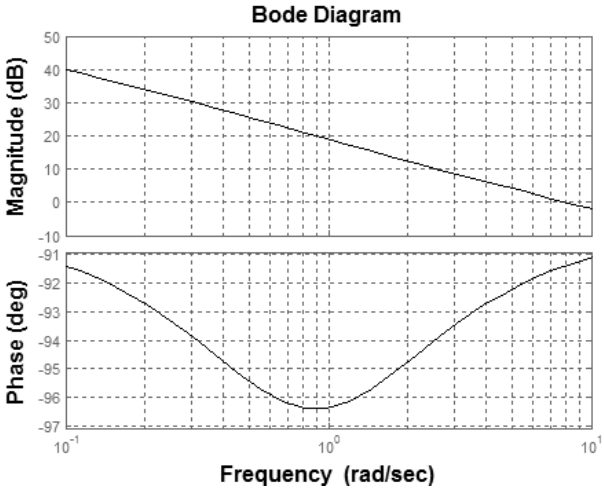


شکل (۲-۳۶) نمودار بود

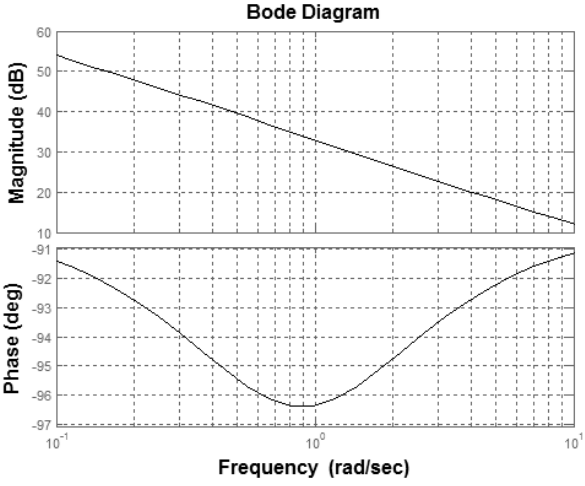
در این حالت حد فاز و پهنای باند بسیار کوچک می‌شود .از اینرو، یک بهره و یک فاز همراه با یک صفر اضافه می‌کنیم. اجازه بدهید که صفر را در 1- قرار دهیم و ببینیم چه اتفاقی می‌افتد.

```
num = 10;
den = [1.25 1];
k=1;
plant = tf(num,den);
numPI = k*[1 1];
denPI = [1 0];
numc=conv(num,numPI);
denc=conv(den,denPI);
sys=tf(numc,denc);
bode(sys)
```

وجود صفر در 1- و یک بهره واحد جواب قابل قبولی به دست می‌دهد. حد فاز بیشتر از 60 درجه، حتی جهشی کمتر از چیزی که انتظار داشتیم و پهنای فرکانسی تقریباً برابر 11rad/sec می‌شود که پاسخ قابل قبولی است. اجازه بدهید پهنای باند فرکانسی را بزرگتر کنیم. بدون اینکه حد فاز تغییر کند مقدار بهره را برابر 5 قرار می‌دهیم تا ببینیم چه اتفاقی می‌افتد. این باعث می‌شود که منحنی دامنه به سمت بالا شیفت پیدا کند ولی فاز در همان حالت باقی بماند.



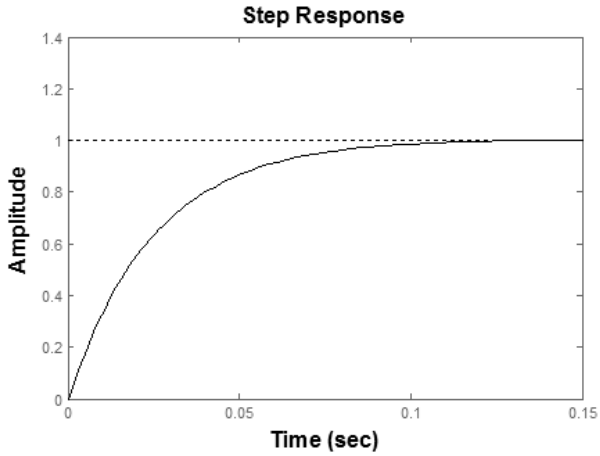
شکل (۲-۳۷) نمودار بود



شکل (۲-۳۸) نمودار بود

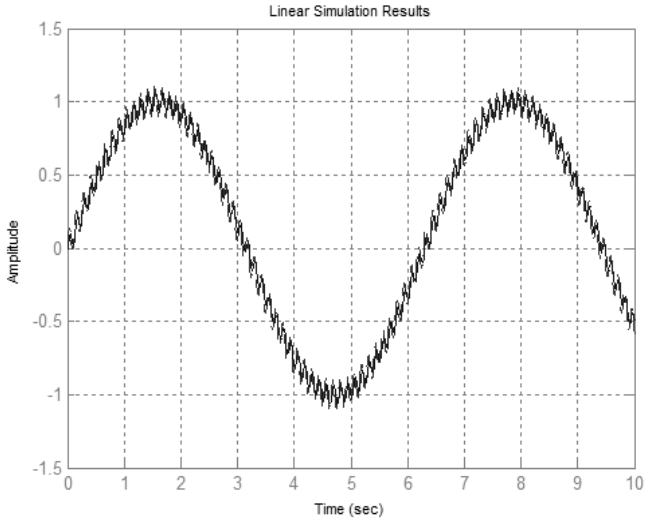
به نظر می‌رسد که جواب بسیار خوب است. حال پاسخ پله حلقه بسته را مشاهده کنید تا نتایج را تائید نماید.

```
sys_cl = feedback(sys,1);
step(sys_cl)
```



شکل (۲-۳۸) پاسخ پله سیستم حلقه بسته

همانطور که می‌بینیم پاسخ بهتر از چیزی است که انتظار می‌رفت. حال باید خروجی سیستم را به ورودی نویزی نیز بررسی کنیم.



شکل (۲-۴۰) خروجی سیستم به ورودی نویزی

ملاحظه می‌کنیم که از آنجا که پهنای باند سیستم کمی زیاد است (البته کلمه زیاد بی مفهوم است چرا که انتخاب پهنای باند به طراح بستگی دارد. ممکن است روی یک سیستم نویز چندانی نباشد و برای طراح مساله ردگیری اهمیت بیشتری داشته باشد.) سیستم نویز را هم ردگیری کرده است، که این امر نامطلوبی است. حال دنباله مساله را به شما واگذار می‌کنیم. می‌توانید پهنای باند را کم کنید. البته توجه کنید که در اثر این کار سرعت ردگیری کاهش می‌یابد!

فصل سوم

طراحی و تحلیل سیستم‌های کنترلی در فضای حالت و دیجیتال

۳ – ۱ تحلیل و طراحی سیستم‌های کنترل در فضای حالت

مدل‌سازی بسیاری از سیستم‌های دینامیکی پیوسته، معمولاً به دسته‌ای از معادلات دیفرانسیل خطی یا غیرخطی منجر می‌شود. پس از اعمال روش‌های خطی‌سازی، می‌توان با روشهای استانداردی معادلات حاصل را به صورت دسته‌ای از n معادله مرتبه اول که در مجموع بیانگر رفتار یک سیستم مرتبه n خواهند بود، تبدیل نمود. این نحوه نمایش سیستمها را روش فضای حالت[1] می‌نامند. روش‌های زیادی برای نمایش یک سیستم در فضای حالت وجود دارد. شکل مشترک همهٔ این بیان‌ها به صورت دوگروه از معادلات، موسوم به معادلات حالت و معادلات خروجی برای یک سیستم با چند ورودی و چند خروجی، در حالت کلی، به صورت زیر می‌باشد:

$$\begin{aligned} \dot{\boldsymbol{x}}(t) &= \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{u}(t) \\ \boldsymbol{y}(t) &= \boldsymbol{C}\boldsymbol{x}(t) + \boldsymbol{D}\boldsymbol{u}(t) \end{aligned} \qquad (۱-۳)$$

که در آنها، x(t) یک بردار $n\times 1$ موسوم به بردار حالت سیستم است. در سیستم‌های مکانیکی، بردار x(t) معمولاً شامل موقعیت‌ها و سرعت‌های جابه‌جایی و دورانی می‌باشد. u(t) نیز بردار m بعدی نمایش دهنده ورودی است که معمولاً در سیستم‌های مکانیکی شامل گشتاورها و نیروهای وارده از عملگرهای کنترلی و یا اعتشاشات وارده از محیط بیرونی به سیستم است. y برداری l بعدی است، که نمایش دهنده خروجی سیستم است. منظور از خروجی، متغیرهائی است که توسط سنسورها مستقیماً اندازه‌گیری می‌شوند. ماتریس‌های $A(n\times n), B(n\times m), C(l\times n), D(l\times m)$ نیز رابطه بین حالت‌های سیستم و ورودی و خروجی سیستم را تعیین می‌کنند.

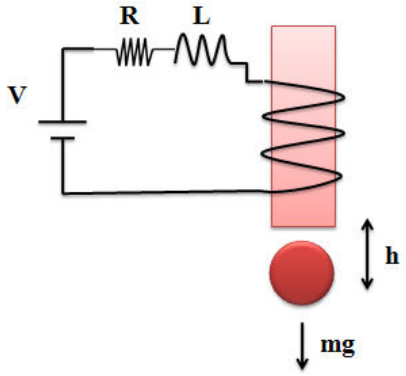
امروزه خوشبختانه بسیاری از محاسبات ریاضی مورد نیاز برای تحلیل و طراحی سیستم های خطی را می‌توان به سادگی در فضای حالت و توسط کامپیوتر انجام داد. این محاسبات در مورد سیستمهای با مرتبه بالا می‌تواند بسیار خسته کننده باشد، در حالی‌که افزایش متغیرهای حالت (مرتبه سیستم یا تعداد قطب‌های تابع تبدیل)، تعداد ورودی‌ها یا تعداد خروجی‌ها هیچکدام پیچیدگی ساختاری معادلات را بیشتر نمی‌کند. مهمترین

[1] State-space

کاربردهای مدل‌های فضای حالت برای سیستم‌های چند ورودی و چند خروجی(MIMO[1]) است، ولی در این قسمت فقط درباره سیستم‌های تک ورودی و تک خروجی(SISO[2]) بحث می‌کنیم.

در طراحی سیستم‌های کنترلی در فضای حالت، دست یابی به موقعیت مطلوب قطب های حلقه بسته، یعنی ریشه‌های معادله مشخصه و یا به عبارت دیگر مقادیر ویژه ماتریس A در سیستم حلقه بسته، این امکان را می‌دهد که به‌سادگی بتوان سیستم‌هایی را طراحی کرد که خواسته‌های طراحی و عملکرد را در حد مناسبی ارضا نمایند.

برای معرفی طراحی به روش فضای حالت یک مثال می‌زنیم. فرض کنید که یک توپ مغناطیسی به صورت معلق قرار گرفته است. جریان عبوری از داخل سیم‌پیچ یک نیروی مغناطیسی به توپ اعمال می‌کند. این نیرو با نیروی جاذبه اعمال شده به توپ در تعادل است. بنابراین توپ در فضای بین، به صورت معلق باقی می‌ماند. معادلات این سیستم به صورت زیر نوشته می‌شود:



شکل (۳-۱) توپ مغناطیسی معلق

$$\sum F = Ma \Rightarrow Mg - \frac{Ki^2}{h} = M\frac{d^2h}{dt^2}$$

$$\sum \Delta V = 0 \;\Rightarrow\; V - L\frac{di}{dt} - iR = 0 \Rightarrow\; V = L\frac{di}{dt} + iR$$

که در آن h ،موقعیت عمودی، i ،جریان عبوری از آهنربای الکتریکی، V، ولتاژ اعمالی، M ،جرم توپ، g ،گرانش، L ،اندوکتانس، R ،مقاومت، و K ،ضریبی است که نیروی مغناطیسی اعمالی به توپ را تعیین می‌کند. برای ساده سازی معادلات، مقادیر زیر را برای این پارامترها در نظر می‌گیریم:

$$M = 0.05\,Kg, K = 0.0001, L = 0.01H, R = 1\Omega, g = 9.81\frac{m}{s^2}$$

[1] Multi –Input, Multi- Output
[2] Single- Input ,Single- Output

سیستم در حالت تعادل است (توپ به صورت معلق است)، پس:

$$M\frac{d^2h}{dt^2}=Mg-\frac{Ki^2}{h}\xrightarrow{\frac{dh}{dx}=0}h=\frac{Ki^2}{Mg}$$

معادلات را حول نقطه h=0.01m که جریان عبوری 7amp است خطی می‌کنیم و معادلات فضای حالت را به دست می‌آوریم. در این سیستم، h ،خروجی،V، ولتاژ ورودی، و $x=\begin{bmatrix}\Delta h & \Delta\dot{h} & \Delta i\end{bmatrix}^T$، متغیرهای حالت سیستم می‌باشند.

$$\begin{cases}\dot{x}(t)=Ax(t)+Bu(t)\\ y(t)=Cx(t)+Du(t)\end{cases}\Rightarrow\begin{cases}\begin{bmatrix}\Delta\dot{h}\\ \Delta\ddot{h}\\ \Delta\dot{i}\end{bmatrix}=A\begin{bmatrix}\Delta h & \Delta\dot{h} & \Delta i\end{bmatrix}^T+BV\\ y=C\begin{bmatrix}\Delta h & \Delta\dot{h} & \Delta i\end{bmatrix}^T\end{cases}$$

که در آن، ماتریس‌های A,B,C,D به صورت زیر می‌باشند:

$$A=\begin{bmatrix}0 & 1 & 0\\ 980 & 0 & -2.8\\ 0 & 0 & -100\end{bmatrix}$$

$$B=[0;0;100]$$

$$C=[1\quad 0\quad 0]$$

$$D=[0]$$

اولین کاری که باید انجام دهیم این است که مقادیر ویژه ماتریس A ،یعنی همان قطب‌های سیستم(با فرض عدم حذف صفرها و قصبهای سیستم در حالت تابع تبدیل) را پیدا کنیم. نمایش‌های فضای حالت و تابع تبدیل به صورت زیر به هم مربوط می‌شوند:

$$\boldsymbol{G(s)=C(sI-A)^{-1}B+D}\qquad(۲-۳)$$

مخرج معادله بالا برابر $|sI-A|=s^n+a_{n-1}s^{n-1}+....+a_1s+a_0$ است، که این معادله را معادله مشخصه سیستم می‌گویند. مقادیر ویژه سیستم یا صفرهای معادله مشخصه از رابطه $|sI-A|=0$ به‌دست می‌آید.

```
poles = eig(A)
>> poles =
   31.3050
  -31.3050
 -100.0000
```

برای بررسی پایداری سیستم در فضای حالت می‌توان مقادیر ویژه A را تعیین کرد. نتایج پایداری را می‌توان به صورت زیر خلاصه نمود:

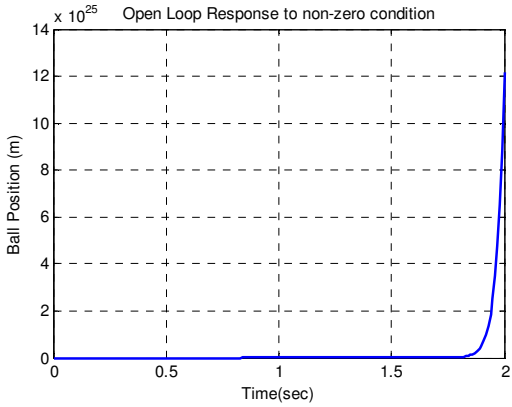
- اگر مقادیر ویژه حقیقی یا قسمت حقیقی مقادیر ویژه مختلط مزدوج A همگی منفی باشند، سیستم پایدار است.
- اگر مقادیر ویژه A موهومی محض باشند، سیستم پایدار بحرانی است؛ یعنی پاسخ، نوسانات سینوسی غیر میرا دارد.
- اگر چند قطب A موهومی محض و در یک نقطه بر روی محور موهومی قرار داشته باشند، سیستم ناپایدار است.
- اگر یکی از قطب ها در سمت راست محور موهومی باشد، سیستم ناپایدار است.

نکته: در سیستم‌های غیرخطی به طور مفصل روی پایداری سیستم‌هایی که مدل خطی شده آن ها دارای قطب یا قطبهایی بر روی محور $j\omega$ می‌باشند، بحث می‌شود. در واقع در آن جا تئوری Center Manifold مطرح می‌گردد و پایداری سیستم غیرخطی مرتبه بالا را با پایداری یک سیستم غیرخطی مرتبه پایین‌تر بررسی می کند.

در مثال فوق، یکی از قطب ها در سمت راست محور موهومی قرار گرفته است و با توجه به نکات بالا، این امر نشان می‌دهد که سیستم حلقه باز ناپایدار است. می‌خواهیم تبعات این موضوع را برای وقتی که شرایط اولیه غیر صفر داشته باشیم مورد بررسی قرار دهیم. بنابراین دستورات زیر را به ادامه m-file قبلی اضافه می‌کنیم:

```
t = 0:0.01:2;
u = 0*t;
x0 = [0.005 0 0];
[y,x] = lsim(A,B,C,0,u,t,x0);
h = x(:,2);
plot(t,h)
```

بعد از اجرای برنامه داریم:



شکل (۳-۲) پاسخ سیستم حلقه باز به شرایط غیرصفر

همانطور که می‌بینید فاصله بین توپ و آهن ربا به سمت بی‌نهایت میل می‌کند و نشان می‌دهد که سیستم حلقه باز ناپایدار است. اگر بخواهیم از فیدبک متغیرهای حالت برای پایدار سازی سیستم استفاده کنیم، باید کنترل پذیری و رویت پذیری سیستم را بررسی کنیم.

۳-۱-۱ بررسی کنترل پذیری و رویت پذیری سیستم

یکی از مفاهیمی که مختص تحلیل و طراحی در فضای حالت است کنترل پذیری و رویت پذیری می‌باشد. علاوه بر این ایده ها باید مفاهیم پایداری و مشخصه‌های حالت گذرا و ... را که در کنترل کلاسیک نیز مطرح بود در اینجا بررسی کنیم، که البته نحوه برخورد با این مفاهیم متفاوت است.

طبق تعریف، سیستمی را **کنترل‌پذیر**[1] گویند،که اگر سیگنال کنترلی u(t) در بازه زمانی دلخواه و محدود $t \in [t_0, t_1]$ وجود داشته باشد، سیستم مورد بحث تحت این ورودی، از حالت اولیه $x(t_0)$ به هر حالت مطلوب $x(t_1)$ سیر نماید. اگر شرط کنترل پذیری وجود نداشته باشد، کنترل حلقه باز کامل و به دنبال آن کنترل حلقه بسته کامل برای سیستم تحت کنترل ممکن نخواهد بود.

ماتریس کنترل‌پذیری به‌صورت $C = \begin{bmatrix} B & AB & A^2B & .... & A^{n-1}B \end{bmatrix}$ تعریف می‌شود و شرط کنترل‌پذیری سیستم آن است که مرتبه[2] این ماتریس باید n، یعنی برابر با بعد ماتریس A باشد. مرتبه یک ماتریس بیانگر تعداد سطرها و یا ستون‌های مستقل خطی آن ماتریس می‌باشد.

سیستمی را **رویت پذیر** [3] می‌گویندکه بتوان بردار شرایط اولیه سیستم را با در دست داشتن مقادیر ورودی و خروجی آن در یک بازه زمانی دلخواه و محدود $t \in [t_0, t_1]$ به صورت یکتا به دست آورد. معنای تلویحی این امر آن است که می‌توان با حل مجدد معادلات حالت تحت ورودی مفروض، مقدار x(t) را در بازه زمانی فوق به دست آورد.

ماتریس رویت پذیری نیز به صورت $O = \begin{bmatrix} C & CA & CA^2 & .... & CA^{n-1} \end{bmatrix}^T$ تعریف می‌شود، که در اینجا $[\bullet]^T$ نماد ترانسپوز یا ترانهاده یک ماتریس است. شرط رویت پذیری سیستم آن است که مرتبه این ماتریس باید n، یعنی برابر با بعد ماتریس A باشد. دستورات زیر برای تست کنترل پذیری و رویت پذیری استفاده می‌شوند:

```
co=ctrb (A,B);
ob=obsv (A,C);
Controllability=rank(co)
Observability=rank(ob)
```

Controllability = 3

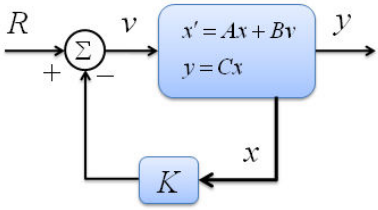
Observability = 3

چون مرتبه ماتریس‌های کنترل پذیری و رویت پذیری برابر با بعد ماتریش A،یعنی 3 است، بنابراین سیستم کنترل پذیر و رویت پذیر می‌باشد.

۳-۱-۲ طراحی کنترل‌کننده حالت با استفاده از روش قطب گذاری

[1] State Controllable
[2] Rank
[3] Observable

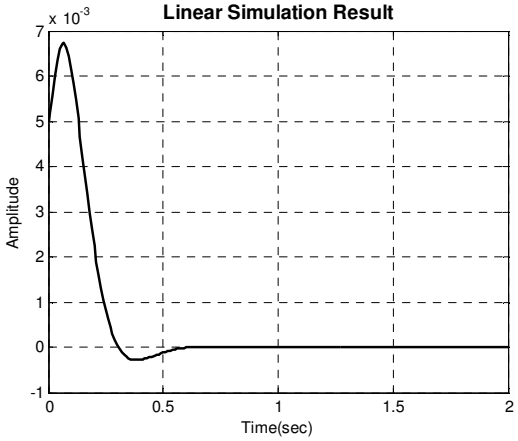
در اینجا می‌خواهیم برای سیستم مثال قبل، یک کنترل کننده حالت طراحی کنیم. ساختار زیر، یک سیستم کنترل حلقه بسته با بازخورد مستقیم تمامی متغیرهای حالت سیستم را نشان می‌دهد:



شکل (۳-۳) سیستم کنترل حلقه بسته با بازخورد مستقیم

پارمتر k را بهره کنترلی نامیده و معادله مشخصه سیستم حلقه بسته از رابطه $|sI-(A-BK)|$ به‌دست می‌آید. ماتریس‌های A و BK هر دو ماتریس‌های 3×3 هستند. پس این سیستم دارای 3 مقدار ویژه است. در صورت کنترل پذیر بودن یک سیستم، با استفاده از فیدبک تمامی متغیرهای حالت، می‌توانیم قطب های حلقه بسته را در هر نقطه‌ای از صفحه s قرار دهیم. برای پیدا کردن مقدار K می‌توانیم از دستور **Place** استفاده کنیم. باید تصمیم بگیریم که قطب‌های حلقه بسته را در کجا قرار دهیم. برای سیستم بالا می‌خواهیم زمان نشست کمتر از 0.5 ثانیه و جهش کمتر از 5% باشد. پس باید سعی کنیم قطب ها را در $-10\pm10i$ ($\zeta=0.7$) و قطب سوم را در مکانی نسبتاً دورتر مثلاً در 50- قرار دهیم.

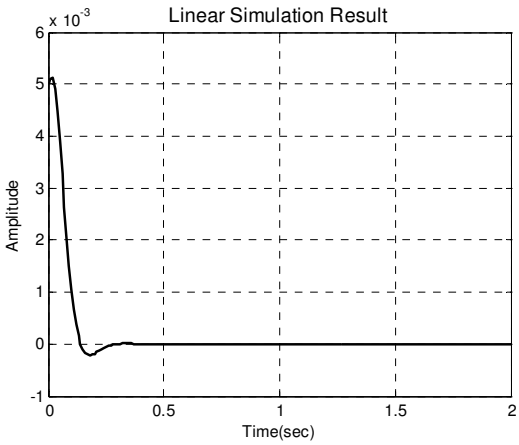
```
p1 = -10 + 10i;p2 = -10 - 10i;p3 = -50;
K = place(A,B,[p1 p2 p3]);
lsim(A-B*K,B,C,0,u,t,x0);
```



شکل (۳-۴) پاسخ شبیه سازی سیستم خطی

مقدار جهش تا حدودی زیاد است (یکسری صفر در تابع تبدیل وجود دارد که باعث افزایش جهش می‌شود). قطب ها را بیشتر به سمت راست بکشید تا پاسخ بهبود پیدا کند.

```
p1 = -20 + 20i;p2 = -20 - 20i;p3 = -100;
K = place(A,B,[p1 p2 p3]);
lsim(A-B*K,B,C,0,u,t,x0);
```



شکل (۳-۵) پاسخ بهینه شده سیستم خطی

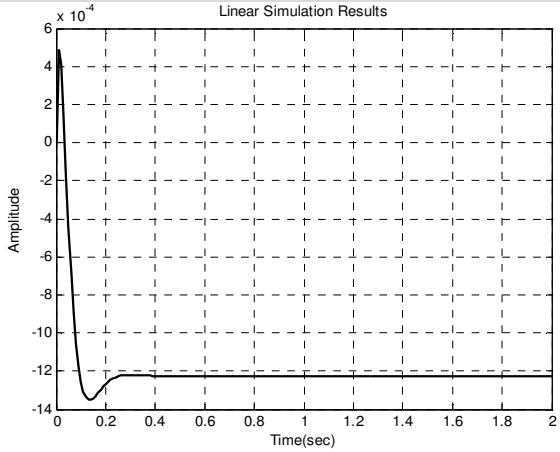
نکته: اگر بخواهید دو یا چند قطب را در یک محل یکسان قرار دهید، نمی‌توانید از دستور place استفاده کنید و باید از دستور acker که فرمت آن به صورت زیر است، استفاده کنید.

```
K = acker(A,B,[p1 p2 p3])
```

۳-۱-۳ اعمال ورودی مرجع[1]

به سیستم کنترلی بالا یک ورودی پله اعمال می‌کنیم (یک مقدار کوچک برای ورودی پله انتخاب می‌کنیم تا در محدوده‌ای که خطی سازی معتبر است باقی بماند). دستورات زیر را به M-file اضافه کنید:

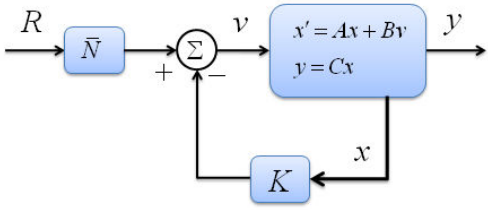
```
t = 0:0.01:2;
u = 0.001*ones(size(t));
lsim(A-B*K,B,C,0,u,t)
```



شکل (۳-۶) پاسخ سیستم خطی به ورودی پله

[1] Reference input

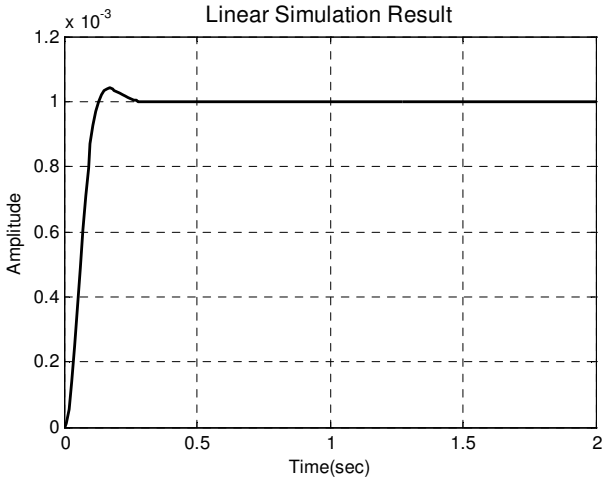
در ساختار کنترلی فوق، خروجی سیستم با ورودی مرجع مقایسه نشد. در عوض تمام حالت ها اندازه‌گیری گردید و در بردار K ضرب شد و نتیجه حاصل را با مقدار مرجع مقایسه کردیم. هیچ دلیلی ندارد که انتظار داشته باشیم که در حالت ماندگار، مقدار خروجی، y، برابر مقدار مطلوب، R، باشد. برای جبران این مساله باید ورودی را به طور مناسبی مقیاس بندی کنیم تا خروجی حاصل برابر ورودی مرجع شود. این فاکتور مقیاس‌بندی $\bar{N}$ نامیده می‌شود که در ساختار زیر نمایش داده شده است. نحوه محاسبه $\bar{N}$ نیز در m-file زیر آمده است.



شکل (۳-۷) سیستم کنترل حلقه بسته با بازخورد مستقیم همراه با ورودی مرجع

```
Nbar=rscale(A,B,C,0,K)
lsim(A-B*K,B*Nbar,C,0,u,t)
```

```
Nbar =
 -285.7143
```



شکل (۳-۸) پاسخ سیستم خطی با ورودی مرجع

توجه : برای استفاده از دستور rscale اول باید برنامه زیر را در یک m-file کپی کنید و آن را در پوشه Work ذخیره کنید.

```
function[Nbar]=rscale(A,B,C,D,K)
s = size(A,1);
Z = [zeros([1,s]) 1];
N = inv([A,B;C,D])*Z';
Nx = N(1:s);
```

```
Nu = N(1+s);
Nbar=Nu + K*Nx;
```

۳-۱-۴ طراحی رویتگر[1]

اغلب اوقات در سیستم‌های عملی، به دلیل مسائل فنی و یا اقتصادی امکان اندازه‌گیری و بازخورد تمامی متغیر های حالت وجود ندارد. همچنین در بسیاری از روشهای مدل سازی مبتنی بر تخمین و شناسائی سیستم‌ها[2]، بردار حالت حاصل ممکن است فاقد معنای فیزیکی مستقیم و روشنی باشد. از اینرو قاعدتاً ممکن است وسیله اندازه‌گیری استانداردی برای اندازه‌گیری این متغیرها موجود نباشد و یا آن که بسیار گران قیمت باشند. گاهی نیز پیش می‌آید که متغیرهای حالت اگرچه معنای فیزیکی دارند، اما سنسور مناسبی برای اندازه گیری آنها وجود ندارد (مانند فاصله خودرو از زمین در سیستمهای تعلیق فعال و یا لایه سوخت موجود در جداره منیفولد مکش در موتورهای احتراق داخلی). گاهی نیز به دلیل وجود نویز شدید در محیط، اندازه‌گیری دقیق بعضی از متغیرهای حالت عملاً با مشکل جدی روبه‌رو است. در تمامی این موارد، می‌توان از یک رویتگر حالات برای تخمین همزمان و برخط متغیرهای حالتی که امکان اندازه گیری آنها وجود ندارد، استفاده نمود.

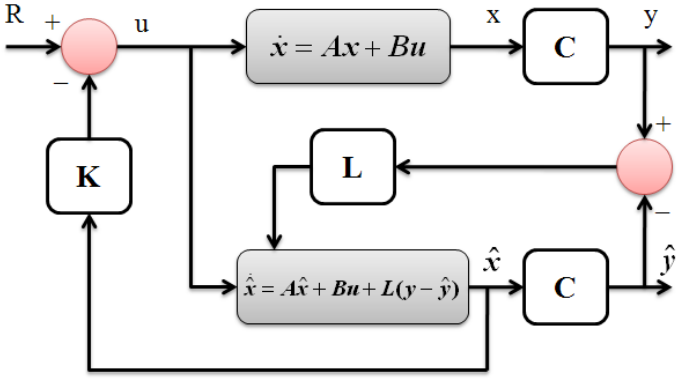
در این قسمت می‌خواهیم با استفاده از یک رویتگر برای توپ مغناطیسی، تمام متغیرهای حالت را تخمین بزنیم. توجه کنید رویتگر یک سیستم دینامیکی است که بردار حالت را تخمین می‌زند، یعنی بعد از یک حالت گذرای سریع مقادیر بردار حالت تخمین زده شده، و با بردار حالت واقعی یکی خواهند شد. با به‌کارگیری رویتگر در یک سیستم، دینامیک رویتگر به صورت جزئی از دینامیک سیستم حلقه بسته ظاهر شده و اثر مستقیمی بر عملکرد سیستم حلقه بسته خواهد گذارد. بهترین روش برای تخمین بردار حالت، استفاده از یک پردازنده رایانه‌ای است که به صورت همزمان با سیستم واقعی، تحت ورودی مشترکی واقع شده و رفتار سیستم واقعی را به نحوی پایدار شبیه سازی نموده و مقادیر حالات سیستم را به صورت همزمان با واقعیت در اختیار سیستم کنترل می‌گذارد.

در عمل سیستم واقعی تحت انواع اغتشاشات و نویزهایی قرار خواهد گرفت که اندازه‌گیری آنها غیرممکن است واز اینرو بین خروجی سیستم واقعی و خروجی سیستم شبیه سازی شده تفاوتی ایجاد خواهد شد. به علاوه، اگر سیستم اصلی پایدار نباشد، باید تمهیداتی را در رویتگر در نظر گرفت که عملاً امکان تخمین مجانبی حالات سیستم فراهم آید. معادلات حالت و خروجی سیستم دینامیکی یک رویتگر عبارت است از:

$$\begin{aligned}\dot{\hat{\boldsymbol{x}}}(t) &= \boldsymbol{A}\hat{\boldsymbol{x}}(t) + \boldsymbol{B}\boldsymbol{u}(t) + \boldsymbol{L}(\boldsymbol{y}(t) - \hat{\boldsymbol{y}}(t)) \\ \hat{\boldsymbol{y}}(t) &= \boldsymbol{C}\hat{\boldsymbol{x}}(t)\end{aligned} \qquad (۳-۳)$$

که ساختار کلی آن در یک سیستم حلقه بسته با فیدبک حالات به صورت زیر است:

[1] Observer Design
[2] System identification methods



شکل (۳-۹) سیستم حلقه بسته همراه با رویتگر

رویتگر، اساساً مشابه سیستم واقعی است که هر دوی آنها تحت یک ورودی قرار دارند و تقریباً معادلات دیفرانسیل یکسانی دارند. ترم اضافی مقایسه‌ای است بین مقادیر خروجی اندازه گیری شده واقعی y و مقادیر خروجی تخمینی $\hat{y}$ ،که این باعث تخمین متغیرهای حالت $\hat{x}$ می‌شود تا به مقادیر واقعی x نزدیک شود. سپس از بردار حالت تخمین زده شده $\hat{x}$ به جای حالت واقعی x در سیستم کنترل استفاده می‌کنیم. خطای رویت برابر است با :

$$e(t) = x(t) - \hat{x}(t) \qquad (۳-۴)$$

و معادله دینامیک خطا به صورت زیر به‌دست می‌آید:

$$\dot{e}(t) = (A - LC)e(t) \qquad (۳-۵)$$

معادله بالا نشان می‌دهد که معادله خطا ورودی نداشته و تنها با شرایط اولیه تحریک می‌شود. بنابراین خطای رویتگر با ورودی سیستم تعیین نمی‌شود. توجه کنید که خطای رویت بعد از یک حالت گذرا صفر خواهد شد. پس دینامیک‌های خطای رویتگر به‌وسیله قطب‌های A-LC شکل داده می‌شود که با انتخاب مناسب ماتریس L می‌توان مقادیر ویژه A-LC را در سمت چپ محور موهومی و در محل‌های مناسب قرار داد. دینامیک تلفیقی سیستم حلقه بسته با کنترل فیدبک حالات رویت شده به صورت زیر می‌باشد:

$$\begin{bmatrix} \dot{x}(t) \\ \dot{e}(t) \end{bmatrix} = \begin{bmatrix} A - BK & BK \\ 0 & A - LC \end{bmatrix} \begin{bmatrix} x(t) \\ e(t) \end{bmatrix} \qquad (۳-۶)$$

بنابراین معادله مشخصه سیستم حلقه بسته عبارت است از:

$$|sI - A + BK||sI - A + LC| = 0 \qquad (۳-۷)$$

این رابطه نشان می‌دهد که قطب‌های حلقه بسته کنترل فیدبک حالت رویت شده، در واقع مجموعه‌ای از قطب های ناشی از طراحی با روش قطب گذاری و قطب‌های ناشی از رویتگر است. این بدان معنی است که مراحل طراحی رویتگر و روش قطب گذاری را می‌توان به صورت جداگانه انجام داد. بنابراین دو فرایند طراحی به طور کاملاً مستقل از یکدیگر انجام می‌شوند.

اصل جدایش[1]

بر اساس اصل جدایش می‌توان گفت که طراحی کنترلر و طراحی رویتگر می‌تواند جدا از هم انجام شود. به این صورت که ابتدا کنترلر را با شرایط مورد نیاز طراحی کرده، سپس رویتگر را طراحی می‌کنیم و سیستم با حضور رویتگر و کنترلر نتایج مطلوبی به ما می‌دهد. نکته قابل توجه این است که اصل بالا مربوط به سیستم‌های خطی است و برای سیستم‌های غیرخطی ممکن است سیستم در حضور کنترلر پایدار باشد ولی با اضافه شدن رویتگر سیستم ناپایدار گردد و باید پایداری را به روش لیاپانوف برای سیستم در حضور کنترلر و رویتگر بررسی کرد.

قطب‌های مطلوب حلقه بسته طوری طراحی می‌شوندکه مشخصه‌های سیستم حلقه بسته برآورده شوند. قطب‌های رویتگر نیز طوری طراحی می‌شوند تا پاسخ رویتگر بسیار سریعتر از پاسخ سیستم باشد. دراین مساله نیاز داریم بهره L رویتگر را تعیین کنیم. بنابراین قطب‌های رویتگر را حداقل ۵ بار دورتر از قطب‌های غالب سیستم در نظر می‌گیریم تا مقادیر ویژه A-LC در سمت چپ قطب‌های حلقه بسته قرار بگیرند. توجه کنید که بالاترین سرعت رویتگر با پارامترهایی مانند نویز محدود می‌شود. اگر ما بخواهیم از دستور place استفاده کنیم باید قطب‌های رویتگر را در مکان‌های مختلف قرار دهیم.

```
op1 = -100;
op2 = -101;
op3 = -102;
```

به‌دلیل دوگانگی (Duality) بین کنترل پذیری و رویت پذیری، می‌توانیم از یک تکنیک مشابه برای پیدا کردن ماتریس کنترل K استفاده کنیم. اما باید به جای ترانهاده ماتریس B، ترانهاده ماتریس C را قرار می‌دهیم.

```
L = place(A',C',[op1 op2 op3])';
```

$$A_t = \begin{bmatrix} A - B \times K & B \times K \\ zeros(size(A)) & A - L \times C \end{bmatrix}$$

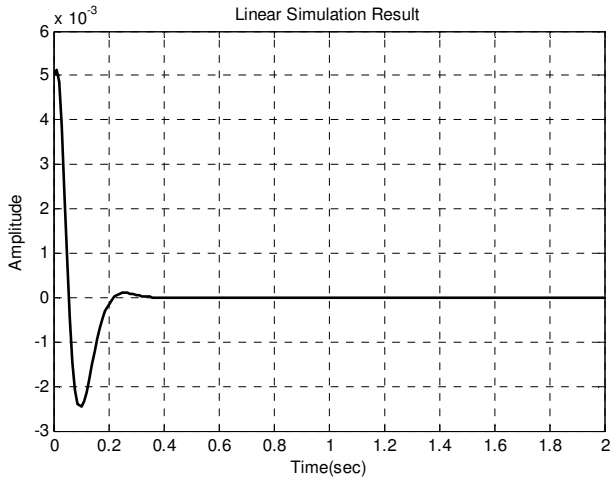
$$B_t = \begin{bmatrix} B \times Nbar \\ zeros(size(B)) \end{bmatrix}$$

$$C = \begin{bmatrix} C & zeros(size(C)) \end{bmatrix}$$

برای مشاهدهٔ اینکه ببینید پاسخ با شرایط اولیه غیر صفر و بدون ورودی رفرنس چگونه است دستورات زیر در یک m-file وارد می‌کنیم:

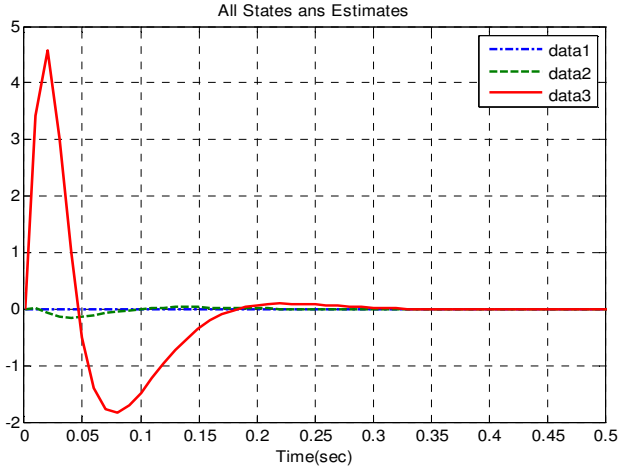
```
sys=ss(At,Bt,Ct,0);
lsim(sys,zeros(size(t)),t,[x0 x0])
[y,x] = lsim(At,Bt,Ct,0,zeros(size(t)),t,[x0 x0]);
plot(t,x(:,1),t,x(:,2),t,x(:,3))
axis([0 0.5 -2 5])
```

[1] Sepration principle



شکل (۳-۱۰) پاسخ با شرایط اولیه غیر صفر و بدون ورودی رفرنس

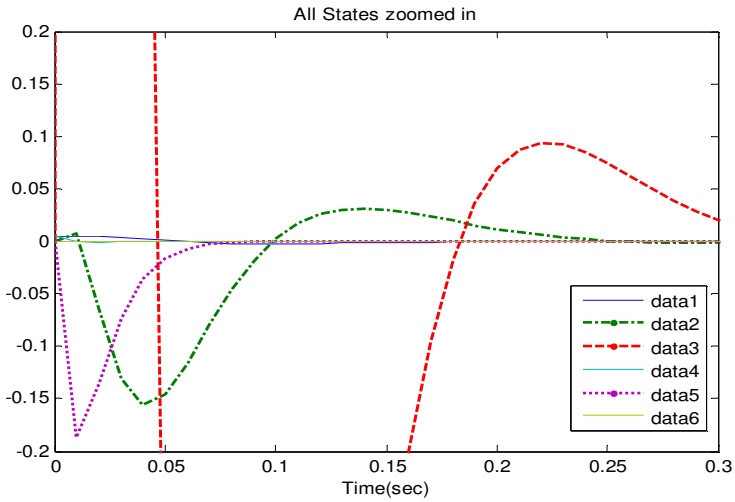
به طور کلی، فرض می‌کنیم که رویتگر با شرایط اولیه صفر $\hat{x} = 0$ شروع می‌کند که شرایط اولیه برای خطا برابر شرایط اولیه متغیرهای حالت می‌گردد. پاسخ برای تمام متغیرهای حالت به صورت زیر است:



شکل (۳-۱۱) پاسخ برای تمام متغیرهای حالت

یادآوری می‌کنیم که دستور lsim به ما مقادیر x,e را می‌دهد و نه $\hat{x}$ را. برای به‌دست آوردن $\hat{x}$ باید $\hat{x} = x - e$ را حساب کنیم. به رویتگری که تمام متغیرهای حالت را تخمین می‌زند رویتگر مرتبه کامل[1] می‌گویند.

[1] Full-Order Observer



شکل (۳-۱۲) پاسخ برای تمام متغیرهای حالت با وضوح دید بیشتر

۳-۱-۵ طراحی سیستم‌های کنترل بهینه خطی[1]

انتخاب مناسب و بهینه قطب‌های مطلوب برای سیستم‌های صنعتی و فرایندهای واقعی تا حدودی پیچیده است. تعیین مکان قطب‌های حلقه بسته در توصیف رفتار مطلوب مورد نظر طراح دشوار است. انتخاب موقعیت قطب‌های حلقه بسته دور از مبداء سریعتر شدن پاسخ دینامیکی سیستم را به همراه دارد. اما با سریعتر کردن پاسخ، سیگنال‌های کنترلی بزرگ شده و عموماً محرک‌های سیستم قادر به اجرای فرامین کنترلی نخواهند بود. اگر قطب‌های حلقه بسته به گونه‌ای جایابی شوند که باعث تجاوز سیگنال‌های کنترلی از حد فیزیکی شوند، یا به عبارت دیگر اشباع گردند، در آن صورت رفتار دینامیکی حلقه بسته مشابه رفتار پیش بینی شده با تحلیل خطی نبوده و حتی ممکن است که رفتار حلقه بسته سیستم واقعی ناپایدار گردد.

دلیل دیگری که برای محدود کردن سرعت پاسخ وجود دارد، مسأله نویز است که معمولاً با سیستم‌های بهره بالا همراه است. پس با انتخاب بهینه قطب‌های حلقه بسته می‌توان به سرعت دلخواه حلقه بسته و اندازه قابل قبول سیگنال کنترلی دست یافت. سیستم داده شده با معادلات حالت زیر را در نظر بگیرید:

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{u}(t) \qquad (۳-۸)$$

در حل مسأله کنترل بهینه، می‌خواهیم ماتریس بهره فیدبک حالت K را در کنترل فیدبک حالت چنان پیدا کنیم که شاخص عملکرد داده شده زیر را حداقل‌سازی کند، به گونه‌ای که مصالحه‌ای بین انرژی سیگنال کنترلی و انرژی خطا ایجاد شود.

$$\boldsymbol{J} = \int_{t_0}^{t_1} [\boldsymbol{x}^T(t).\boldsymbol{Q}(t).\boldsymbol{x}(t) + \boldsymbol{u}^T(t).\boldsymbol{R}.\boldsymbol{u}(t)]dt \qquad (۳-۹)$$

[1] Linear Optimal Control

که در آن، Q ماتریس وزنی متقارن معین (یا نیمه معین) ، حقیقی و مثبت، و R ماتریس متقارن معین مثبت است.

ماتریس K به صورت U=-KX به‌دست می‌آید که K برابراست با $K = R^{-1}B^T P$ و P، یک ماتریس حقیقی متقارن مثبت معین است که معادله جبری ریکاتی زیر را برآورده می‌کند:

$$A^T P + PA - PBR^{-1}B^T P + Q = 0 \qquad (۱۰-۳)$$

ماتریس K را می‌توان با دستور زیر در MATLAB محاسبه کرد:

[K]= lqr (A,B,Q,1)

۳-۱-۶ تبدیل بین نمایش سیستم ها

۳-۱-۶-۱ تبدیل فضای حالت به تابع تبدیل

فرض کنید معادلات فضای حالت سیستم را در اختیار داریم. برای تبدیل این نمایش به مدل تابع تبدیل معادل از دستور زیر استفاده می‌کنیم:

[num,den] = ss2tf(A,B,C,D,iu)

این دستور تابع تبدیل برای ورودی iu ام را حساب می‌کند. بیشتر سیستم‌های که بررسی می‌کنیم سیستم‌های دارای یک ورودی هستند. بنابراین iu=1 و نیازی نیست که در دستور نوشته شود. فرض کنید معادلات فضای حالت سیستم به صورت زیر می‌باشد:

$$\begin{bmatrix} \dot{x} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & -b/m \end{bmatrix} \begin{bmatrix} x \\ v \end{bmatrix} + \begin{bmatrix} 0 \\ 1/m \end{bmatrix} u$$

$$y = [0 \;\; 1] \begin{bmatrix} x \\ v \end{bmatrix} + 0[u]$$

و فرض کنید $m = 1000, b = 50, u = 500$

پس در یک M-file دستورات زیر را وارد کنید:

$$A = \begin{bmatrix} 0 & 1 \\ 0 & -0.05 \end{bmatrix}$$

$$B = \begin{bmatrix} 0 \\ 0.001 \end{bmatrix}$$

$$C = \begin{bmatrix} 0 & 1 \end{bmatrix}$$

$$D = \begin{bmatrix} 0 \end{bmatrix}$$

```
[num,den]=ss2tf(A,B,C,D)
```

MATLAB خروجی زیر را می‌دهد:

```
          0.001s + 0
------------------
s^2 + 0.05s + 0
```

همیشه تابع تبدیل را چک کنید. چون ممکن صفرهای بی‌نهایت در سیستم وجود داشته باشد و یک تابع تبدیل اشتباه را به ما بدهد.

نکته: تعریف صفرهای بینهایت

در صورتی می‌گوئیم که سیستم صفر در بینهایت دارد که اگر $s \to \infty$ ،مقدار تابع تبدیل برابر صفر شود و این زمانی اتفاق می‌افتد که تعداد قطب ها بیشتر از تعداد صفرها باشد. در مکان هندسی ریشه ها به صورت مجانبی به سمت بینهایت می‌رود (تعداد مجانب ها برابر تعداد صفرهای در بی نهایت است). نرم افزار MATLAB گاهی این صفرهای در بینهایت را محاسبه می‌کند و به صورت یکسری اعداد بزرگ نمایش می‌دهد. بهترین مثال معادلات حالت زیر است:

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{\phi} \\ \ddot{\phi} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -0.18 & 2.67 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & -0.45 & 31.18 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \phi \\ \dot{\phi} \end{bmatrix} + \begin{bmatrix} 0 \\ 1.81 \\ 0 \\ 4.54 \end{bmatrix} u$$

$$y = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \phi \\ \dot{\phi} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} u$$

دستورات زیر را در یک M-file وارد کنید:

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -0.1818 & 2.6727 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & -4.545 & 31.1818 & 0 \end{bmatrix}$$

$$B = [0;\ 1.8182;\ 0;\ 4.5455]$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$D = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

```
[num,den]=ss2tf(A,B,C,D)
```

که تابع تبدیل سیستم به صورت زیر به‌دست می‌آید:

```
num =

         0    0.0000    1.8182   -0.0000  -44.5460
         0    0.0000    4.5455    0.0000         0

den =
    1.0000    0.1818  -31.1818   -4.4541         0
```

اگر به صورت کسر نگاه کنید می‌بینید که عضو اول برابر صفر است و عضو دوم و چهارم در هر سطر نیز برابر 0.0000 است. اگر خیلی دقیق تر به این اعداد نگاه کنیم در می‌یابیم که این اعداد برابر صفربوده، و در حقیقت اعداد بسیار کوچکی هستند. برای دیدن این مقادیر دستورات num(1,2), num(1,4), num(2,2) or num(2,4) را وارد کنید و در جواب خواهید دید که به ترتیب مقادیر زیر به‌دست می‌آیند:

7.1e-15 ,-6.2e-15 ,1.23e-14 ,4.44e-15

و با دستور roots((num(1,:)) می‌بینید که صورت کسر ریشه‌هایی در بینهایت دارد.

تمام اعداد کوچک با صفر جایگزین شده اند، همیشه مراقب باشید و به تابع تبدیل توجه کنید و قبل از شروع پروسه طراحی متوجه شوید که این صفرها به چه معناست.

۳-۱-۶-۲ تبدیل تابع تبدیل به فضای حالت

معکوس دستور ss2tf دستور tf2ss است که تابع تبدیل سیستم را به فرم فضای حالت سیستم تبدیل می‌کند .

```
[A,B,C,D] = tf2ss(num,den)
```

نمایش فضای حالت و تابع تبدیلی به صورت زیر به هم مرتبط می‌شوند: $G(s) = C(sI - A)^{-1}B + D$

که در آن $\{A, B, C, D\}$را یک تحقق[1] از $G(s)$می‌نامند. تحققهای یک تابع تبدیل منحصر به فرد نیستند. ولی توجه کنید که معادله مشخصه سیستم برای نمایش‌های مختلف فضای حالت آن یکسان است.

نکته: اگر در تشکیل تابع تبدیل صفر و قطب مشترکی پیدا شوند و برای تعیین $G(s)$ با درجه حداقل، مجبور به حذف صفر وقطب شویم، آن تحقق کنترل ناپذیر یا رویت ناپذیر (یا هر دو) خواهد بود.

یک نکته بسیار مهم که باید به آن اشاره کرد این است که برای یک سیستم فقط یک تابع تبدیل منحصر به فرد وجود دارد ولی می‌توان معادلات فضای حالت متعددی نوشت. دستور tf2ss مدل فضای حالت به فرم کانونیکال را می‌دهد. پس اگر مدل فضای حالت را به فرم تابع تبدیل مبدل کنیم و سپس دوباره تابع تبدیل را به مدل فضای حالت تبدیل کنیم، ممکن است این مدل فضای حالت با مدل فضای حالت قبلی یکسان نباشد. مگر اینکه مدل فضای حالت اولی به فرم کانونیکال باشد. به عنوان مثال:

```
[A,B,C,D] = tf2ss(num,den)
```

همان‌طوری که می‌بینید ماتریس‌های حاصل با ماتریس‌های اولیه متفاوت هستند.

۳-۱-۶-۳ تبدیل فضای حالت به صفر/قطب و تبدیل تابع تبدیل به صفر و قطب

روش سوم برای نمایش سیستم ها، مدل قطب- صفر است. این مدل نیز مشابه روش مدل تابع تبدیل است، با این تفاوت که چند جمله‌ای مخرج به قطب ها و چند جمله‌ای صورت به صفر ها تجزیه شده است. فرمت اصلی آن به صورت زیر است:

$$G(s) = K\frac{(s-z_1)(s-z_2)...(s-z_m)}{(s-p_1)(s-p_2)...(s-p_n)} \qquad (۱۱-۳)$$

به یاد داریدکه برای یک تابع تبدیل سره و مناسب تعداد قطب‌های n باید بزرگتر یا مساوی صفرهای m باشد.

برای انجام تبدیل فضای حالت و تابع تبدیل به مدل صفر- قطب از دستورات زیر استفاده می‌کنیم:

[1] Realization

[z,p,k] = tf2zp(num,den)

و

[z,p,k] = ss2zp(A,B,C,D,iu)

هر دو دستور بالا سه متغیر را به ما می‌دهند. متغیر z همه صفر ها را در ستون‌هایی می‌دهد که هر ستون مربوط به صورت یک تابع تبدیل است. متغیر p تمام قطب ها را در یک ستون نمایش می‌دهد. متغیر K یک ستون از مقادیر بهره ها را ارائه می‌دهد. به طور مثال فرض کنید:

$$num = \begin{bmatrix} 1.8182 & 0 & -44.546 \\ 4.5455 & -7.4373 & 0 \end{bmatrix}$$

$$den = \begin{bmatrix} 1 & 0.1818 & -31.1818 & 6.4786 & 0 \end{bmatrix}$$

```
[z,p,k] = tf2zp(num,den)
```

```
z =
    4.9498    1.6362
   -4.9498         0

p =
         0
    0.2083
   -5.7753
    5.3851

k =
    1.8182
    4.5455
```

در این قسمت دو ستون صفر وجود دارد. بنابراین ماتریس Kدو سطردارد، یعنی هر سطر برای هر ستون ماتریس صفر.دستورات زیر را وارد کنید:

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -0.1818 & 2.6727 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & -4.545 & 31.1818 & 0 \end{bmatrix}$$

$$B = \begin{bmatrix} 0; 1.8182; 0; 4.5455 \end{bmatrix}$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$D = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

```
[z,p,k]=ss2zp(A,B,C,D)
```

که همان جواب قبلی را می‌دهد.

۳-۱-۶-۴ تبدیل صفر/قطب به تابع تبدیل و صفر و قطب به فضای حالت

برای تبدیل مدل قطب – صفر به مدل فضای حالت از دستور زیر استفاده می‌کنیم:

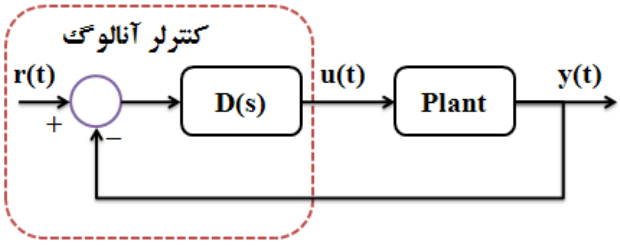
```
[A,B,C,D] = zp2ss(z,p,k)
```

باز یادآور می‌شویم که توجه کنید برای یک سیستم، بیش از یک مجموعه فضای حالت وجود دارد. مدل فضای حالت خروجی این دستور به فرم کانونیکال است. برای تبدیل مدل قطب – صفر به مدل تابع تبدیل از دستور زیر استفاده می‌کنیم:

```
[num,den] = zp2tf(z,p,k)
```

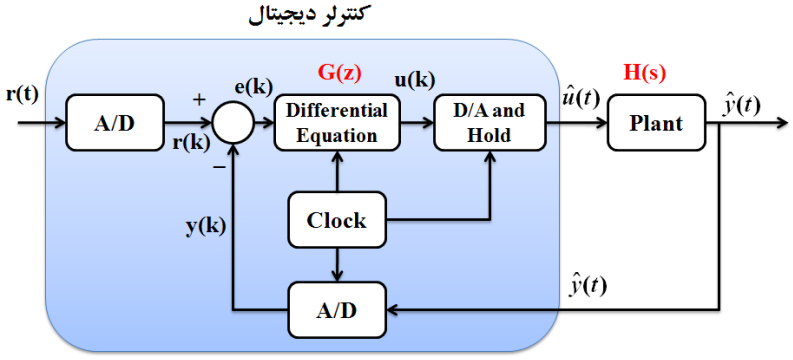
۳ – ۲ روش کنترل دیجیتالی

شکل (۳-۱۳) یک سیستم کنترل فیدبک پیوسته را نمایش می‌دهد. تقریباً تمام کنترلرهای پیوسته می‌توانند با الکترونیک آنالوگ ساخته شوند.



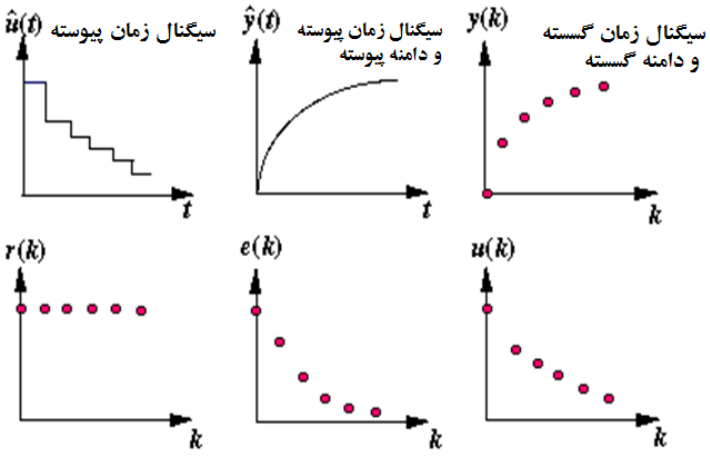
شکل (۳-۱۳) سیستم کنترل فیدبک پیوسته

کنترل کننده پیوسته که در داخل یک مربع خط چین قرار گرفته است می‌تواند با استفاده از یک کنترل کننده دیجیتال جایگزین شود و عملکردی مشابه آن داشته باشد. تفاوت اصلی در این است که عملگرهای سیستم کنترل دیجیتالی با سیگنال‌های گسسته کار می‌کنند.



شکل (۳-۱۴) سیستم کنترل فیدبک دیجیتالی

انواع مختلف سیگنال ها در شکل (۳-۱۵) نشان داده شده است.



شکل (۳-۱۵) انواع مختلف سیگنال‌های موجود در سیستم کنترل دیجیتالی

۳-۲-۱ روش‌های طراحی سیستم‌های کنترل دیجیتال

روش های طراحی سیستم‌های کنترل دیجیتال را به سه دسته اصلی می‌توان تقسیم نمود:

الف) روش غیر مستقیم: در این روش، ابتدا یک کنترل پیوسته برای سیستم طراحی شده و سپس با استفاده از تقریبهای عددی، کنترل ناپیوسته معادل به دست می‌آید. بدیهی است که برای حفظ تشابه رفتاری سیستم دیجیتال با معادل پیوسته آن استفاده از فرکانس نمونه برداری بزرگ ضروری است.

ب) روش طراحی مستقیم: در این روش، ابتدا یک مدل ناپیوسته از سیستم تحت کنترل را به دست آورده و سپس با استفاده از روشهای طراحی دیجیتال که در این کتاب به تفصیل خواهد آمد، اقدام به طراحی کنترل کننده می‌نمایند. مزیت این روش آن است که در اغلب موارد می‌توان کنترل کننده مناسبی را برای سیستم محاسبه نمود. اما به دلیل آنکه تمام فرآیند طراحی در زمان ناپیوسته انجام می‌شود، عملاً هیچ تضمینی برای مناسب بودن رفتار سیستم حلقه بسته در زمانهای مابین لحظات نمونه برداری وجود ندارد. جهت حصول اطمینان از صحت عملکرد سیستم در شرایط واقعی، لازم است فاصله نمونه برداری سیگنالها چندین برابر سریعتر از زمان نشست حلقه بسته انتخاب شود. بدیهی است که به علت آنکه زمان نشست سیستم حلقه بسته از قبل معلوم نیست، انتخاب پریود نمونه برداری باید در یک فرآیند رفت و برگشتی انجام گیرد.

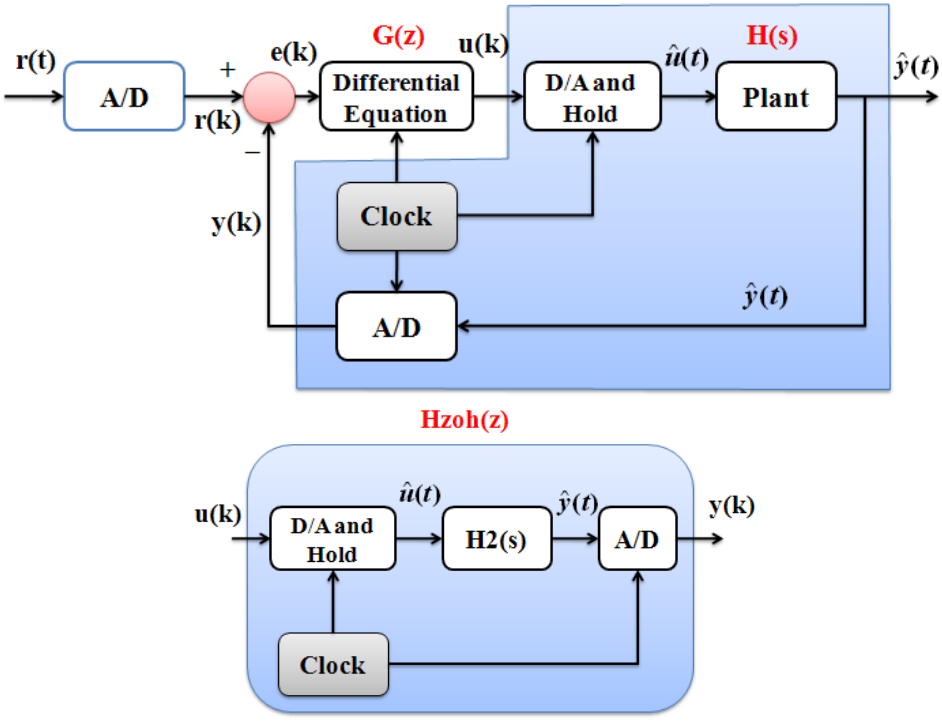
ج) روش هیبرید: دراین روش، رفتار سیستم کنترل دیجیتال در حوزه زمان پیوسته مورد مطالعه و بهینه سازی قرار می‌گیرد. این روش نسبت به دو روش قبلی از پیچیدگی‌های نظری زیادی برخوردار است، ولی مزیت آن در حفظ دقت طراحی حتی در زمانهای بین فواصل نمونه برداری است.

چنانکه گفتیم، در این کتاب عمدتاً روش دوم یعنی طراحی مستقیم در زمان ناپیوسته مورد بحث قرار می‌گیرد، که شروع آن از یافتن معادل ناپیوسته برای سیستم تحت کنترل آغاز می‌گردد. با توجه به آنکه سیستمهای

کنترل صنعتی عمدتاً از نگهدارنده‌های مرتبه صفر (zero-order hold) برای تبدیل خروجی ناپیوسته کنترل کننده به سیگنالهای پله وار برای اعمال به عملگرهای کنترلی استفاده می‌نمایند، در بخش بعدی این موضوع مورد بحث قرار می‌گیرد.

۳-۲-۲ معادل سازی گسسته سیستم تحت کنترل با نگهدارنده مرتبه صفر[1]

در شکل‌های بالا مشاهده کردیم که سیستم کنترلی هم شامل قسمت پیوسته و هم گسسته است. وقتی می‌خواهیم یک سیستم کنترل دیجیتالی را به روش مستقیم طراحی کنیم حتماً نیاز داریم که معادل دیجیتالی قسمت پیوسته را پیدا کنیم؛ زیرا با توابع گسسته روبرو هستیم. بنابراین از شکل بالا، قسمتی را که در شکل زیر مشخص شده است، جدا می‌کنیم.



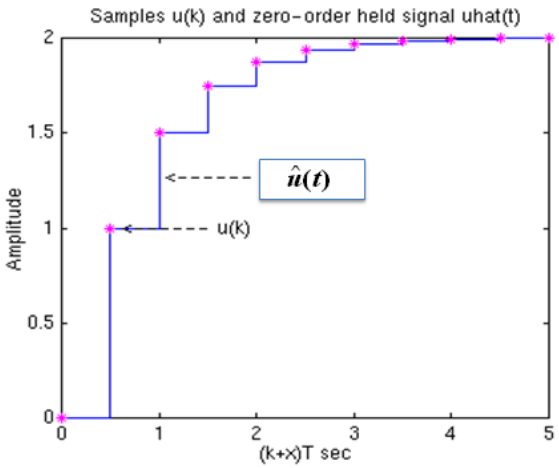
شکل (۳-۱۶) جداسازی نگهدارنده مرتبه صفر (zero-order hold)

Clock به هر دو تبدیل کننده A/D و D/A متصل است و هر T ثانیه یکبار پالس تولید می‌کند. این تبدیل کننده ها هر زمان که پالس دریافت کنند سیگنال می‌فرستند. هدف از وجود این پالس این است که Hzoh(z)

[1] Zero-order hold equivalent discrete model

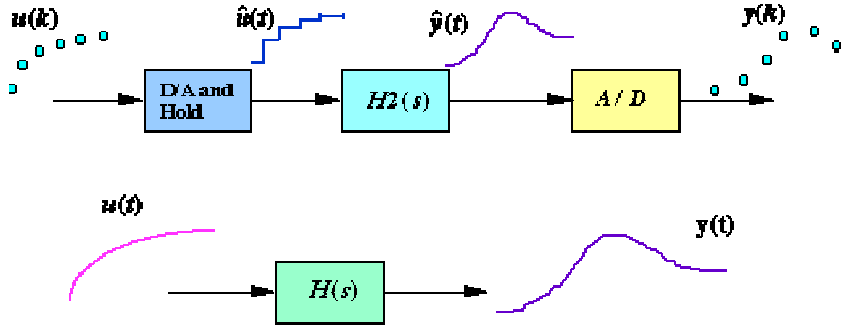
فقط از نمونه‌های ورودی u(k) خروجی y(k) را تولید کند. Hzoh(z) را می‌توان به صورت یک تابع گسسته نوشت.

فلسفه این طراحی این است که ما می‌خواهیم تابع گسسته‌ای را پیدا کنیم که ورودی‌های تکه‌ای که وارد سیستم پیوسته H(s) می‌شوند بعد از خروج از این سیستم پیوسته دوباره به حالت گسسته تبدیل شوند. فرض کنید که سیگنال u(k) نمایش دهنده نمونه برداری‌های انجام شده از سیگنال پیوسته ورودی باشد. بعد از نگه داشتن این سیگنال‌ها یک سیگنال پیوسته $\hat{u}(t)$ تولید می‌کند. شکل (۳-۱۷) نشان می‌دهد که سیگنال $\hat{u}(t)$ بعد از نگه داشتن سیگنال u(k) در بازه زمانی kT تا (k+1)T تولید می‌شود. این عمل نگه داشتن سیگنال نمونه برداری شده در یک بازه زمانی T ثانیه‌ای، نگهدارنده مرتبه صفر نامیده می‌شود.

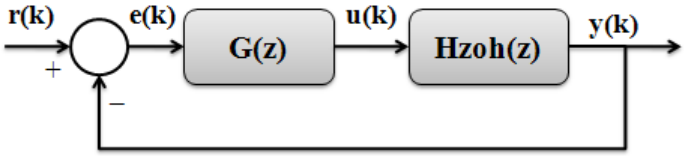


شکل (۳-۱۷) نمونه برداری از u(k) و سیگنال خروجی zoh

سیگنال $\hat{u}(t)$ از H2(s) عبور می‌کند و یک سیگنال پیوسته y(t) را تولید می‌کند، دوباره از این سیگنال خروجی نمونه برداری می‌شود و سیگنال y(k) را که به صورت تکه‌ای و ناپیوسته است، تولید می‌شود.



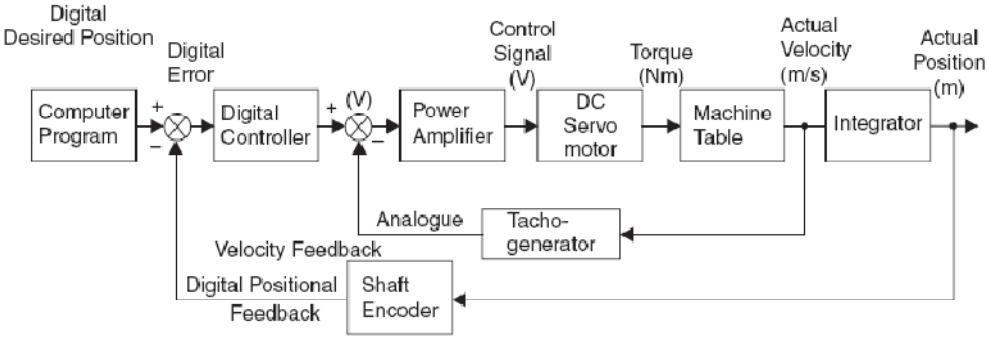
حال ما می‌توانیم شکل بالا را دوباره بکشیم و Hzoh(z) را جایگزین قسمت پیوسته نماییم.



شکل (۳-۱۸) سیستم کنترل فیدبک دیجیتالی خلاصه شده

با جایگزین کردن Hzoh(z) می‌توانیم سیستم کنترل دیجیتالی را با استفاده از توابع گسسته طراحی کنیم.

شکل زیر بلوک نمودار سیستم کنترلی ماشین ابزار CNC را نشان می‌دهد که یک کنترل کننده دیجیتالی برای آن در نظر گرفته شده است.



شکل (۳-۱۹) سیستم کنترلی ماشین ابزار CNC

۳-۲-۳ تبدیل با استفاده از دستور c2dm

دستور بسیار مفیدی به نام c2dm در MATLAB وجود دارد که سیستم‌های پیوسته داده شده به فرم تابع تبدیلی یا فضای حالت را با استفاده از zoh به فرم گسسته تبدیل می‌کند. فرم کلی این دستورات به صورت زیر می‌باشد:

```
[numDz,denDz] = c2dm (num,den,Ts,'zoh')
[F,G,H,J] = c2dm (A,B,C,D,Ts,'zoh')
```

زمان نمونه برداری T_s (sec) باید کوچکتر از $1/(30\times BW)$ باشد که BW فرکانس پهنای باند سیستم حلقه بسته می‌باشد.

۴-۲-۳ تابع تبدیل

تابع تبدیل یک سیستم پیوسته را به صورت زیر در نظر بگیرید:

$$\frac{X(s)}{F(s)}=\frac{1}{Ms^2+bs+k}$$

که $F(s)=1$ ، $k=20N/m$ ، $b=10N.s/m$ ، $M=1kg$ می‌باشد.

فرض کنید که فرکانس پهنای باند سیستم حلقه بسته بزرگتر از 1rad/sec است. ما زمان نمونه برداری را برابر

$$T_s=\frac{1}{1000}\sec$$

در نظر می‌گیریم. حال دستورات زیر را در یک m-file وارد کنید:

```
M=1;b=10;k=20;
num=[1];
den=[M b k];
Ts=1/100;
[numDz,denDz]=c2dm(num,den,Ts,'zoh')
```

بعد از اجرای برنامه، تابع تبدیل دیجیتالی سیستم به دست می‌آید. این تابع به صورت زیر می‌باشد:

$$\frac{X(z)}{F(z)}=\frac{0.0001(0.4837z+0.4678)}{z^2-1.902z+0.9048}$$

۳-۲-۵ فضای حالت

فرض کنید که مدل فضای حالت یک سیستم پیوسته به صورت زیر باشد:

$$\begin{bmatrix}\dot{x}\\ \dot{v}\end{bmatrix}=\begin{bmatrix}0 & 1\\ \frac{-k}{M} & \frac{-b}{M}\end{bmatrix}\begin{bmatrix}x\\ v\end{bmatrix}+\begin{bmatrix}0\\ \frac{1}{M}\end{bmatrix}F$$

$$y=\begin{bmatrix}1 & 0\end{bmatrix}\begin{bmatrix}x\\ v\end{bmatrix}+0\times F$$

حال دستورات زیر را در یک M-file وارد کنید:

```
M=1;b=10;k=20;
A=[ 0      1
   -k/m   -b/M]
B=[0
   1/M]
C=[1 0];
D=[0];
Ts=1/100;
[F,G,H,J] = c2dm (A,B,C,D,Ts,'zoh')
```

بعد از اجرای برنامه، مدل فضای حالت گسسته به صورت زیر به‌دست می‌آید:

$$\begin{bmatrix}x(k)\\ v(k)\end{bmatrix}=\begin{bmatrix}0.999 & 0.0095\\ -0.1903 & 0.9039\end{bmatrix}\begin{bmatrix}x(k-1)\\ v(k-1)\end{bmatrix}+\begin{bmatrix}0\\ 0.0095\end{bmatrix}[F(k-1)]$$

$$y=\begin{bmatrix}1 & 0\end{bmatrix}\begin{bmatrix}x(k-1)\\ v(k-1)\end{bmatrix}+0\times[F(k-1)]$$

۳-۲-۶ بررسی پایداری و پاسخ حالت گذرا

برای سیستم‌های پیوسته، رفتار سیستم از روی موقعیت قطب‌های آن سیستم در صفحه S قابل بررسی است. به طور مثال اگر یک قطب در سمت راست محور موهومی باشد در آن صورت سیستم ناپایدار خواهد بود. برای

آنالیز رفتار سیستم‌های گسسته نیز ما می‌توانیم موقعیت قطب ها را در صفحه z بررسی کنیم. مشخصات یک سیستم در صفحه z را می‌توان با رابطه $z = e^{sT}$ با صفحه s ارتباط داد.

که در رابطه بالا:

T زمان نمونه برداری بر حسب sec/sample

S: موقعیت در صفحه s

Z : موقعیت در صفحه z

محدوده پایداری در صفحه z آنقدرگسترده نیست و فقط یک دایره واحد است $|z| = 1$ و سیستم در صورتی پایدار است که تمامی قطب ها در داخل دایره واحد قرار بگیرند. زمانی که یکی از قطب ها خارج این دایره واحد باشد درآن صورت سیستم ناپایدار است. برای آنالیز پاسخ حالت گذرا از روی موقعیت قطب ها در صفحه z، ازسه معادله زیر که در سیستم‌های پیوسته مورد استفاده قرارگرفتند، در اینجا نیز استفاده می‌کنیم.

$$\zeta\omega_n \geq \frac{4.6}{T_s} \qquad (۱۲-۳)$$

$$\omega_n \geq \frac{1.8}{T_r} \qquad (۱۳-۳)$$

$$\zeta \geq \sqrt{\frac{(\ln M_p / \pi)^2}{1+(\ln M_p / \pi)^2}} \qquad (۱۴-۳)$$

که در آن ζ، نسبت میرایی، ω_n، فرکانس طبیعی با واحد $rad/\sec$ ، T_s زمان نشست، T_rزمان خیز، و M_P حداکثر جهش می‌باشند.

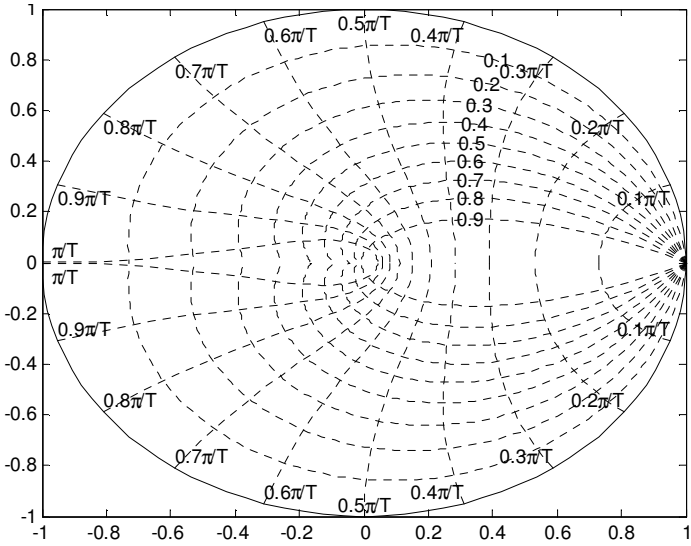
نکته مهم: فرکانس طبیعی در حوزه Z با واحد rad/sample است و وقتی می‌خواهیم از معادلات بالا استفاده کنیم باید از واحد rad/s استفاده شود.

فرض کنید تابع تبدیل سیستم در حوزه گسسته به صورت زیر است:

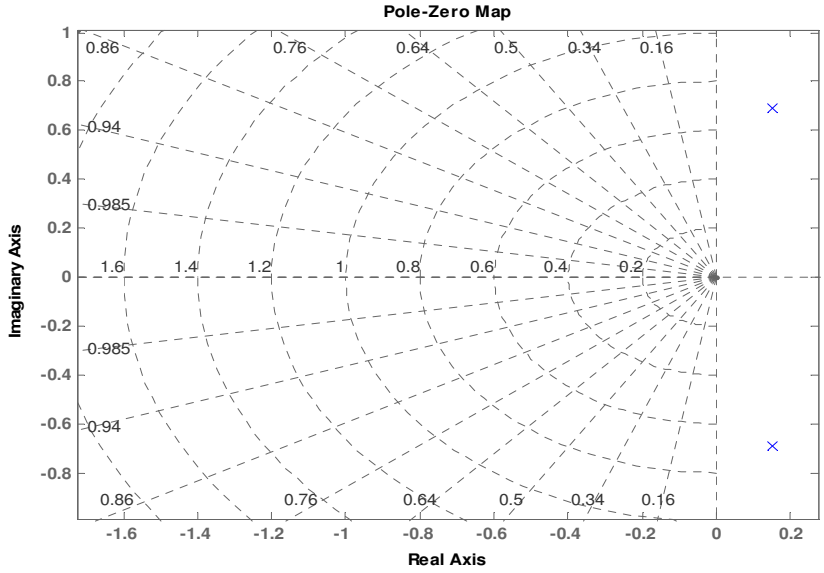
$$\frac{Y(z)}{F(z)} = \frac{1}{z^2 - 0.3z + 0.5}$$

دستورات زیر را در یک m-file وارد کرده و برنامه را اجرا کنید. بعد از اجرای برنامه، نمودار زیر با خطوط ثابت نسبت میرایی و فرکانس طبیعی حاصل می‌شود.

```
numDz=[1];
denDz=[1 -0.3 0.5];
pzmap(numDz,denDz)
axis([-1 1 -1 1])
zgrid
```



شکل (۳-۲۰) خطوط ثابت نسبت میرایی و فرکانس طبیعی



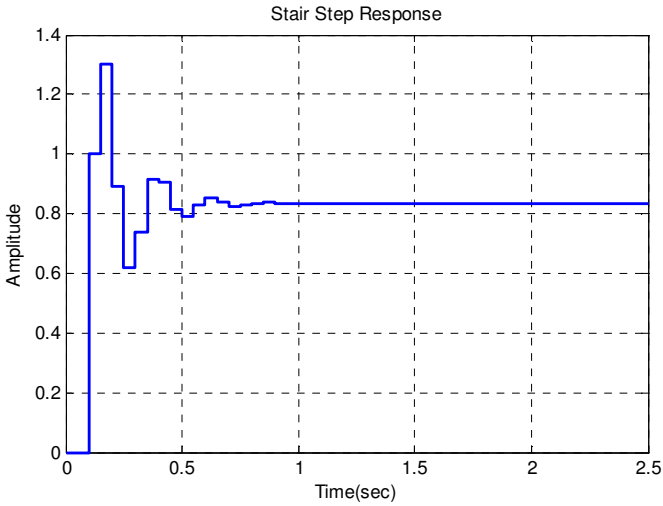
شکل (۳-۲۱) مکان قطب ها و صفرها

از نمودار مشاهده می‌شود که قطب‌ها تقریباً در فرکانس طبیعی $\frac{0.45\pi}{T}\text{rad/sample}$ و نسبت میرایی برابر 0.25 قرار گرفته اند. با فرض اینکه زمان نمونه برداری برابر 0.05sec باشد در آن صورت فرکانس طبیعی برابر 28.2rad/sec به‌دست می‌آید.

با توجه به سه معادلهٔ بالا، زمان خیز برابر 0.06 ثانیه و زمان نشست برابر 0.65 ثانیه و حداکثر جهش برابر 45% به‌دست می‌آید (0.45 بیشتر از مقدار حالت ماندگار).

حال اجازه بدهید پاسخ پله سیستم را به‌دست آوریم و ببینیم که مقادیر به‌دست آمده درست هستند یاخیر. پس دستورات زیر را در یک M-file وارد کنید. بعد از اجرای برنامه باید چنین نموداری حاصل شود:

```
[x] = dstep (numDz,denDz,51);
t = 0:0.05:2.5;
stairs (t,x)
```

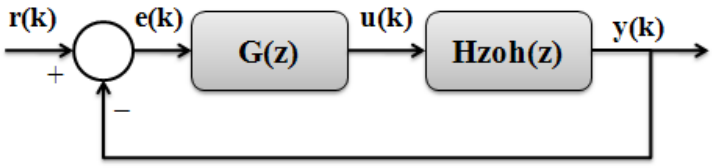


شکل (۳-۲۲) پاسخ پله سیستم

همانطور که از روی شکل مشخص است، برای مقادیر جهش ،زمان خیز، و زمان نشست همان مقادیری به‌دست آمدند که انتظار می‌رفت.

۳-۲-۷ بررسی مکان هندسی ریشه ها در حالت گسسته

به محل ریشه های معادله مشخصه یک سیستم کنترل، وقتی پارامتر k از صفر تا بی نهایت تغییر می‌کند، مکان هندسی ریشه های آن معادله گفته می شود. معادله مشخصه سیستم با فیدبک واحد به صورت زیر می‌باشد:



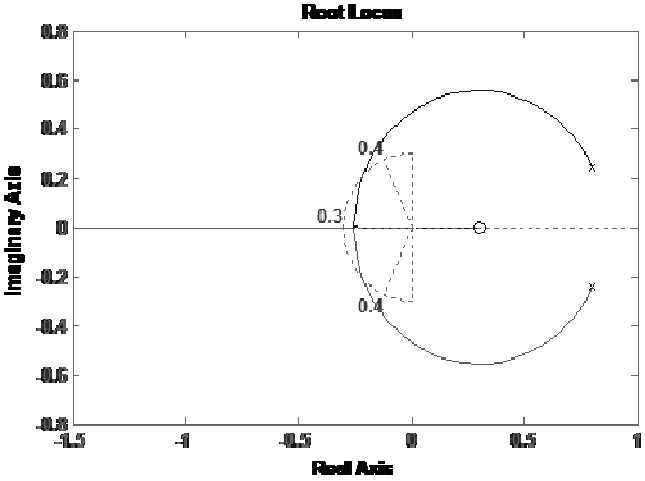
$$1 + K.G(z).Hzoh(z) = 0 \qquad (۱۵-۳)$$

که در آن $G(z)$ جبران کننده پیاده سازی شده در کنترل کننده دیجیتالی و $Hzoh(z)$ تابع تبدیل سیستم در حوزه z است. رسم نمودار Loci در حوزه z دقیقاً مشابه نمودار Locus در حوزه s است. با دستور zgrid(zeta,wn) می‌توانیم خطوط ثابت نسبت میرایی و فرکانس طبیعی را رسم کنیم. فرض کنید تابع تبدیل گسسته سیستم به صورت زیر باشد:

$$\frac{Y(z)}{F(z)} = \frac{z-0.3}{z^2-1.6z+0.7}$$

و می‌خواهیم نسبت میرایی بزرگتر از 0.6 و فرکانس طبیعی بزرگتر از 0.4rad/sample باشد. دستورات زیر را در یک m-file وارد کرده و برنامه را اجرا کنید.

```
numDz=[1 -0.3];denDz=[1 -1.6 0.7];
rlocus (numDz,denDz)
axis ([-1 1 -1 1])
zeta=0.4; Wn=0.3;
zgrid (zeta,Wn)
```



شکل (۳-۲۳) مکان هندسی ریشه‌ها

از روی شکل کاملاً مشخص است که سیستم پایدار است، چون تمامی قطب ها در داخل دایره واحد قرار گرفته-اند. فرکانس طبیعی نیز بزرگتر از 0.3 است و از طرفی قطب ها داخل خطوط zeta=0.4 قرار گرفته اند و تقریباً نسبت میرایی بزرگتر از 0.6 می‌باشد.

۳-۲-۸ بررسی پاسخ حالت گذرا[1]

در این قسمت قصد داریم تا کمی بیشتر درباره موقعیت قطب ها و به‌دنبال آن پاسخ حالت گذرا بحث کنیم.

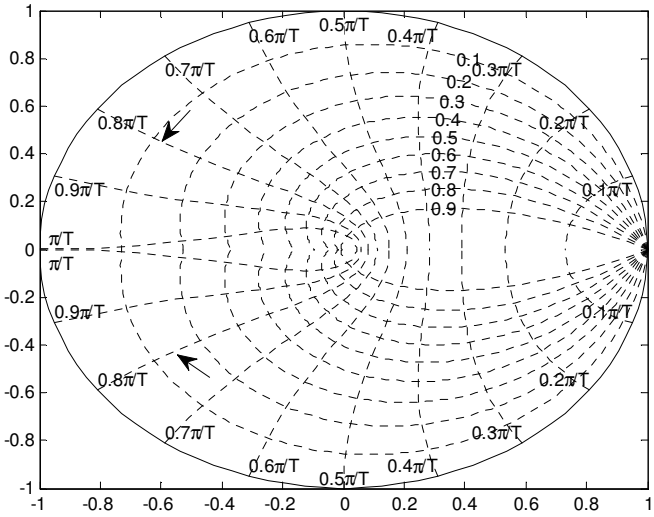
[1] Transient Response

میرایی کم ($\omega_n = \frac{4\pi}{5T}$ ، $\zeta = 0.1$): فرض کنید تابع تبدیل گسسته سیستم به صورت زیر است:

$$\frac{Y(z)}{F(z)} = \frac{1}{z^2 + 1.2z + 0.57}$$

دستورات زیر، موقعیت قطب‌های تابع را نشان می‌دهند. پس دستورات زیر را در یک M-file وارد کنید. بعد از اجرای برنامه نمودار (۳-۲۴) حاصل می‌شود.

```
numDz=[1];
denDz=[1 1.2 0.57];
[poles,zeros] = pzmap (numDz,denDz)
pzmap (numDz,denDz)
axis([-1 1 -1 1])
zgrid
```

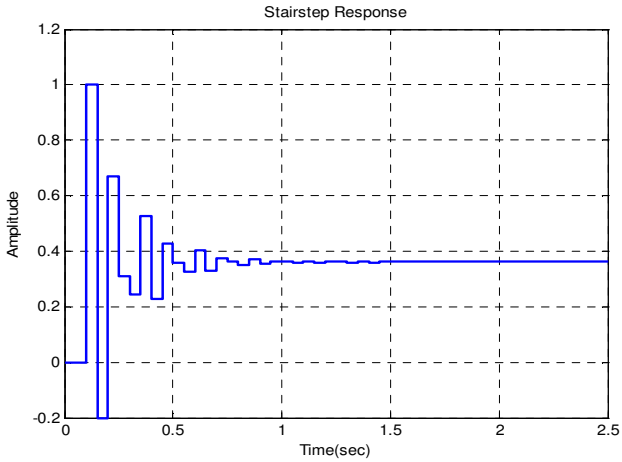


شکل (۳-۲۴) مکان قطب ها و خطوط ثابت نسبت میرایی و فرکانس طبیعی

از روی نمودار مشخص است که قطب ها در فرکانس طبیعی $\omega_n = \frac{4\pi}{5T}$ و ثابت میرایی $\zeta = 0.1$ واقع شده‌اند.

در اینجا فرض کنید که زمان نمونه برداری برابر 0.05 است و با استفاده از معادلات بالا (dig-1) زمان خیز برابر 0.03 ثانیه ،زمان نشست 0.9 ثانیه و جهش 70% به‌دست می‌آید. حال پاسخ پله سیستم را حساب کنید. برای انجام این کار دستورات زیر را به m-file اضافه کنید.

```
[x] = dstep (numDz,denDz,51);
t = 0:0.05:2.5;
stairs (t,x)
```



شکل (۳-۲۵) پاسخ پله سیستم

نمودار شکل (۳-۲۵) نشان می‌دهد که برای زمان خیز ، زمان نشست، و جهش مقداری به‌دست آمد که ما انتظار داشتیم.

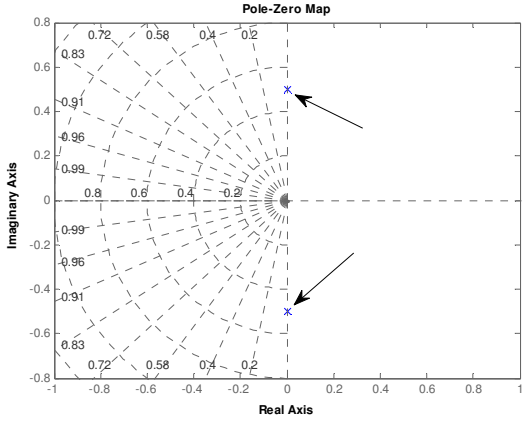
میرایی متوسط ($\zeta = 0.4, \omega_n = \frac{11\pi}{20T}$): فرض کنید تابع تبدیل گسسته سیستم به صورت زیر است.

$$\frac{Y(z)}{F(z)} = \frac{1}{z^2 + 0.25}$$

دستورات زیر موقعیت قطب‌های تابع را به شما نشان می‌دهند؛ پس دستورات زیر را در یک m-file وارد کنید.

بعد از اجرای برنامه نمودار (۳-۲۶) حاصل می‌شود.

```
numDz=[1];denDz=[1 0 0.25];
[poles,zeros]=pzmap(numDz,denDz)
pzmap(numDz,denDz)
axis([-1 1 -1 1]);zgrid
```

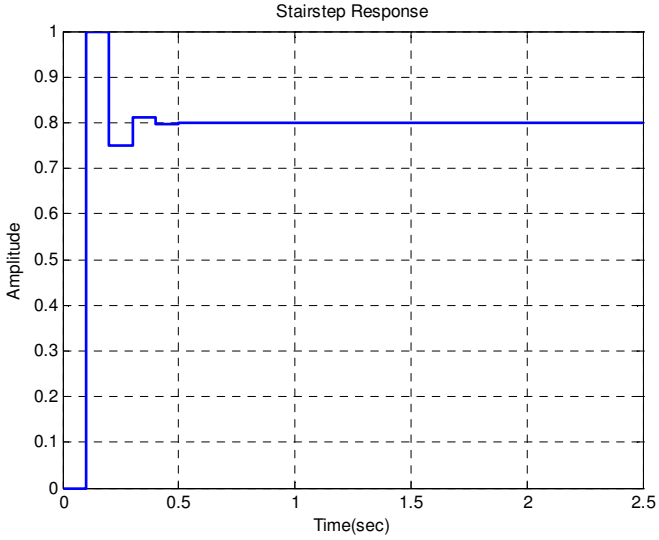


شکل (۳-۲۶) مکان قطب ها و خطوط ثابت نسبت میرایی و فرکانس طبیعی

از روی نمودار مشخص است که قطب ها در فرکانس طبیعی $\omega_n = \frac{11\pi}{20T}$ و ثابت میرایی $\zeta = 0.4$ واقع شده-اند. در اینجا فرض کنید که زمان نمونه برداری برابر 0.05 است و با استفاده از معادلات بالا (dig-1) زمان خیز برابر 0.05 ثانیه ،زمان نشست 0.3 ثانیه، و جهش 25% به‌دست می‌آید.

حال پاسخ پله سیستم را حساب کنید. برای انجام این کار دستورات زیر را به m-file اضافه کنید.

```
[x] = dstep (numDz,denDz,51);
t = 0:0.05:2.5;
stairs (t,x)
```



شکل (۳-۲۷) پاسخ پله سیستم

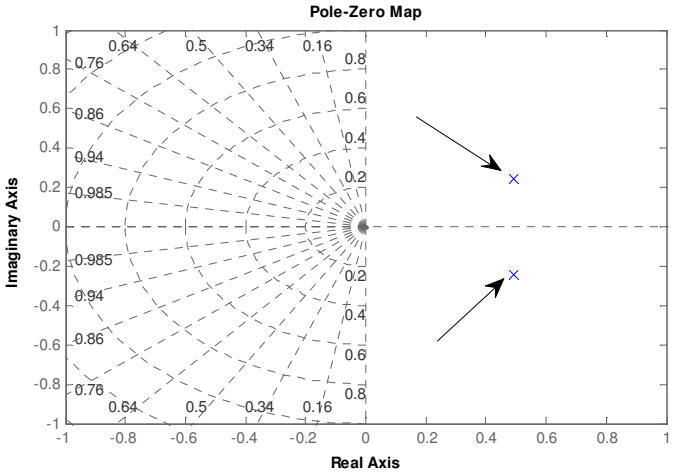
نمودار شکل (۳-۲۷) نشان می‌دهد که برای زمان خیز، زمان نشست، و جهش مقادیری به‌دست آمدند که ما انتظار داشتیم.

میرایی زیاد ($\omega_n = \frac{\pi}{4T}$,$\zeta = 0.8$): فرض کنید تابع تبدیل گسسته سیستم، به صورت زیر است.

$$\frac{Y(z)}{F(z)} = \frac{1}{z^2 - 0.98z + 0.3}$$

مطابق حالت‌های قبل دستورات زیر را در یک M-file وارد کنید. بعد از اجرای برنامه نمودار زیر حاصل میشود.

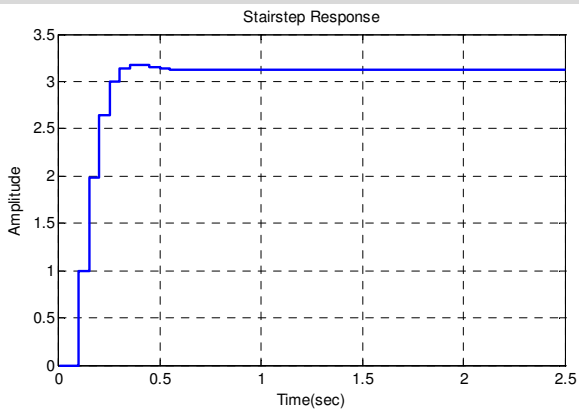
```
numDz=[1];denDz=[1 -0.98 0.3];
[poles, zeros]=pzmap (numDz,denDz)
pzmap (numDz,denDz)
axis([-1 1 -1 1]); zgrid
```



شکل (۳-۲۸) مکان قطب ها و خطوط ثابت نسبت میرایی و فرکانس طبیعی

از روی نمودار مشخص است که قطب ها در فرکانس طبیعی $\omega_n = \frac{\pi}{4T}$ و ثابت میرایی $\zeta = 0.8$ واقع شده اند. در اینجا فرض کنید که زمان نمونه برداری برابر 0.05 است و با استفاده از معادلات بالا (dig-1) زمان خیز برابر 0.1 ثانیه، زمان نشست 0.36 ثانیه، و جهش 1٪ به‌دست می‌آید. حال پاسخ پله سیستم را حساب کنید. برای انجام این کار دستورات زیر را به m-file اضافه کنید.

```
[x] = dstep (numDz,denDz,51);
t = 0:0.05:2.5;
stairs (t,x)
```

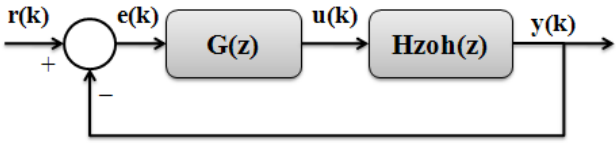


شکل (۳-۲۹) پاسخ پله سیستم

با استفاده از این سه مثال نشان داده شده که با استفاده از موقعیت قطب ها می‌توانیم پاسخ حالت گذرای سیستم را تخمین بزنیم. این آنالیز در مکان هندسی ریشه‌ها برای به‌دست آوردن یک پاسخ مناسب، بسیار کارآمد می‌باشد.

۳-۲-۹ معادلات تفاضلی

از آنجاکه سیستم‌های پیوسته با معادلات دیفرانسیل توصیف می‌شوند، سیستم‌های گسسته نیز با معادلات تفاضلی بیان می‌شوند. باتوجه به شکل زیر معادلات تفاضلی رابطه بین سیگنال‌های ورودی $e(k)$ و سیگنال‌های خروجی $u(k)$ را نشان می‌دهند. فرض کنید که سیگنال خروجی k ام را می‌خواهیم. برای رسیدن به این خروجی نیاز داریم که سیگنال‌های ورودی بین $e(0)$ و $e(k)$ و سیگنال‌های خروجی بین $u(0)$ و $u(k-1)$ را داشته باشیم.



$$u(k)=f(e(0),e(1),\ldots,e(k);u(0),u(1)\ldots,u(k-1)) \quad (۱۶-۳)$$

فرض می‌کنیم که تابع f خطی بوده و به تعداد محدودی از سیگنال‌های u,e وابسته است. پس ساختار پایه‌ای این معادله تفاضلی به صورت زیر نوشته می‌شود:

(۱۷-۳)

$$u(k)=b_0.e(0)+b_1.e(1)+\ldots\ b_k.e(k)-a_0.u(0)-a_1.u(1)\ldots.-a_{k-1}u(k-1)$$

به طور مثال یک معادله تفاضلی به صورت زیر نوشته می‌شود:

$$u(k)=0.75u(k-1)+e(k)-0.95e(k-1) \quad (۱۸-۳)$$

۳-۲-۱۰ پیدا کردن تابع تبدیل با استفاده از تبدیل Z

تبدیل z به صورت زیر تعریف می‌شود:

$$Z\{f(k)\}=F(z)=\sum_{k=0}^{\infty}f(k)z^{-k} \quad (۱۹-۳)$$

که f(k) دامنه نمونه برداری و k=0,1,2… زمان‌های نمونه برداری گسسته می‌باشند.

$$Z\{f(k-m)\}=z^{-m}F(z) \quad (۲۰-۳)$$

فرض کنید که قصد داریم تابع تبدیل سیستم تعریف شده با معادله تفاضلی (۳-۱۸) را به‌دست آوریم. در ابتدا از طرفین تبدیل Z می‌گیریم. باید به معادله زیر برسید:

$$U(z)=0.75z^{-1}U(z)+E(z)-0.95z^{-1}E(z)$$

بعد از چند مرحله محاسبات جبری تابع تبدیل فرم گسسته به صورت $\frac{U(z)}{E(z)}=\frac{z-0.95}{z-0.75}$ به‌دست می‌آید.

۳-۲-۱۱ به‌دست آوردن معادلات فضای حالت از معادلات اختلافی

ساختار فضای حالت گسسته به صورت زیر می‌باشد:

$$\begin{cases} x(k) = Fx(k-1) + Ge(k-1) \\ y(k-1) = Hx(k-1) + Je(k-1) \end{cases} \quad (۲۱-۳)$$

که در آن ماتریس‌های F,G,H,J حالات گسسته ماتریس‌های A,B,C,D می‌باشند. با فرض $e(k) = \varepsilon(k-1)$ مدل فضای حالت سیستم را می‌توان به صورت زیر نوشت :

$$\begin{bmatrix} e(k) \\ u(k) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & 0.75 \end{bmatrix} \begin{bmatrix} \varepsilon(k-1) \\ u(k-1) \end{bmatrix} + \begin{bmatrix} 0 \\ -0.95 \end{bmatrix} [e(k-1)]$$

$$y(k-1) = [0 \quad 1] \begin{bmatrix} \varepsilon(k-1) \\ u(k-1) \end{bmatrix} + [0][e(k-1)]$$

۳-۲-۱۲ نمایش در محیط MATLAB

تابع تبدیل: تابع تبدیل گسسته نیز مشابه تابع تبدیل سیستم‌های پیوسته به‌دست می‌آید. ماتریس صورت و مخرج کسر را باید به صورت توان‌های کاهشی z وارد کنید.

```
numDz=[1 -0.95];
denDz=[1 -0.75];
sys = tf(numDz,denDz,-1);
```

از 1- موقعی استفاده می‌شود که زمان نمونه برداری مشخص نباشد.

فضای حالت: برای به‌دست آوردن مدل فضای حالت سیستم دقیقاً مشابه مدل‌های پیوسته عمل می‌کنیم. به طور مثال مدل فضای حالت سیستم بالا به صورت زیر به‌دست می‌آید:

$$F = \begin{bmatrix} 1 & 0 \\ 1 & 0.75 \end{bmatrix}$$

$$G = \begin{bmatrix} 0 \\ -0.95 \end{bmatrix}$$

```
H=[0 1];
J=[0];
sys = ss(F,G,H,J,-1);
```

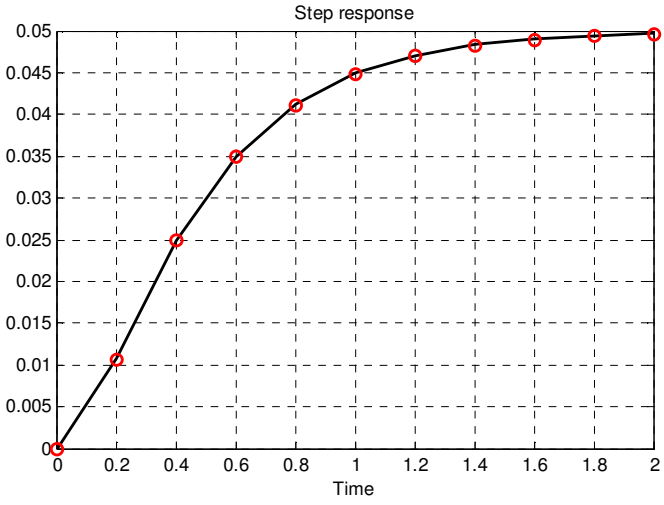
از 1- زمانی استفاده می‌شود که زمان نمونه برداری مشخص نباشد.

۳-۲-۱۳ تأخیر انداختن با استفاده از نگهدارنده ها

یکی از مهمترین مشکلاتی که در پیاده سازی سیستم‌های کنترل دیجیتالی با آن روبرو هستیم اثر تأخیر اندازی Holding است. دستورات زیر را در یک m-file بنویسید:

```
num=[1];
den=[1 10 20];
numDz=[0.0107 0.0055];
denDz=[1 -0.8106 0.1353];
t=0:0.2:2;
step (num,den,t)        %plots continuous output response
```

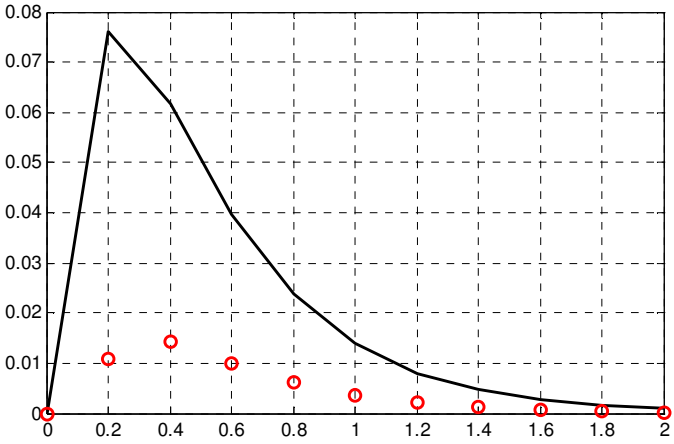
```
hold
[x]=dstep (numDz,denDz, 11);
plot (t,x,'ro')          %plots discrete output response
hold off
```



شکل (۳-۳۰) انطباق پاسخ سیستم گسسته بر روی پاسخ حالت پیوسته

از روی نمودار شکل (۳-۳۰) مشخص است که پاسخ سیستم گسسته بر روی پاسخ حالت پیوسته منطبق است، زیرا تابع پله ورودی با زمان تغییر نمی‌کند. حال اگر تابع ورودی یک تابع متغیر با زمان باشد این دو نمودار بر هم منطبق نخواهند شد. برای نشان دادن این موضوع یک ورودی ضربه به سیستم اعمال کنید. پس در برنامه m-file تغییرات زیر را اعمال نمائید:

```
step to impulse
dstep to dimpulse
```



شکل (۳-۳۱) عدم انطباق پاسخ سیستم گسسته بر روی پاسخ حالت پیوسته با ورودی ضربه‌ای

همانطور که مشخص است خروجی گسسته با خروجی سیستم پیوسته یکی نمی‌باشد و پاسخ گسسته نسبت به پاسخ پیوسته تأخیر دارد.

۳-۲-۱۴ پاسخ پله سیستم گسسته

برای رسم پاسخ پله سیستم گسسته از دستور dstep و stairs استفاده می‌کنیم. در دستور dstep از عدد N استفاده می‌کنیم که N تعداد نمونه‌هایی است که توسط کاربر تعیین می‌شود.

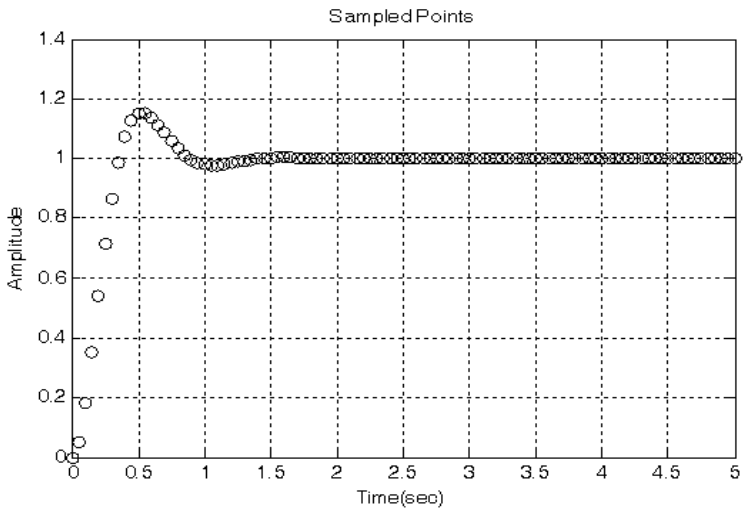
تابع تبدیل: فرض کنید تابع تبدیل سیستم گسسته به صورت زیر می‌باشد:

$$\mathrm{Hzoh(z)} = \frac{\mathrm{z}-0.95}{\mathrm{z}-0.75}$$

از دستور dstep برای به‌دست آوردن برداری از نقاط نمونه برداری استفاده می‌کنیم. دستورات زیر را در یک M-file وارد کرده و برنامه را اجرا کنید. بعد از اجرا ۱۰۱ نقطه به‌دست می‌آید. این دستور از بردار زمانی t استفاده نمی‌کند بلکه به جای آن از نقاط نمونه برداری شده استفاده می‌کند.

```
numDz=[0.05 0.05];
denDz=[1 -1.6 0.7];
IU=1;   %This is the input value
N=101;
[x] = dstep (IU*numDz,denDz, N);
```

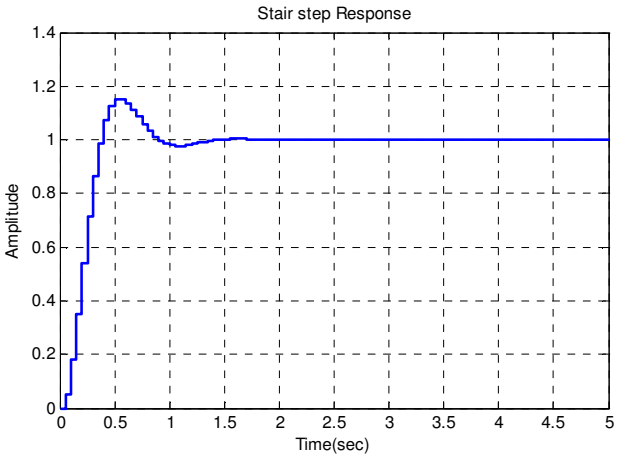
برای به‌دست آوردن پاسخ به صورت تابعی از زمان باید رابطه بین تعداد نمونه ها و زمان نمونه برداری Ts را بدانید. اگر فرض کنیم سیستم کنترل دیجیتالی زمان نمونه برداری (Ts(sec/sample دارد و تعداد نمونه ها برابر N باشد. مدت زمان کل نمونه برداری برابر $T_s \times (N-1)$ می‌شود. در مثال قبل تعداد نمونه‌ها برابر 101 و زمان نمونه‌برداری $0.05\,\mathrm{sec}/\,sample$ می‌باشد پس زمان کل برابر 5 ثانیه است.



شکل (۳-۳۲) پاسخ پله سیستم به صورت تابعی از زمان

با استفاده از دستور stairs این نقاط را به هم وصل می‌کنیم، زیرا خروجی $y(k)$ از ZOH عبور کرده است.
برای انجام این کار دستورات زیر را در ادامه m-file بنویسید:

```
t=0:0.05:5;
stairs (t,x);
```



شکل (۳-۳۳) پاسخ پله سیستم به صورت تابعی از زمان پیوسته

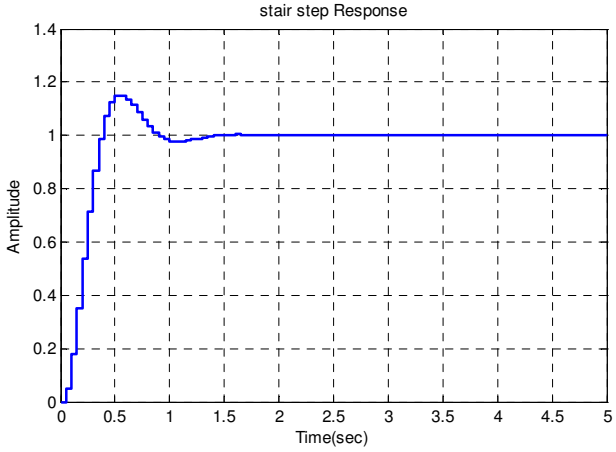
فضای حالت: برای به‌دست آوردن پاسخ پله از مدل فضای حالت نیز می‌توان استفاده کرد و روشی مشابه تابع تبدیل برای آن انجام می‌دهیم. تنها تفاوت این است که به جای صورت و مخرج از ماتریس‌های F,G,H,J استفاده می‌کنیم. فرض کنید مدل فضای حالت به صورت زیر می‌باشد:

$$x(k)=\begin{bmatrix}1 & 0\\ 1 & 0.75\end{bmatrix}x(k-1)+\begin{bmatrix}0\\ 1\end{bmatrix}u(k-1)$$

$$y(k-1)=[0.05\ \ 0.05]x(k-1)+[0].u(k-1)$$

دستورات زیر پاسخ پله سیستم به فرم فضای حالت را رسم می‌کند:

```
F = [1.6  -0.7
     1     0 ]
G = [1
     0]
H = [0.05 0.05];
J = [0];
N=101;
IU=1;   %Input value
[x] = dstep (F,G,H,J,IU,N)
t=0:0.05:5;
stairs (t,x);
```



شکل (۳-۳۴) پاسخ پله از مدل فضای حالت

۳-۲-۱۵ جبران کننده PID با تقریب دوخطی[1]

روش‌های زیادی برای نگاشت از صفحه s به صفحه z وجود دارد. به طور مثال از روشهای تقریب تفاضلی رو به جلو[2] و رو به عقب[3] می‌توان استفاده کرد. یکی دیگر از این روش ها، روش دوخطی است که نیم صفحه چپ s را به دایره واحد در حوزه z می‌نگارد. این تبدیل دوخطی به صورت زیر تعریف می‌شود:

$$s=\frac{2}{T_s}\frac{z-1}{z+1} \qquad (۲۲-۳)$$

که T_s زمان نمونه برداری می‌باشد.

در حالت پیوسته تابع تبدیل کنترل کننده PID به صورت زیر می‌باشد:

$$PID: K_p+\frac{K_I}{s}+K_D s \qquad (۲۳-۳)$$

برای نگاشتن به حوزه z باید تابع تبدیل دوخطی را در تابع تبدیل کنترل کننده PID جایگزین کنیم.بنابراین تابع تبدیل PID در حوزه z به صورت زیر خواهد بود:

$$PID: K_p+\frac{K_I}{\frac{2}{T_s}\frac{z-1}{z+1}}+K_D\times\frac{2}{T_s}\frac{z-1}{z+1}$$

$$PID: \frac{(K_p+\frac{K_I.T_s}{2}+\frac{2K_D}{T_s}).z^2+(K_I.T_s-\frac{4K_D}{T_s}).z+(-K_p+\frac{K_I T_s}{2}+\frac{2K_D}{T_s})}{z^2-1} \qquad (۲۴-۳)$$

[1] Bilinear
[2] Forward
[3] backward

برای به‌دست آوردن تابع تبدیل گسسته کنترل کننده PID دستورات زیر را در یک m-file وارد کنید:

```
%Discrete PID Controller
numcz = [Kp+(Ki*Ts*0.5)+((2*Kd)/Ts)  (Ki*Ts)-(4*(Kd/Ts))
Kp+(Ki*Ts*0.5)+((2*Kd)/Ts)]
dencz = [1 0 -1]
```

بعلاوه با استفاده از دستور c2dm نیز می‌توان این نگاشت را بدون هیچگونه محاسبه‌ای انجام داده، و کنترل کننده PID پیوسته را به کنترل کننده گسسته تبدیل کنیم. برای تقریب دوخطی باید از روش tustin استفاده کنید.

```
%Discrete PID Controller using c2dm command
[dencz,numcz] = c2dm([1 0],[Kd Kp Ki],Ts,'tustin')
```

توجه کنید که از آنجایی که تابع تبدیل کنترل کننده PID کاملاً سره نیست، بنابراین صورت و مخرج معکوس شده‌اند. تعداد صفر ها بیشتر از تعداد قطب ها می‌باشد، اما MATLAB اجازه این کار را نمی‌دهد. بنابراین با عوض کردن صورت و مخرج دستور c2dm را فریب می‌دهیم.

۳-۲-۱۶ خطای حالت ماندگار دیجیتالی

در طراحی سیستم‌های پیوسته اغلب از تئوری مقدار نهایی استفاده می‌کنیم:

$$\lim_{t\to\infty} x(t) = X_{ss} = \lim_{s\to 0} s.x(s) \qquad (۲۵-۳)$$

برای پیدا کردن خطای حالت ماندگار سیستم یک تئوری مقدار نهایی در حوزه گسسته وجود دارد:

$$\lim_{k\to\infty} x(k) = X_{ss} = \lim_{z\to 1} (1-z^{-1})x(z) \qquad (۲۶-۳)$$

البته اگر تمام قطب‌های $(1-z^{-1})x(z)$ داخل دایره واحد قرار بگیرند.

به‌طور مثال فرض کنید تابع تبدیل گسسته به صورت زیر است:

$$\frac{x(z)}{u(z)} = \frac{z+0.5}{z^2-0.6z+0.3}$$

اول باید قطب‌های تابع تبدیل را به‌دست آوریم و ببینیم که آیا داخل دایره واحد قرار گرفته است یا خیر. برای پیدا کردن قطب ها هم می‌توانیم به صورت دستی و هم با استفاده از دستورات MATLAB این کار را انجام دهیم. هر دو قطب داخل دایره واحد قرار گرفته اند. پس اکنون می‌توانیم تئوری مقدار نهایی را اعمال کنیم.

```
numDz=[1 0.5];
denDz=[1 -0.6 0.3];
[p,z]=pzmap(numDz,denDz)
```

```
p =
0.3000 + 0.4583i
0.3000 - 0.4583i
z =
-0.5000
```

۳-۲-۱۷ پیدا کردن خطای حالت ماندگار به ورودی پله

اجازه بدهید $u(z)$ را ورودی پله واحد در نظر بگیریم پس:

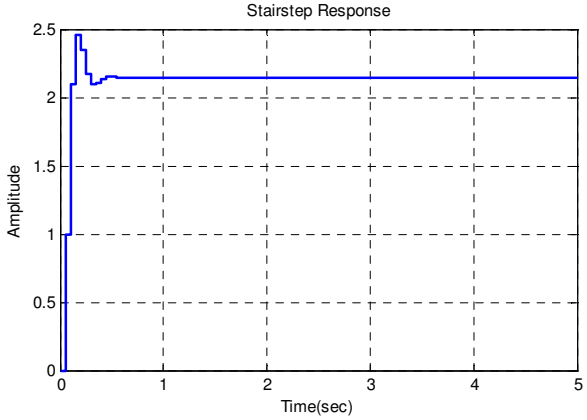
$$x(z)=\frac{z+0.5}{z^2-0.6z+0.3}\times\frac{1}{1-z^{-1}} \quad \text{و} \quad u(z)=\frac{1}{1-z^{-1}}$$

با اعمال تئوری مقدار نهایی داریم:

$$X_{ss}=\lim_{z\to 1}(1-z^{-1})x(z)=\lim_{z\to 1}(1-z^{-1})\times\frac{z+0.5}{z^2-0.6z+0.3}\times\frac{1}{1-z^{-1}}=2.14$$

پس مقدار حالت ماندگار برای سیستم گسسته به ورودی پله واحد برابر 2.14 است. یعنی خطا برابر 114% است. دستورات زیر را در یک m-file کپی کنید و پاسخ پله را مشاهده کنید:

```
numDz=[1 0.5];
denDz=[1 -0.6 0.3];
[x]=dstep (numDz,denDz, 101);
t=0:0.05:5;
stairs (t,x)
```



شکل (۳-۳۵) خطای حالت ماندگار به ورودی پله

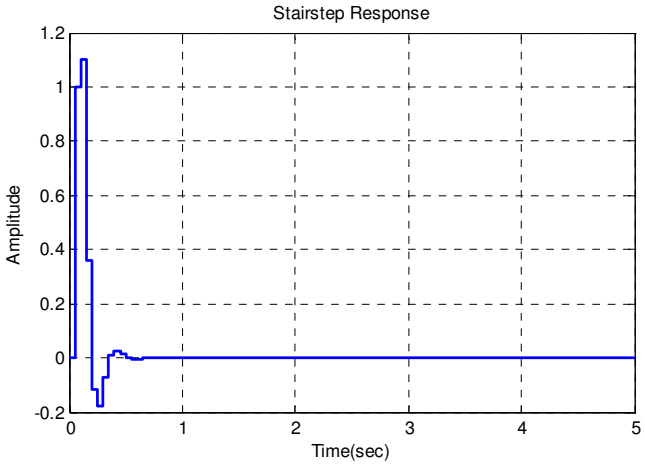
۳-۲-۱۸ پیدا کردن خطای حالت ماندگار به ورودی ضربه

برای ورودی ضربه، $u(z)=1$ ،و با اعمال قضیه مقدار نهایی داریم:

$$\lim_{z\to 1}(1-z^{-1})\times\frac{z+0.5}{z^2-0.6z+0.3}\times 1=0$$

بنابراین مقدار حالت ماندگار به ورودی ضربه برابر صفر است و خطای حالت ماندگارنیز برابر صفر خواهد شد.در دستورات بالا فرمان dstep را به dimpulse تغییر دهید و برنامه را اجرا کنید.

```
numDz=[1 0.5];denDz=[1 -0.6 0.3];
[x]=dimpulse (numDz,denDz, 101);
t=0:0.05:5;
stairs(t,x)
```

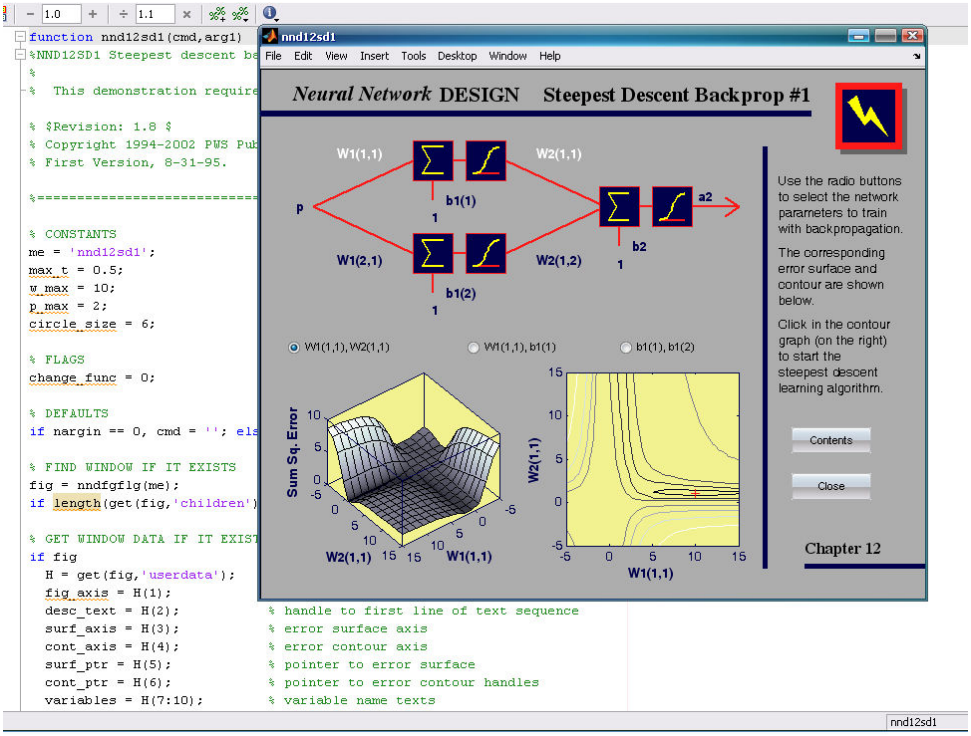


شکل (۳-۳۶) خطای حالت ماندگار به ورودی ضربه

در نهایت چنین خروجی‌ای حاصل می‌شود و همانطور که انتظار داشتیم خطای حالت ماندگار برابر صفر است.

قسمت سوم

- ✓ **فصل اول** شبکه عصبی
- ✓ **فصل دوم** فهرست دستورهای جعبه ابزار سیستم‌های کنترلی در نرم‌افزار **MATLAB**



یکی از موضوعات جدید و به روز علوم مهندسی که قادر به تخمین توابع غیرخطی پیچیده و شناسایی سیستم‌ها است، استفاده از سیستم‌های هوشمند است. در این قسمت، به آموزش کامل طراحی شبکه-های عصبی که به صورت On-line وOff-line کار می‌کنند و کاربرد آن در شناسایی سیستم‌ها پرداخته شده است. در انتهای این قسمت نیز فهرست دستورات کاربردی در MATLAB و کنترل سیستم‌ها آورده شده است.

فصل اول

شبکه عصبی

الف– شبکه عصبی چیست؟

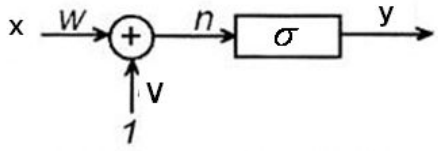
۱– ۱ مقدمه

شبکه‌های عصبی را می‌توان با اغماض زیاد، مدل‌های الکترونیکی از ساختار عصبی مغز انسان نامید. مکانیسم فراگیری و آموزش مغز اساساً بر تجربه استوار است. مدل‌های الکترونیکی شبکه‌های عصبی طبیعی نیز بر اساس همین الگو بنا شده‌اند و روش برخورد چنین مدل‌هایی با مسائل، با روش‌های محاسباتی که به‌طور معمول توسط سیستم‌های کامپیوتری در پیش گرفته شده‌اند، تفاوت دارد. این شبکه ها چیزی نیستند جز چند ماتریس که می‌توانند یاد بگیرند و تصمیم گیری کنند. از این شبکه‌ها می‌توانید در کارهایی که از یک الگوریتم دقیق پیروی نمی‌کنند، استفاده کنید.

یک نرون بیولوژیک، پس از دریافت سیگنال‌های ورودی (به شکل یک پالس الکتریکی) از سلول‌های دیگر، آن سیگنال‌ها را با یکدیگر ترکیب کرده و پس از انجام یک عمل[1] بر روی سیگنال ترکیبی، آن را به‌صورت خروجی ظاهر می‌سازد. نرون‌ها از چهار بخش اصلی ساخته شده‌اند. دندریت‌ها[2]، سوما[3]، اکسون[4]، و بالاخره سیناپس[5]. دندریت‌ها، همان اجزایی هستند که به‌شکل رشته‌های طویل از مرکز سلول به اطراف پراکنده می‌شوند. دندریت‌ها نقش کانال‌های ارتباطی را برای انتقال سیگنال‌های الکتریکی به مرکز سلول بر عهده دارند. در انتهای دندریت‌ها، ساختار بیولوژیکی ویژه‌ای به‌نام سیناپس واقع شده است، که نقش دروازه‌های اتصالی کانال‌های ارتباطی را ایفا می‌کند. در واقع ، سیگنال‌های گوناگون از طریق سیناپس‌ها و دندریت‌ها به مرکز سلول منتقل می‌شوند و در آنجا با یکدیگر ترکیب می‌شوند. دندریت‌ها که به‌صورت درخت گونه پخش هستند اطلاعات دریافتی به شکل سیگنال را دریافت نموده و آن را به هسته سلول هدایت می‌کنند. یک عمل جمع ساده از کل سیگنال‌ها بسته به وزن و شدت هر یک، در هسته انجام می‌گردد و نتیجه توسط اکسون هدایت می‌شود و بسته به شدت آن ممکن است پالس الکتریکی را از سیناپس با شدت بیشتر یا با شدت کمتر عبور دهند و یا ممکن است به دلیل ضعیف بودن بار الکتریکی آن را عبور ندهند.

[1] Operation
[2] Dendrite
[3] Soma
[4] Axon
[5] Synapse

در مدل‌سازی ریاضی می‌توان آن‌را یک عمل جمع معمولی در نظر گرفت، که پس از آن تابع ویژه‌ای بر روی سیگنال اثر داده شده و خروجی به شکل سیگنال الکتریکی متفاوتی از طریق اکسون (و سیناپس آن) به سلول‌های دیگر انتقال داده می‌شود. برای نشان دادن یک سیگنال ورودی از علامت x استفاده می‌شود. در واقع در این مدل، یک سیگنال ورودی پس از تقویت (یا تضعیف) شدن به اندازه پارامتر W، به‌صورت یک سیگنال الکتریکی با اندازه xW وارد نرون می‌شود. به‌دلیل ساده‌سازی مدل ریاضی، فرض می‌شود که در هسته سلول عصبی، سیگنال ورودی با سیگنال دیگری به اندازه V جمع می‌گردد. در واقع سیگنال V خود به معنی آن است که سیگنالی به اندازه واحد در پارامتری مانند V ضرب (تقویت یا تضعیف) می‌شود. مجموع حاصل، یعنی سیگنالی به اندازه xW + V، قبل از خارج شدن از سلول تحت عمل یا فرآیند دیگری واقع می‌شود، که در اصطلاح فنی به آن تابع انتقال[1] می‌گویند. ورودی این جعبه همان سیگنال xW + V است و خروجی آن یا همان خروجی سلول، با علامت y نشانه‌گذاری می‌شود. پارامتر W یا همان ضریبی که سیگنال ورودی x در آن ضرب می‌شود، در اصطلاح ریاضی به نام پارامتر وزن[2] خوانده می‌شود.



شکل(۱-۱) مدل ریاضی نرون

زمانی‌که از کنار هم قرار دادن تعداد بسیار زیادی از سلول‌های فوق یک شبکه عصبی بزرگ ساخته شود، شبکه‌ای در دست خواهیم داشت که رفتار آن علاوه بر تابع خروجی f، کاملاً به مقادیر W و V وابسته خواهد بود. در چنین شبکه بزرگی، تعداد بسیار زیادی از پارامترهای W و V باید توسط طراح شبکه مقداردهی شوند. این پروسه از کار، در اصطلاح دانش شبکه‌های عصبی، به فرآیند یادگیری معروف است. در واقع در یک آزمایش واقعی، پس از آن‌که سیگنال‌های ورودی چنین شبکه بزرگی اتصال داده شدند، طراح شبکه با اندازه‌گیری خروجی و با انتخاب پارامترهایW و V به‌گونه‌ای که خروجی مطلوب به‌دست آید، شبکه را آموزش می‌دهد. به این ترتیب پس از آنکه چنین شبکه‌ای به ازای مجموعه‌ای از ورودی‌ها برای ساختن خروجی‌های مطلوب آموزش دید، می‌توان از آن برای حل مسائلی که از ترکیب متفاوتی از ورودی‌ها ایجاد می‌شوند، بهره برد. اگر قرار باشد خروجی فقط یکی از مقادیر صفر یا یک را شامل شود ، در این حالت، خروجی چنین شبکه‌ای فقط می‌تواند بر حسب ورودی‌های متفاوت، مقدار یک یا صفر باشد.

در هر صورت، پس از آنکه ورودی‌ها با یکدیگر ترکیب شدند، سیگنال حاصل به واحد دیگری که در آن تابع انتقال به سیگنال اعمال می‌شود، هدایت می‌گردد. خروجی این بخش، سیگنال‌های حقیقی خواهند بود. بدین ترتیب جعبه‌ای در دست خواهیم داشت که تعداد n عدد سیگنال ورودی را به m عدد سیگنال خروجی تبدیل

[1] Transfer Function
[2] Weight

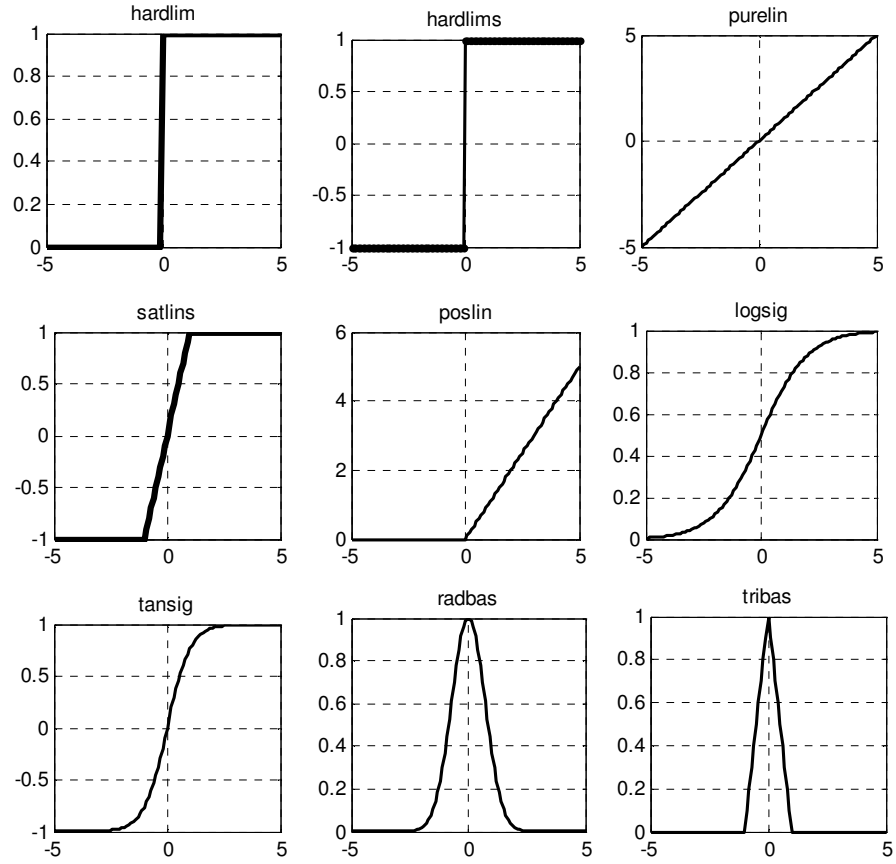
می‌کند. در عمل توابع انتقالی، معمولاً یکی از توابع سینوسی، تانژانت هذلولی، سیگموئید[1]، نظایر این‌هاست. این تابع انتقال، سیگنال خروجی واحد ترکیب را به سیگنال خروجی تبدیل می‌کند که مقدار (یا اندازه آن) می‌تواند بین صفر و یک باشد.

در شکل (۱-۱) تابع غیرخطی σ تابع محرکه[2] می‌باشد که هر یک از این توابع محرکه در شکل(۱-۲) نمایش داده شده‌اند، و بنا به کاربرد در زمینه‌های مختلف، انتخاب می‌شوند. در بسیاری از الگوریتم‌های آموزش شبکه عصبی مثل الگوریتم پس انتشار[3] خطا مشتق تابع σ نیاز می‌باشد، که در این‌گونه موارد تابع محرکه باید مشتق پذیر باشد. با استفاده از دستورات زیر، نمایشی از رفتار این توابع در محدوده 5- تا 5 را می‌توان دید.

```
n=-5:0.1:5;
subplot(3,3,1)
a1 = hardlim(n);
plot(n,a1,'k','linewidth',2.2)
title('hardlim')
subplot(3,3,2)
a2 = hardlims(n);
plot(n,a2,'k','linewidth',2.2)
title('hardlims')
subplot(3,3,3)
a3 = purelin(n);
plot(n,a3,'k','linewidth',2.2)
title('purelin')
subplot(3,3,4)
a4 = satlins(n);
plot(n,a4,'k','linewidth',2.2)
title('satlins')
subplot(3,3,5)
a5 = poslin(n);
plot(n,a5,'k','linewidth',2.2)
title('poslin')
subplot(3,3,6)
a6 = logsig(n);
plot(n,a6,'k','linewidth',2.2)
title('logsig')
subplot(3,3,7)
a7 = tansig(n);
plot(n,a7,'k','linewidth',2.2)
title('tansig')
```

[1] Sigmoid
[2] Activation Function
[3] Backpropagation Algorithm

```
subplot(3,3,8)
a8 = radbas(n);
plot(n,a8,'k','linewidth',2.2)
title('radbas')
subplot(3,3,9)
a9 = tribas(n);
plot(n,a9,'k','linewidth',2.2)
title('tribas')
```



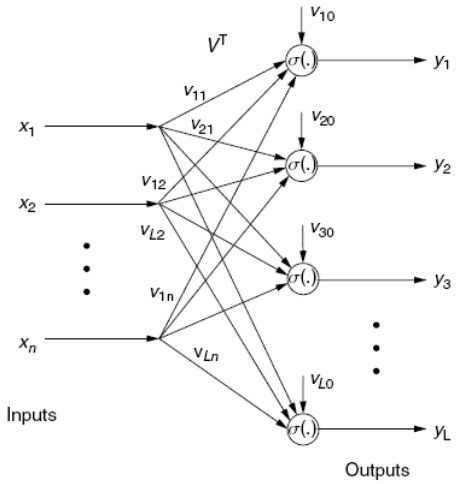
شکل(۱-۲) انواع توابع محرکه مورد استفاده در شبکه‌های عصبی

۱-۲ ساختار شبکه

دو یا چند نرون می‌توانند در یک لایه از شبکه با هم ترکیب شوند. همچنین یک شبکه ممکن است شامل یک یا چندین لایه باشد.

۱-۶-۱ پرسپترون[1] تک لایه

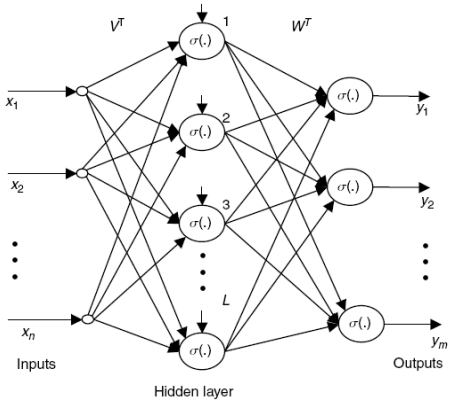
حال یک شبکه تک لایه با چند نرون درلایه مخفی را درنظر بگیرید. تعداد نرون‌های لایه آخر برابر L می‌باشد. $y(k)$ بیان می‌کند که چطور خروجی از سیگنال‌های ورودی و وزن‌های شبکه تشکیل شده است.



شکل(۱-۳) مدل شبکه عصبی تک لایه

۱-۶-۱ پرسپترون چند لایه

یک شبکه عصبی دو لایه که در لایه اول L نرون و در لایه دوم m نرون وجود دارد ، در این قسمت در نظر گرفته شده است. وزن‌های ارتباطی بین ورودی‌ها و نرون‌های لایه اول با V و وزن‌های ارتباطی بین نرون‌های لایه اول و لایه دوم با W نمایش داده شده‌اند.



شکل(۱-۴) شبکه عصبی دولایه

[1] Perceptron

$$y_i = \sigma(\sum_{l=1}^{L} W_{il}\sigma(\sum_{j=1}^{n}(V_{lj}x_j + V_{l0}) + W_{i0}))) \qquad i = 1,2,....m$$

خروجی لایه اول $Z_l = \sigma(\sum_{j=1}^{n} V_{lj}x_j + V_{l0}) \qquad l = 1,2,....L$ (۱-۱)

خروجی لایه دوم $y_i = \sigma(\sum_{l=1}^{L} W_{il}Z_l + W_{i0}) \qquad i = 1,2,....m$

با ترکیب ماتریس‌های درونی W_0, W, V_0, V می‌توان خروجی شبکه چند لایه را به صورت زیر نمایش داد:

$$y = \sigma(W^T\sigma(V^T(x))) \qquad (۲-۱)$$

اگر تابع محرک لایه آخر را به صورت خطی تعریف کنیم خروجی شبکه برابر خواهد بود با:

$$y = W^T\sigma(V^T x(k)) \qquad (۳-۱)$$

در بسیاری از شبکه‌های عصبی، اتصالات بین‌نرونی به‌گونه‌ای است که نرون‌های لایه‌های میانی، ورودی خود را از تمام نرون‌های لایه پائینی خود (به طور معمول لایه نرون‌های ورودی) دریافت می‌کنند. بدین ترتیب در یک شبکه عصبی، سیگنال‌ها به تدریج از یک لایه نرونی به لایه‌های بالاتر حرکت می‌کنند و در نهایت به لایه آخر و خروجی شبکه می‌رسند. چنین مسیری در اصطلاح فنی پیشخور[1] نامیده می‌شود. ارتباطات بین نرونی به نوعی قدرت یک شبکه عصبی را تعیین می‌کنند. قاعده آن است که ارتباطات بین نرونی را به دو گروه تقسیم‌بندی می‌کنند:

- یک نوع از ارتباطات بین نرونی، به‌گونه‌ای است که باعث جمع شدن سیگنال در نرون بعدی می‌شود.
- گونه دوم ارتباطات بین نرونی باعث تفریق سیگنال در نرون بعدی می‌شود. در اصطلاح محاوره‌ای گروهی از ارتباطات انگیزش ایجاد می‌کنند و گروه دیگر ممانعت به عمل می‌آورند.

نوع دیگری از ارتباط بین نرونی در شبکه‌های عصبی به ارتباط بازخورد[2] معروف است. در این نوع از ارتباطات، خروجی یک لایه نرونی به لایه قبلی (یا به لایه‌ای که چند مرحله پائینتر است) اتصال داده می‌شود. در نرم‌افزارهای پیشرفته شبکه‌های عصبی، کاربر و طراح شبکه عصبی می‌تواند نوع ارتباطات بین نرون‌ها و لایه‌های آنها را تعیین کند. در نخستین مرحله آموزش، مقادیر W و V به‌طور تصادفی انتخاب می‌شوند. زیرا تا این پارامترها مقدار نداشته باشند، شبکه عصبی قابل استفاده نخواهد بود. در حین آموزش دیدن شبکه عصبی (یعنی به تدریج همزمان با افزایش دفعاتی که مقادیر پارامترها برای رسیدن به خروجی مطلوب‌تر، تنظیم می‌شوند) مقدار پارامترها به مقدار حقیقی و نهایی خود نزدیک‌تر می‌شوند.

[1] Feed Forward
[2] Feedback

دو روش برای آموزش شبکه‌های عصبی وجود دارد:

۱.روش نظارت شده[۱]

۲.روش نظارت نشده [۲]

در روش نظارت نشده، شبکه عصبی باید بدون کمک گرفتن از جهان خارج، بتواند کار آموزش را انجام دهد. واقعیت آن است که در عمل از روش نظارت شده و یا حداکثر از روش‌های ترکیبی استفاده می‌شود. در چنین فرآیندی، پس از اعمال مجموعه‌های داده‌های آموزشی، پارامترهای شبکه به تدریج به سمت مقادیر نهایی خود همگرا می‌شوند. در مواردی ممکن است که شبکه عصبی اصولاً موفق به فراگیری نشود. بدین معنی که پارامترهای شبکه پس از زمان‌های طولانی به مقدار مشخصی همگرا نشود. چنین مواردی ممکن است بر اثر ناکافی بودن داده‌های آموزشی و یا اصولاً نقص طراحی شبکه ایجاد شوند. حتی مواردی در عمل وجود دارند که شبکه عصبی مشخصی، بر اثر آموزش بیش از حد، در اصطلاح Over trained شود.

۱– ۳ تقریب تابع

از آنجایی که شبکه عصبی تک لایه توانایی تقریب توابع عمومی را ندارند، بنابراین شبکه‌هایی که برای تقریب استفاده می‌شوند، حداقل باید دو لایه داشته باشند تا بتوانیم از آنها در کنترل حلقه بسته استفاده کنیم. از طرفی تابع محرکه باید مشتق پذیر و هموار باشد.

$$f(x): R^n \to R^m$$
$$f(x) = W^T \sigma(V^T x) + \varepsilon \qquad \|\varepsilon\| = \varepsilon_N \tag{۴-۱}$$

ε خطای تقریبی توابع با استفاده از شبکه عصبی NN نامیده می‌شود. مقدار ε زمانی که تعداد نرون‌های لایه مخفی افزایش می‌یابد، کاهش می‌یابد. مسأله تعیین تعداد نرون‌ها که بتواند تقریب کافی و خوبی را بزنند تا به امروز حل نشده است.

۱– ۴ آموزش شبکه عصبی تک لایه–گرادیان کاهشی

شبکه عصبی چند لایه پیشخور شامل لایه‌هایی از نرون‌ها هستند که وزن‌ها، خروجی‌های یک لایه را به ورودی لایه بعدی اتصال می‌دهند و در شناسایی و کنترل سیستم ها بسیار مرسوم هستند. یک شبکه عصبی چند لایه پیشخور را می‌توان یک نگاشت از فضای برداری ورودی $X_{NN} \subset R^n$ به فضای برداری خروجی $Y \subset R^m$ در نظر گرفت. وزن‌های شبکه عصبی برای تقریب توابع نیاز به روزرسانی شدن[۳] دارند. بنابراین دراین قسمت تنظیم وزن‌های شبکه عصبی تک لایه انجام شده است.

---

[۱] Supervised

[۲] Unsupervised

[3] Update rules

$$y = \sigma(V^T x)$$
$$x = [x_1, x_2, ...x_n]^T \in R^{n+1} \qquad (۵-۱)$$
$$y \in R^L$$

که در آن، V ماتریس وزن‌ها می‌باشد. الگوریتم‌های آموزشی زیادی در حال حاضر برای آموزش شبکه استفاده می‌شود. یکی از این روش‌ها روش BP می‌باشد که برای راحتی در فرمول‌بندی آموزش شبکه، از روش‌های ماتریسی استفاده می‌شود. از آنجایی‌که آموزش NN معمولاً توسط کامپیوترهای دیجیتالی انجام می‌شود، راحت‌ترین فرم برای به روزرسانی وزن‌ها به صورت گسسته است. اندیس تکرار با k نمایش داده می‌شود. توجه شود که نباید k را با زمان اشتباه گرفت. $V_{lj}(K)$ وزن‌های شبکه در تکرار K می‌باشد.

$$y_l(k) = \sigma(\sum_{j=1}^{n} V_{lj}(k) x_j + V_{l0}(k)) \qquad l = 1,2,....L \qquad (۶-۱)$$

در طی آموزش شبکه عصبی، ورودی X_j ثابت باقی می‌ماند. الگوریتم به روز رسانی عمومی وزن‌ها با استفاده از معادله به روز رسانی برگشتی زیر بیان می‌شود:

$$V_{lj}(k+1) = V_{lj}(k) - \eta \frac{\partial E(K)}{\partial V_{lj}(k)} \qquad (۷-۱)$$

$E(k)$ تابع هزینه می‌باشد که با توجه به کاربرد انتخاب می‌شود. وزنهای V_{lj} در هر دوره تکرار k به روز رسانی می‌شود.، به گونه‌ای که تابع هزینه $E(k)$ کاهش یابد. سرعت یادگیری η کوچکتر از ۱ در نظر گرفته می‌شود. الگوریتم گرادیان کاهشی مقدار تابع هزینه را کاهش می‌دهد.

$$e_l(k) = Y_l - y_l(k)$$
$$MMSE: E(k) = \frac{1}{2}\sum_{l=1}^{L}(e_l^{\,2}(k)) = \frac{1}{2}\sum_{l=1}^{L}(Y_L(k) - y_l(k))^2 \qquad (۸-۱)$$

توجه شود که ورودی x و خروجی دلخواه y تابعی از عدد تکرار k نمی‌باشد. برای حل باید از تابع هزینه نسبت به وزن‌ها مشتق بگیریم:

$$\frac{\partial E(k)}{\partial V_{lj}(k)} = -e_l(k)\sigma'(\sum_{j=1}^{n} V_{lj}(k) X_j) X_j \qquad (۹-۱)$$

σ' نشان دهنده مشتق از تابع محرکه می‌باشد. بنابراین الگوریتم گرادیان کاهشی برای به روز رسانی وزن‌ها به صورت زیر خواهد بود:

$$V_{lj}(k+1) = V_{lj}(k) + \eta.e_l(k).\sigma'(\sum_{j=1}^{n} V_{lj}(k) X_j).X_j \qquad (۱۰-۱)$$

برای توابع محرکه خطی، الگوریتم تنظیم به صورت زیر خواهد شد:

$$V_{lj}(k+1) = V_{lj}(k) + \eta e_l(k).X_j \qquad (۱۱-۱)$$

۱– ۵ تنظیم وزن ها با استفاده از الگوریتم پس انتشار خطا– آموزش شبکه‌های چند لایه

از آنجایی‌که شبکه تک لایه توانایی تقریب توابع را ندارد، بنابراین باید از شبکه‌های چندلایه استفاده کنیم. سیگنال‌های ورودی لایه به لایه در جهت مستقیم در شبکه انتشار پیدا می‌کنند. شبکه به‌صورت نظارت شده[1] با استفاده از الگوریتم BP آموزش داده می‌شود. این الگوریتم یادگیری بر اساس قانون تصحیح خطا می‌باشد. به این صورت که سیگنال خطا در جهت عکس در شبکه انتشار پیدا می‌کند و وزن‌های شبکه براساس خطا طوری تنظیم می‌شوند که نتیجه مطلوب به‌دست آمده و خطا به سمت صفر میل کند.

-۱)

$$y_i = \sigma(\sum_{l=1}^{L} W_{il}\sigma(\sum_{j=1}^{n}(V_{lj}x_j + V_{l0})) + W_{i0}) \qquad i = 1,2,....m \qquad (۱۲$$

$$\begin{cases} W_{il}(k+1) = W_{il}(k) - \eta\dfrac{\partial E}{\partial W_{il}(k)} \\ V_{lj}(k+1) = V_{lj}(k) - \eta\dfrac{\partial E}{\partial V_{lj}(k)} \end{cases} \qquad (۱۳\text{-}۱)$$

حداقل میانگین مربعات به عنوان تابع خطا به صورت زیر تعریف می‌شود:

$$\begin{cases} e_i(k) = Y_i(k) - y_i(k) \\ E(K) = \dfrac{1}{2}e^T(k)e(k) = \dfrac{1}{2}\sum_{i=1}^{m} e_l{}^2(k) \end{cases} \qquad (۱۴\text{-}۱)$$

در الگوریتم BP معمولاً از شبکه پیشخور چند لایه استفاده می‌شود ، در این شبکه ها نیز از تابع log-sigmoid بهره می‌برند. تابع محرک logsig خروجی بین ۰و۱ را در ازای ورودی‌های منفی و مثبت ایجاد می‌کند. اگر در لایه آخر شبکه‌های چندلایه از تابع سیگموئید استفاده شود، خروجی‌های شبکه بین محدوده کوچکی قرار می‌گیرد، ولی اگر از تابع محرک خطی استفاده شود، خروجی‌های شبکه هر مقداری را می‌توانند بگیرند.

۱– ۶ ایجاد شبکه عصبی

اولین مرحله از آموزش شبکه پیشخور، ایجاد اعضای شبکه می‌باشد. تابع newff یک شبکه پیشخور را ایجاد می‌کند، که شامل چهار آرگومان ورودی بوده و خروجی آن به صورت object است، که همان شبکه مورد نظر می‌باشد. اولین آرگومان ورودی، ماتریس ورودی است که یک ماتریس با ابعاد n*L است (شکل (۱-۴) را ببینید) که مقادیر مینیمم و ماکزیمم برای هر عنصر ورودی را شامل می‌شود. آرگومان ورودی دوم، یک بردار شامل تعداد نرون‌های هر لایه شبکه است. سومین آرگومان ورودی آرایه‌ای است که شامل نام توابع محرکه‌ای می‌باشد که در هر لایه استفاده می‌شود. آخرین ورودی تابع آموزش مورد استفاده است.

```
>>net=newff([-1 2;0 5],[3 1],{'tansig','purelin'},'traingd')
```

[1] Supervised

این عبارت شبکه‌ای شامل دولایه که بردار ورودی آن دو عنصر دارد، می‌شود. اولین مقدار برای عنصر بردار ورودی بین ۱- و ۲ و دومین عنصر بردار ورودی بین ۰و ۵ است. در لایه اول ۳ نرون با تابع محرک tansig و در لایه دوم ۱ نرون با تابع محرک purelin وجود دارد.

۱-۶-۱ مقدار دهی اولیه وزن‌های شبکه

دستور newff یک شبکه با مقادیر وزن و بایاس تصادفی ایجاد می‌کند. ولی قبل از آموزش شبکه عصبی پیشخور، وزن بایاس‌ها باید مقدار دهی اولیه شوند. برای اینکه بتوانیم مقداردهی را دوباره انجام دهیم از init استفاده می‌کنیم. این تابع به ما امکان مقداردهی اولیه وزن‌ها و بایاس‌ها را می‌دهد.

```
>>net=init(net)
```

۱-۶-۲ شبیه سازی شبکه

تابع sim شبکه را شبیه‌سازی می‌کند. این تابع با گرفتن ورودی شبکه و خود شبکه (net)، خروجی شبکه را ایجاد می‌کند.

```
>>a = sim(net,p)
```

۱-۶-۳ آموزش شبکه

هنگامی‌که وزن‌ها و بایاس‌ها مقدار اولیه گرفتند، شبکه برای آموزش آماده می‌شود. شبکه می‌تواند برای تخمین یک تابع (رگرسیون غیرخطی) و عمل دسته بندی آموزش داده شود. در حین آموزش وزن‌ها و بایاس‌های شبکه بارها خود را به گونه‌ای تطبیق می‌دهد که تابع هزینه شبکه مینیمم شود. دو روش جهت اجرای این الگوریتم وجود دارد:

- **روش نمونه‌ای**: در این روش گرادیان و وزن‌ها هر بار بعد از هر ورودی که به شبکه اعمال می‌شود، محاسبه و تنظیم می‌شوند.
- **روش دسته‌ای**: همه ورودی ها به شبکه قبل از تنظیم وزن‌ها اعمال می‌شوند. وزن‌ها و بایاس‌های شبکه بعد از آموزش کاملی که به شبکه داده می‌شود ، تنظیم می‌شوند.

۱- ۷ گرادیان دسته‌ای

پنج پارامتر اصلی تابع آموزش دسته‌ای عبارتند از:

۱- epochs: تعداد تکرارهای آموزش شبکه است.

۲- show: با این پارامتر وضعیت آموزش در دو مرحله نمایش داده می‌شود ، در صورتی که NaN باشد وضعیت آموزش نشان داده نمی‌شود.

۳- time: زمان آموزش است.

۴- goal: مقدار نهایی تابع است.

۵- lr: سرعت یادگیری شبکه است. اگر سرعت یادگیری شبکه بسیار بزرگ باشد الگوریتم به صورت ناپایدار در می‌آید و اگر سرعت یادگیری خیلی کوچک باشد، در آن صورت زمان زیادی طول می‌کشد تا الگوریتم به جواب برسد.

آموزش شبکه زمانی متوقف می‌شود که:

- تعداد تکرارها از مقدار پارامتر epochs بیشتر شود.
- تابع هزینه کمتر از پارامتر goal شود.
- زمان آموزش بیشتر از time شود.

توجه کنید که برای آموزش دسته‌ای همه بردارهای ورودی در یک ماتریس قرار می‌گیرند. به عنوان مثال فرض کنید، ماتریس ورودی P و بردار هدف t به صورت زیر باشد:

```
p = [-1 -1 2 2;0 5 0 5];
t = [-1 -1 1 1];
```

حال باید یک شبکه ایجاد کنیم:

```
Net=newff(MinMax(p),[3,1],{'tansig','purelin'},'traingd');
```

در این قسمت می‌خواهیم، پارامترهای آموزش شبکه را تعریف کنیم:

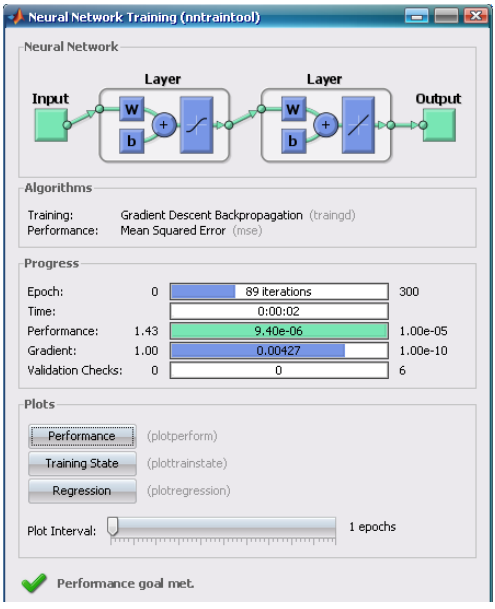
```
Net.trainParam.Show=50;
Net.trainParam.Lr=0.05;
Net.trainParam.epochs=300;
Net.trainParam.goal=1e-5;
```

حال شبکه برای آموزش آماده است:

```
[Net,tr]=train(Net,p,t);
```

بعد از اینکه شبکه آموزش داده شد، می‌توان شبیه سازی را آغاز کرد.

```
a = sim(Net,p)
a =

   -0.9967   -1.0019    0.9959    1.0026
```

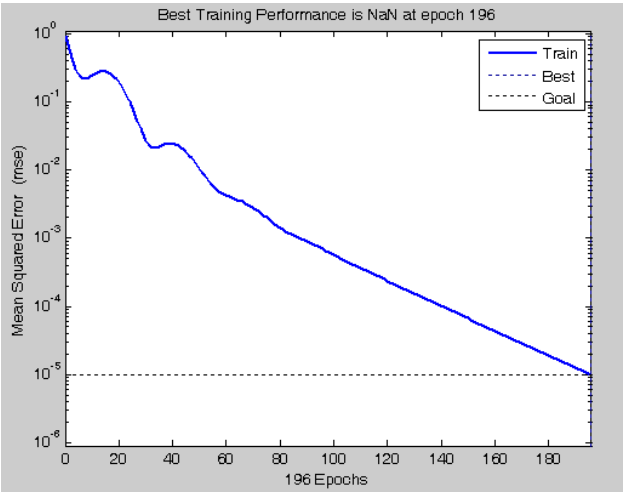


شکل(۱-۵) آموزش شبکه عصبی

۱– ۸ گرادیان دسته‌ای با اعمال ممنتوم

علاوه بر traingd، الگوریتم دیگری برای شبکه‌های پیشخور مورد استفاده قرار می‌گیرد که باعث همگرایی سریعتر شبکه می‌شود. این الگوریتم traingdm نام دارد که با مومنتوم به کار می‌رود. پارامتر مومنتم مثل فیلتری می‌ماند که به شبکه این امکان را می‌دهد که از جزئیات کوچک در سطح خطا چشم‌پوشی کند. بدون مومنتم شبکه ممکن است در یک مینیمم local همگرا شود. ولی با مومنتم شبکه قادر است که به سمت مینیمم Global حرکت کند. ثابت مومنتم هر مقداری بین ۰ و ۱ را می‌تواند اختیار کند.

```
p = [-1 -1 2 2;0 5 0 5];
t = [-1 -1 1 1];
Net=newff(MinMax(p),[3,1],{'tansig','purelin'},'traingdm');
Net.trainParam.Show=50;
Net.trainParam.Lr=0.05;
Net.trainParam.epochs=300;
Net.trainParam.goal=1e-5;
[Net,tr]=train(Net,p,t);
a = sim(Net,p)
a =

    -1.0039   -0.9968    1.0015    0.9966
```



شکل(۱–۶) تابع عملکرد شبکه عصبی

زمانی که شبکه آموزش داده شد، شبکه الگوهای آموزش داده شده را به خاطر می‌سپارد، اما قابلیت تعمیم به حالت جدید را ندارد.

۱– ۹ تابع هزینه اصلاح شده

تابع هزینه‌ای که برای شبکه‌های پیشخور مورد استفاده قرار می‌گیرد برابر با مجموع مجذورهای خطاهای شبکه است.

$$F = mse = \frac{1}{N}\sum_{i=1}^{N} e_i^2 \qquad (۱۵-۱)$$

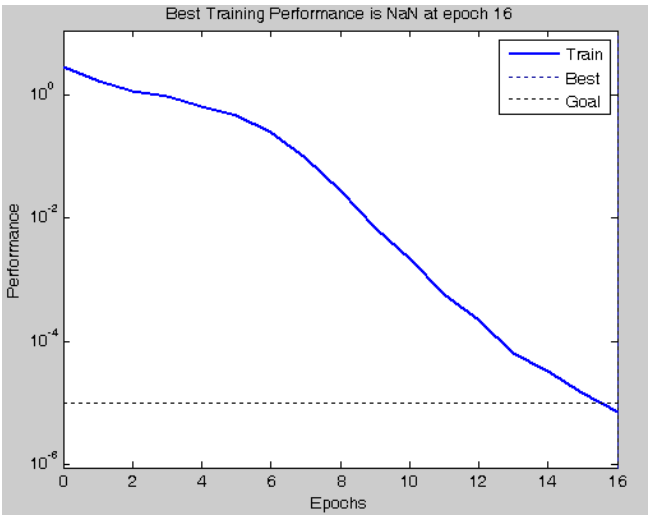
بهبود تنظیم شبکه زمانی میسر است که تابع هزینه را با عبارتی تصحیح کنیم که شامل مجموع مجذور وزن‌ها و بایاس‌های شبکه می‌شود. در روابط زیر γ سرعت عملکرد می‌باشد.

$$msereg = \gamma.mse + (1-\gamma)msw$$

$$msw = \frac{1}{n}\sum_{j=1}^{n} w_j^2 \qquad (۱۶-۱)$$

استفاده از این تابع عملکرد باعث می‌شود که شبکه، وزن‌ها و بایاس‌های کوچکتری داشته و شبکه را قادر می‌سازد که آسانتر جواب دهد.

```
p = [-1 -1 2 2;0 5 0 5];
t = [-1 -1 1 1];
net=newff(minmax(p),[3,1],{'tansig','purelin'},'trainbfg');
net.performFcn = 'msereg';
net.performParam.ratio = 0.5;
net.trainParam.show = 5;
net.trainParam.epochs = 300;
net.trainParam.goal = 1e-5;
[net,tr]=train(net,p,t);
```

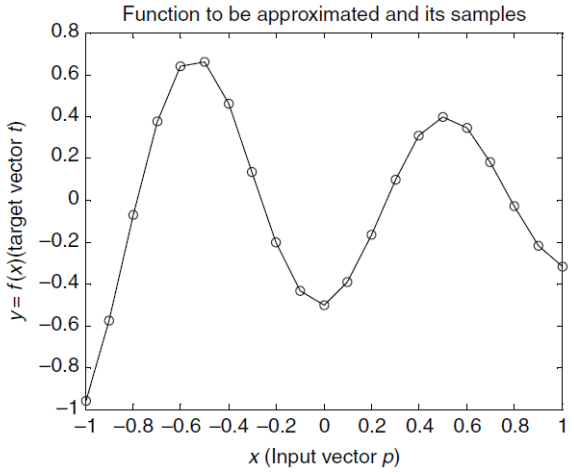


شکل(۱–۷) تابع عملکرد شبکه عصبی

تعیین بهینه پارامترهای تنظیم به صورت اتوماتیک با الگوریتم trainbr امکان‌پذیر است. الگوریتم trainbr زمانی‌که ورودی و اهداف شبکه بین [-1,1] قرار داشته باشند مورد استفاده قرار می‌گیرد.

```
p = [-1:.05:1];
t = sin(2*pi*p)+0.1*randn(size(p));
net=newff(minmax(p),[20,1],{'tansig','purelin'},'trainbr');
net.trainParam.show = 10;
net.trainParam.epochs = 50;
randn('seed',192736547);
net = init(net);
[net,tr]=train(net,p,t);
```

مثال: می‌خواهیم یک شبکه دولایه را طوری طراحی کنیم که تابع نشان داده شده در شکل (۱-۸) را تقریب بزند.



شکل (۱-۸) تابع y=f(x) که توسط شبکه عصبی دولایه تقریب زده می‌شود

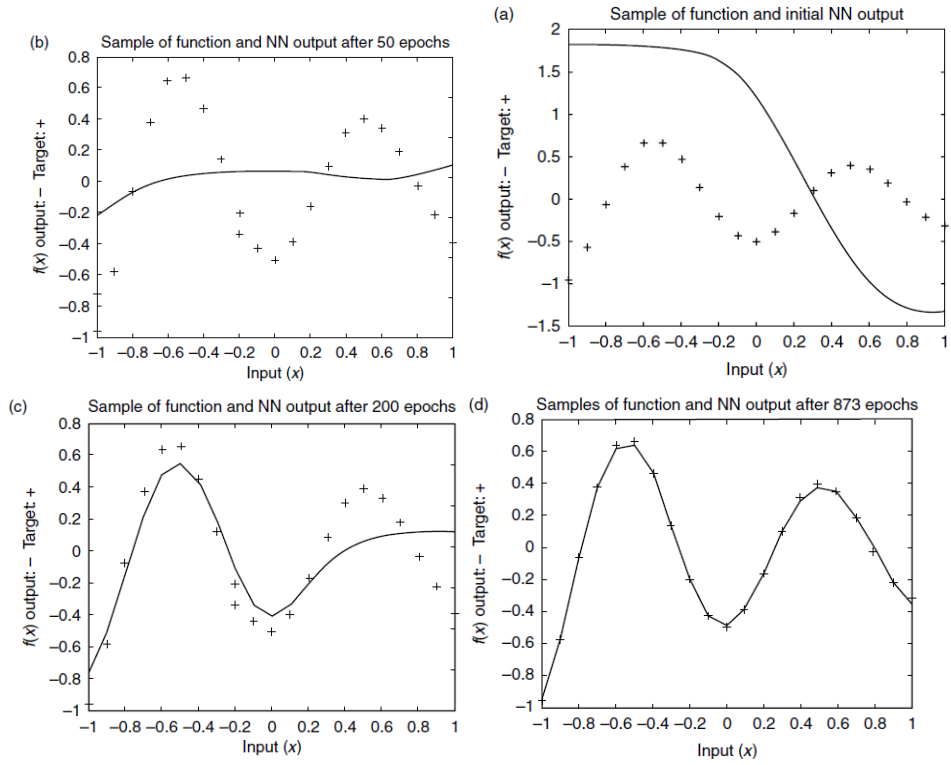
```
p=-1:0.1:1;
t=[-0.960 -0.577 -0.073 0.377 0.641 0.660 0.461 0.134...
    -0.201  -0.434  -0.500  -0.393  -0.165  0.099  0.307  0.396
0.345...
    0.182 -0.031 -0.219 -0.320];
```

تابع تبدیل لایه مخفی تانژانت هایپربولیک و تابع تبدیل لایه خروجی خطی است. لایه مخفی پنج نرون دارد.

```
net = newff(minmax(p),[5,1],{'tansig','purelin'},'trainlm');
net.trainParam.show = 10;
net.trainParam.lr = 0.01;
net.trainParam.epochs = 50;
net.trainParam.goal = 0.005;
net = train(net,p,t);
y0 = sim(net,p);
```

```
figure;
plot(p,t,'o',p,y0,'-')
title('Samples of function and NN output after 50 epochs');
xlabel('Input (x)');
ylabel('f(x) Output: - Target: +');
```

که نتیجه را در شکل‌های زیر مشاهده می‌کنید:

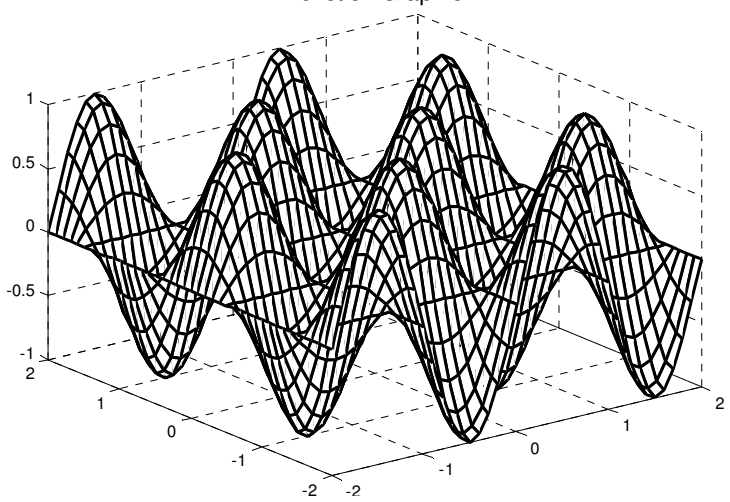


شکل(۱-۹) تقریب تابع y=f(x) توسط شبکه عصبی دولایه با a- مقادیر اولیه شبکه، b-۵۰ سیکل، c- ۲۰۰ سیکل، d-۸۷۳ سیکل

مثال: با استفاده از شبکه‌های عصبی چند لایه (MLP) همراه با الگوریتم BP، تابع دو متغیره غیرخطی زیر را تقریب بزنید.

$$f(x,y)=\sin(\pi x).\cos(\pi y) \qquad -2\le x\le 2, -2\le y\le 2$$

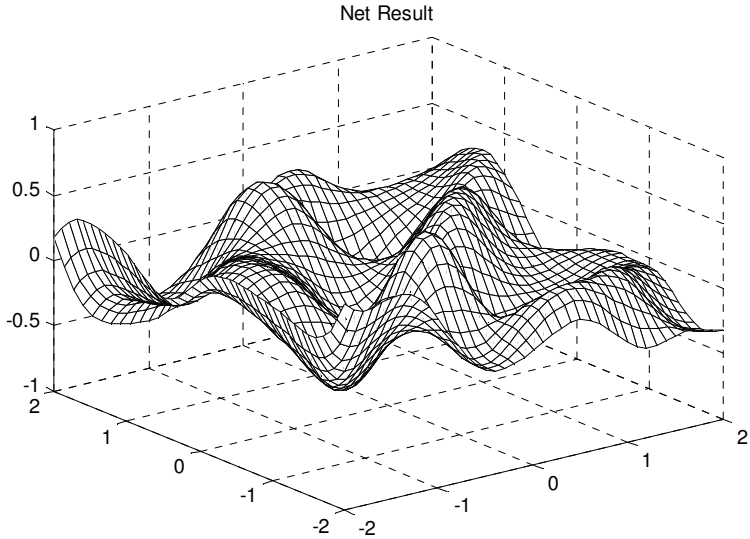
```
figure(1);
[X,Y] = meshgrid(-2:0.1:2);
z = sin(pi*X).*cos(pi*Y);
mesh(X,Y,z);
title('Function Graphics');
```



شکل (۱-۱۰) تابع دو متغیره $f(x,y)=\sin(\pi x).\cos(\pi y)$

```
% Generate Input & Target data. Totally 2000 groups.
for i=1:2000
    P(:,i) = 4*(rand(2,1)-.5);
    T(:,i)=sin(pi*P(2*i-1))*cos(pi*P(2*i));
end
% BP training (1).
% Here a two-layer feed-forward network is created. The
network's
% input ranges from [-2 to 2]. The first layer has twenty
TANSIG
% neurons, the second layer has one PURELIN neuron. The
TRAINGD (Basic gradient descent)
net1=newff(minmax(P),[20,1],{'tansig','purelin'},'traingd');
net1.inputWeights{:,:}.initFcn = 'rands';
net1.layerWeights{:,:}.initFcn= 'rands';
net1.trainParam.show = 50;
net1.trainParam.epochs = 1000;
net1.trainParam.goal = 1e-5;
[net1,tr]=train(net1,P,T);

a=zeros(41,41);
[X,Y] = meshgrid(-2:0.1:2);
for i = 1 : 1681
    a(i) = sim(net1,[X(i);Y(i)]);
end
mesh(X,Y,a);
```

```
title('Net1 result');
```



شکل(۱-۱۱) تقریب تابع دو متغیره

```
% BP training (2).
%    Now    we    use    TRAINGDM    (Gradient    descent    with
momentum)training function.
%This time we introduce validation set.
for i=1:2000
    P(:,i) = 4*(rand(2,1)-.5);
    T(:,i)=sin(pi*P(2*i-1))*cos(pi*P(2*i));
end
for i=1:50
    P1(:,i) = 4*(rand(2,1)-.5);
    T1(:,i)=sin(pi*P1(2*i-1))*cos(pi*P1(2*i));
end
val.P=P1;
val.T=T1;
net2=newff(minmax(P),[10,1],{'tansig','purelin'},'traingdm');
net2.inputWeights{:,:}.initFcn = 'rands';

net2.layerWeights{:,:}.initFcn= 'rands';
net2.trainParam.show = 50;
net2.trainParam.epochs = 1000;
net2.trainParam.goal = 1e-5;
[net2,tr]=train(net2,P,T,[],[],val);
b=zeros(41,41);
[X,Y] = meshgrid(-2:0.1:2);
```

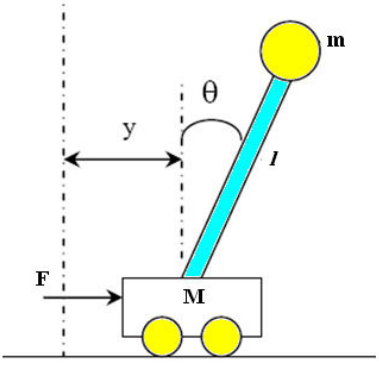
```
for i = 1 : 1681
    b(i) = sim(net2,[X(i);Y(i)]);
end
mesh(X,Y,b);
title('Net2 result');
```

همانطور که از شکل‌های بالا واضح است، نتیجه به‌دست آمده مناسب نمی‌باشد. این شبکه را باید با تعداد نرون های متفاوت و توابع تبدیل مختلف بررسی کنیم تا به نتیجه مطلوب برسیم.

ب–کنترل پاندول معکوس با استفاده از شبکه عصبی

۱– ۱۰ طراحی کنترلر برای مدل خطی و غیرخطی پاندول معکوس

معادلات دینامیکی پاندول معکوس با نوشتن معادلات نیوتن به صورت زیر به‌دست می‌آید:



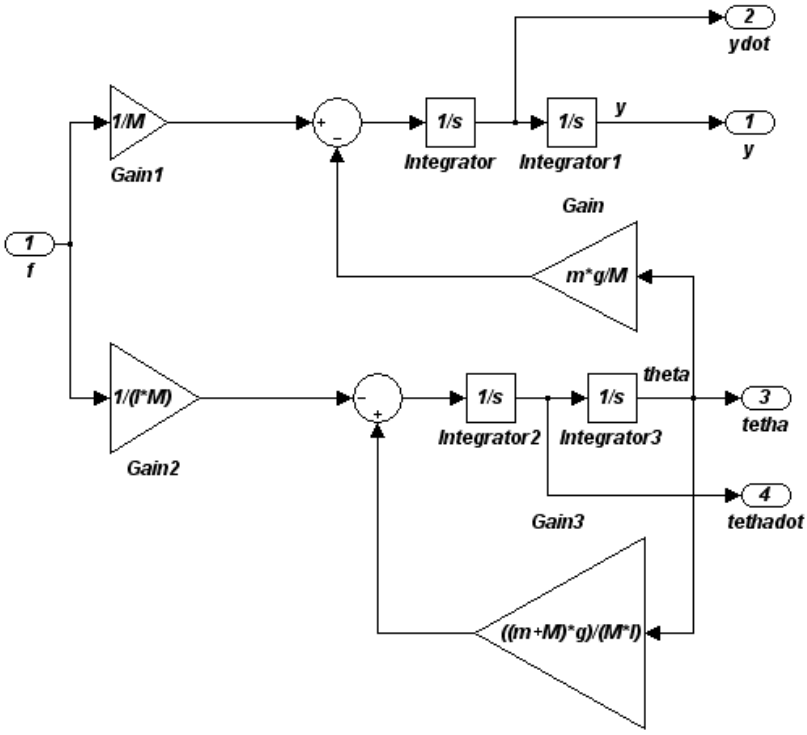
شکل(۱–۱۲) مدل پاندول معکوس

$$\begin{cases}(M+m)\ddot{y}+ml\ddot{\theta}Cos\theta-ml\dot{\theta}^2Sin\theta=F\\ ml^2\ddot{\theta}-mglSin\theta+ml\ddot{y}Cos\theta-\dot{y}\dot{\theta}ml\sin\theta=0\end{cases} \quad (۱۷-۱)$$

در صورتی‌که فرض کنیم θ بسیار کوچک باقی می‌ماند، می‌توانیم معادلات غیرخطی را حول $\theta=0$ خطی فرض کنیم. بنابراین معادلات خطی شده به صورت زیر خواهد بود :

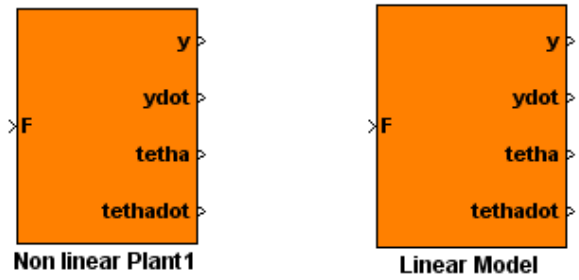
$$\begin{cases}(M+m)\ddot{y}+ml\ddot{\theta}=F\\ ml^2\ddot{\theta}-mgl\theta+ml\ddot{y}=0\end{cases}\Rightarrow\begin{cases}\ddot{y}=\dfrac{f}{M}-\dfrac{mg}{M}\theta\\ \ddot{\theta}=\dfrac{-f}{Ml}+\dfrac{M+m}{Ml}g\theta\end{cases} \quad (۱۸-۱)$$

شبیه‌سازی مدل خطی پاندول معکوس با فرض M=2kg، m=0.11kg، g=9.8m/s^2 و l=0.4m، در محیط سیمولینک مطابق شکل(۱–۱۳) است.

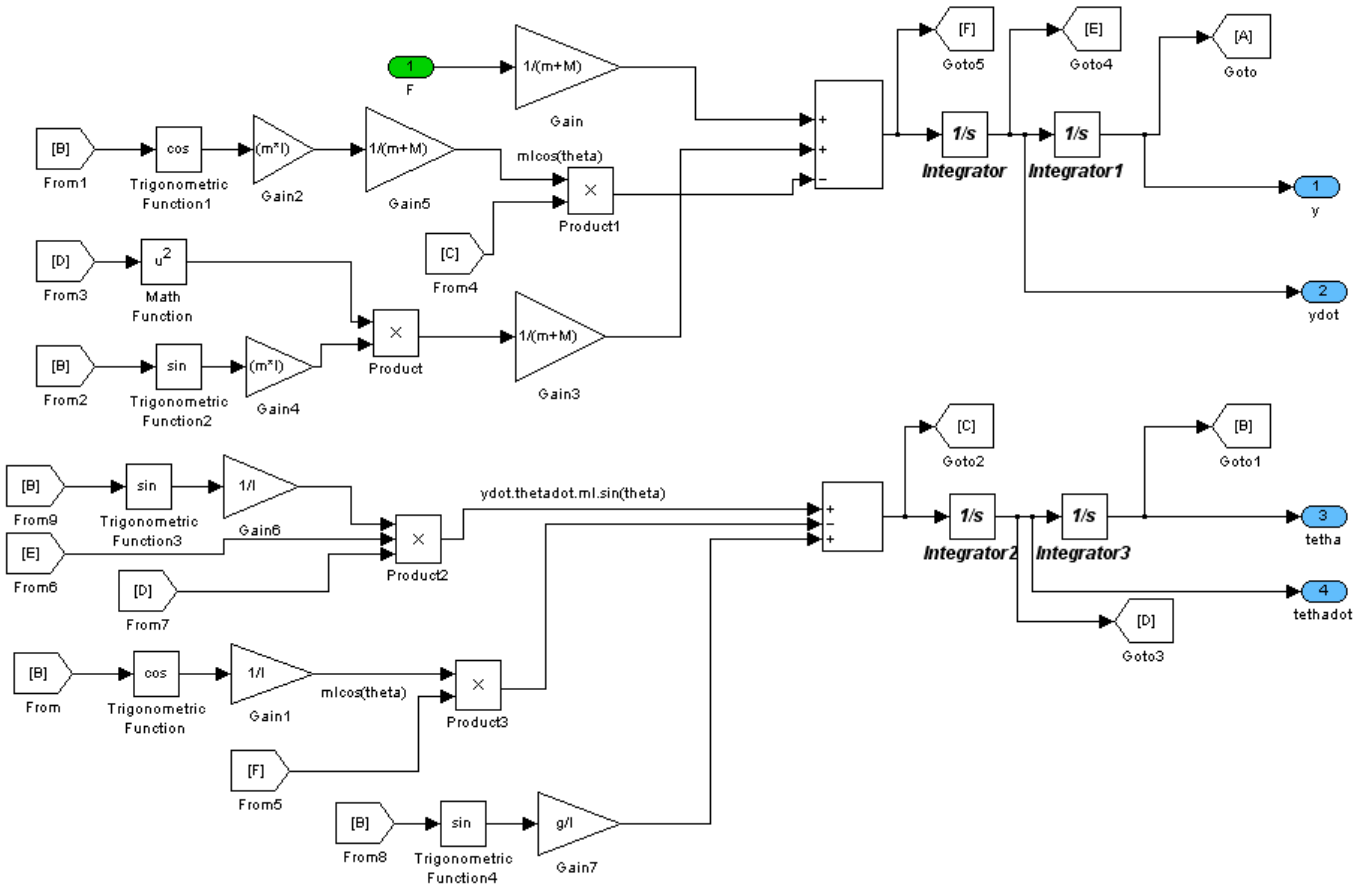


شکل(۱-۱۳) مدل خطی پاندول معکوس در سیمولینک

شبیه‌سازی مدل غیرخطی پاندول معکوس در محیط سیمولینک در شکل (۱-۱۵) آورده شده است. کلیهٔ بلوک-ها در دو مدل خطی و غیرخطی را انتخاب کرده و با کلیک راست کردن بر روی بلوک‌ها، گزینه create subsystem را انتخاب می‌کنیم، سپس مدل subsystemهای خطی و غیرخطی زیر را ایجاد می‌کنیم.

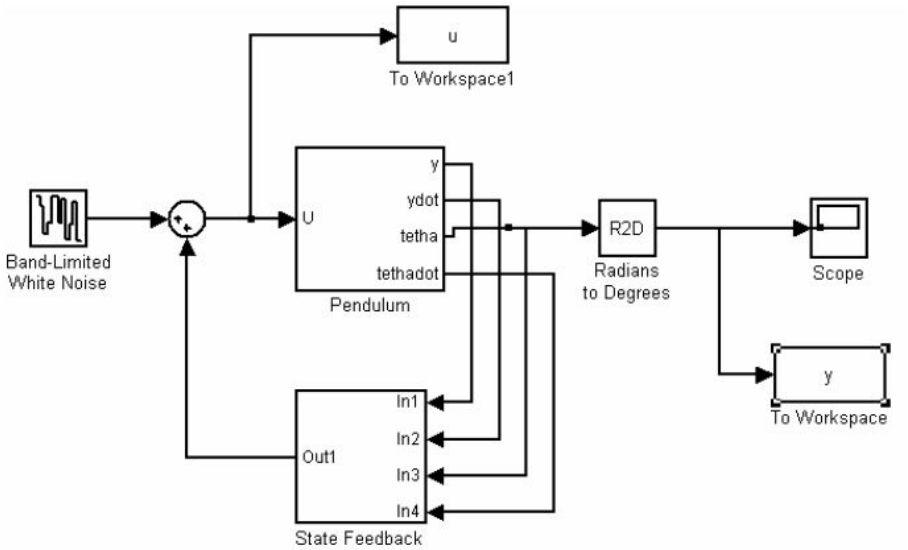


شکل(۱-۱۴) مدل‌های خطی و غیرخطی پاندول معکوس

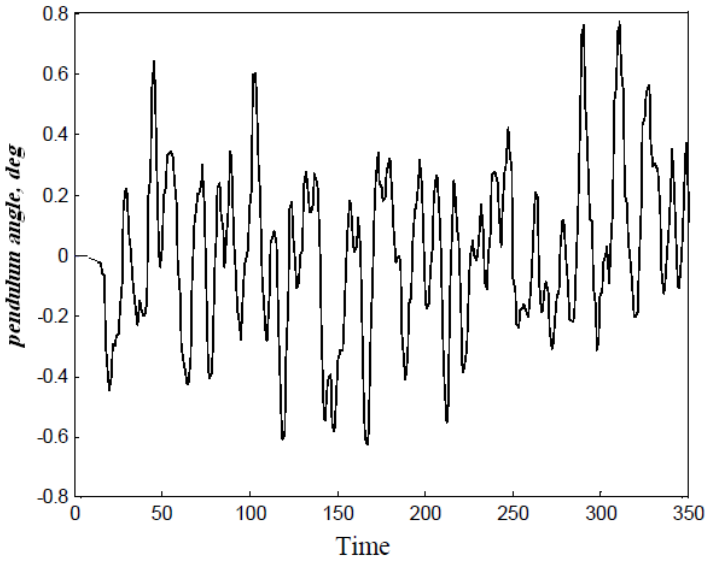


شکل(۱-۱۵) مدل غیرخطی پاندول معکوس در سیمولینک

در بخش ۴ پروژه ۵ به تفصیل ملاحظه خواهید نمود که شبیه‌سازی کنترلر با فیدبک کلیه متغیرهای حالت برای مدل پاندول معکوس خطی شده به صورت زیر است:



شکل(۱-۱۶) طراحی کنترلر با فیدبک کلیه متغیرهای حالت برای پاندول معکوس



شکل(۱-۱۷) زاویه پاندول معکوس با کنترلر فیدبک کلیه متغیرهای حالت

طراحی کنترلر برای پاندول معکوس با معادلات غیرخطی بسیار مشکل است. طراحی کنترلرهای خطی همچون کنترلر PID و فیدبک متغیرهای حالت برای کنترل سیستم غیرخطی ناموفق است. برای کنترل این سیستم

غیرخطی از روش Feedback Linearization یا خطی‌سازی به کمک فیدبک استفاده می‌کنیم. این روش کنترلی، غیرخطی‌های موجود در سیستم را حذف می‌کند تا اینکه سیستم حلقه بسته به یک سیستم خطی تبدیل شود.

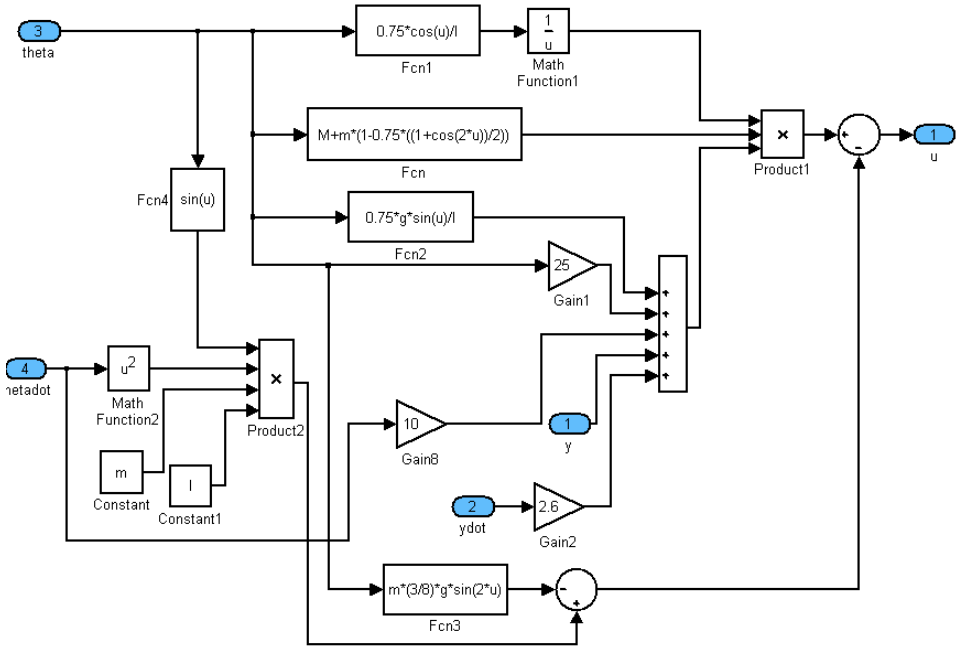
$$h_1 = \frac{3}{4l} g \sin\theta, h_2 = \frac{3}{4l}\cos\theta$$
$$f_1 = m(l.\sin\theta.\dot{\theta}^2 - \frac{3}{8} g.\sin 2\theta) \ , f_2 = M + m(1 - \frac{3}{4}\cos^2\theta) \quad (۱۹-۱)$$

$$u = \frac{f_2}{h_2}\left[h_1 + k_1(\theta - \theta_d) + k_2\theta + c_1(\dot{y} - \dot{y}_d) + c_2\dot{y}\right] - f_1 \quad (۲۰-۱)$$

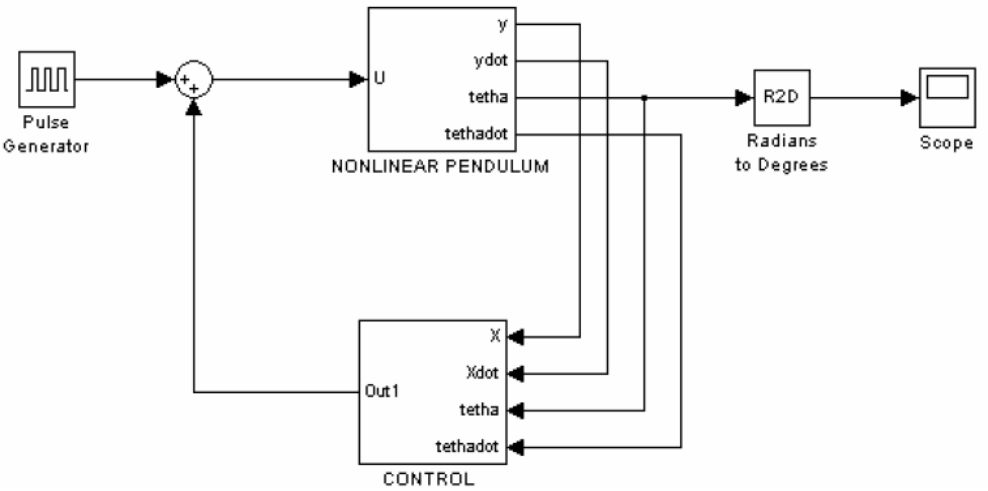
بنابراین نیروی u با معادله (۱-۲۰) پاندول را به حالت پایدار نگه می‌دارد. برای شبیه‌سازی، مقادیر پارامترهای معادله را به‌صورت زیر در نظر گرفتیم:

$$M = 1.2kg, m = 0.11kg, l = 0.4m, \ g = 9.8m/s^2, \ k_1 = 25,$$
$$k_1 = 10, \ C_1 = 1, \ C_2 = 2.6, \ \theta_d = 0rad, y_d = 0$$

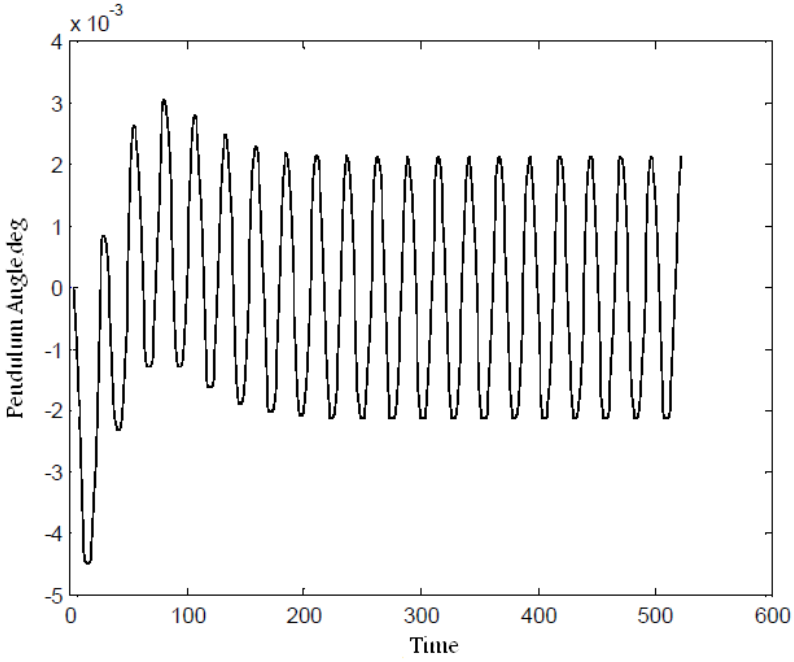
شبیه سازی کنترلر در محیط سیمولینک در شکل (۱-۱۸) آورده شده است:



شکل(۱-۱۸) شبیه سازی کنترلر با استفاده از خطی‌سازی به‌کمک فیدبک



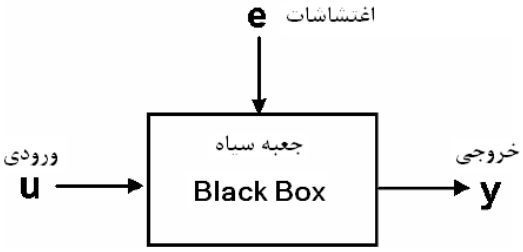
شکل(۱-۱۹) شبیه سازی سیستم کنترلی با استفاده از خطی‌سازی به‌کمک فیدبک



شکل (۱-۲۰) شبیه سازی سیستم غیرخطی با کنترلر خطی‌سازی به‌کمک فیدبک

۱– ۱۱ شناسایی سیستم[1]

در بخش‌های قبل مدل‌های خطی و غیرخطی پاندول معکوس شبیه سازی شدند و دریافتیم که سیستم در حالت حلقه باز ناپایدار است. از طرفی برای شناسایی دقیق سیستم نیاز داریم که سیستم پایدار باشد، به این علت کنترل‌های فیدبک برای مدل خطی و غیرخطی طراحی شدند. شناسایی یک سیستم یعنی ایجاد یک مدل ریاضی از سیستم دینامیکی براساس داده‌های ورودی و خروجی سیستم واقعی. این بدان معنی است که می‌توان از سیستم واقعی، از داده‌های ورودی و خروجی نمونه‌گیری کرده و از این داده‌ها یک مدل ریاضی ایجاد کرد. یک قدم بسیار مهم در طراحی سیستم کنترلی این است که یک مدل ریاضی از سیستم تحت کنترل ایجاد کنیم، زیرا وقتی سیستم واقعی در حال کار است دیگر نیازی به محاسبه دوباره معادلات دینامیکی و پارامترهای مدل سیستم نیست. شکل (۱-۲۱) ورودی‌ها و خروجی‌های یک سیستم را نشان می‌دهد. مدل ریاضی در این نمونه یک جعبه سیاه است که تنها رابطه بین ورودی و خروجی سیستم را توصیف می‌کند.



شکل(۱-۲۱) نمایش سیگنال ورودی، خروجی و اغتشاش

در بخش‌های قبلی دریافتیم که شبکه عصبی قابلیت بسیار خوبی در مدل‌سازی سیستم‌های غیرخطی دارد. قبل از اینکه از شبکه عصبی برای شناسایی سیستم استفاده کنیم، از تکنیک‌های خطی همچون ARX و ARMAX استفاده کرده و آن‌ها را به مدل خطی پاندول معکوس اعمال می‌کنیم.

اساساً شناسایی سیستم با تنظیم پارامترهای مدل انجام می‌شود تا خروجی مدل، مشابه خروجی سیستم واقعی شود. به طور کلی برای شناسایی سیستم سه قدم اصلی وجود دارد:

۱- استخراج داده‌های ورودی و خروجی که از آزمایش روی مدل اصلی به‌دست آمده است. در مدل پاندول معکوس ورودی نیروی f و خروجی زاویه پاندول است.

۲- انتخاب ساختار مدل. به طور مثال ساختار مدل ARX به‌صورت زیر است:

$$Ay(t) = Bu(t) + e(t)$$

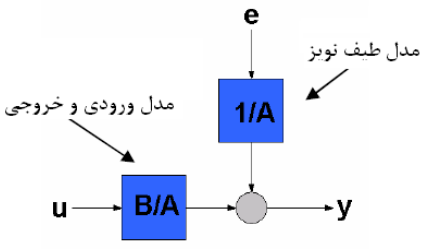
۳- پارامترهای A و B باید طوری تنظیم شوند که خروجی مدل برابر خروجی مدل اصلی شود.

چند نمونه از مدل‌های خطی استاندارد برای شناسایی سیستم در ادامه توضیح داده شده‌اند.

[1] System Identification

۱-۱۱-۱ مدل ARX

رابطهٔ بین ورودی و خروجی با استفاده از تابع تبدیل B/A مدل‌سازی می‌شود. در این مدل فرض می‌شود که طیف نویزی، مدل ورودی، و خروجی خصوصیات دینامیکی یکسانی دارند.

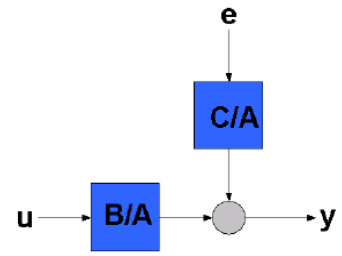


$$y(t)=\frac{B}{A}u(t)+\frac{1}{A}e(t)$$

شکل (۱-۲۲) مدل و معادله ARX

۱-۱۱-۲ مدل ARMAX

این مدل یک پارامتر اضافه به نام C در مدل طیف نویز دارد که دقت بیشتری به ARMAX نسبت به ARX می‌دهد.

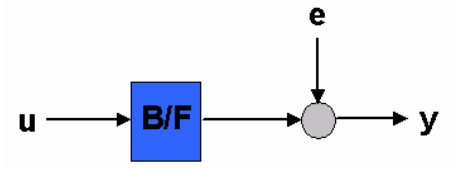


$$y(t)=\frac{B}{A}u(t)+\frac{C}{A}e(t)$$

شکل (۱-۲۳) مدل و معادله ARMAX

۱-۱۱-۳ مدل Output Error (OE)

در این مدل فرض می‌شود که اغتشاشات نویز سفید هستند. رابطه ورودی‌ها و خروجی‌ها در بلوک نمودار B/F تعریف می‌شوند.

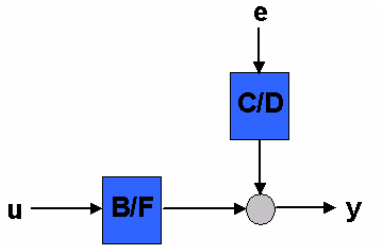


$$y(t)=\frac{B}{F}u(t)+e(t)$$

شکل (۱-۲۴) مدل و معادله Output Error

۱-۱۱-۴ مدل Box Jenkins(BJ)

این مدل تابع تبدیل‌های مختلفی برای رابطه بین ورودی و خروجی و طیف نویزی دارد، بنابراین این مورد، برتری این مدل نسبت به مدل ARMAX است، زیرا در مدلARMAX مخرج ها یکی بودند.

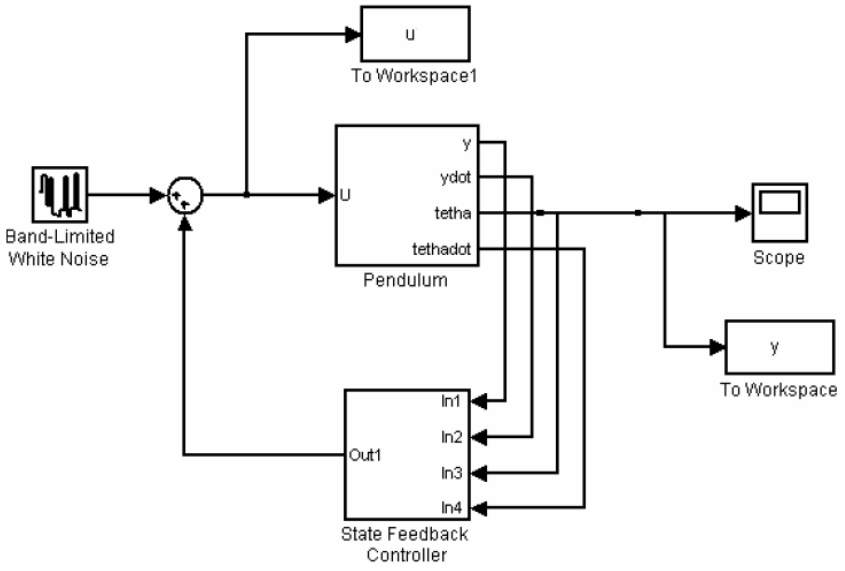


$$y(t)=\frac{B}{F}u(t)+\frac{C}{D}e(t)$$

شکل (۱-۲۵) مدل و معادله Box-Jenkins

۱- ۱۲ شناسایی خطیِ مدل سیستم

به منظور ایجاد مدل ARX،ARMAX،BJ، و OE از پاندول معکوس خطی شده، باید داده‌های ورودی و خروجی سیستم خطی جمع‌آوری شوند. شکل (۱-۲۶) مدل سیمولینک پاندول خطی شده با کنترل فیدبک کلیه متغیرهای حالت را نشان می‌دهد که داده‌ها از این طریق جمع آوری می‌شوند.



شکل (۱-۲۶) استفاده از داده‌های مدل خطی پاندول معکوس با کنترلر فیدبک کلیه متغیرهای حالت، به‌عنوان اهداف شبکه عصبی

ورودی نویز سفید به عنوان سیگنال تحریک استفاده می‌شود. داده‌ها را به دو دسته داده‌های تخمینی و داده‌های ارزیابی تقسیم می‌کنیم. از نیمی از داده‌ها برای ایجاد مدل و از نیمی دیگر برای تست عملکرد مدل استفاده می شوند. حال داده‌های ورودی و خروجی را به دو دسته تقسیم کرده و با استفاده از مدل ARX، سیستم را تخمین می‌زنیم.

```
z1 = [y(1:500) u(1:500)];
z2 = [y(501:1000) u(501:1000)];
nn = [5 3 1];
th = arx(z1,nn);
[yh,fit1] = compare(z2,th);
[num,den] = th2tf(th);
sysarx = tf(num,den);
```

آرگومان دوم مدل ARX به صورت زیر تعریف می‌شود:

nn= [na nb nc];

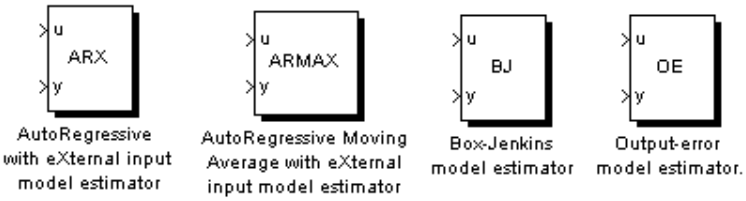
na= تعداد پارامترهایی که در مخرج باید تخمین زده شوند

nb= تعداد پارامترهایی که در صورت باید تخمین زده شوند

nc= تأخیر زمانی در مدل

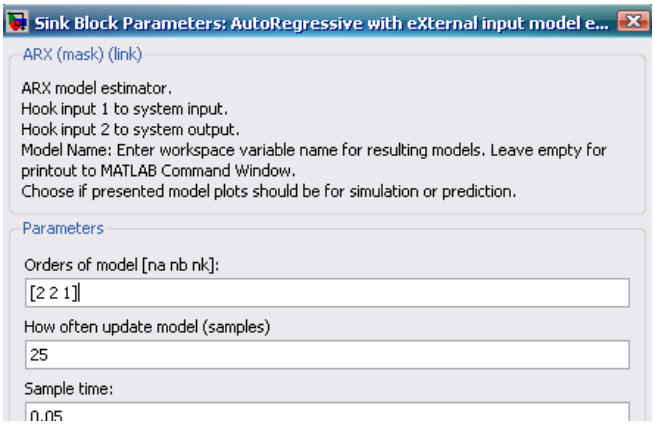
```
Transfer function:
num/den =
                0.00075744 z^4 - 5.0207e-007 z^3 - 0.00075709 z^2
   ------------------------------------------------------------------------
   z^5 - 1.6422 z^4 + 0.33295 z^3 + 0.34816 z^2 + 0.00031091 z - 0.00018177
Noise model:
num/den =
                                      z^5
   ------------------------------------------------------------------------
   z^5 - 1.6422 z^4 + 0.33295 z^3 + 0.34816 z^2 + 0.00031091 z - 0.00018177
```

بلوک‌های شناسایی مدل در کتابخانه سیمولینک و در جعبه ابزار System Identification قرار دارند.

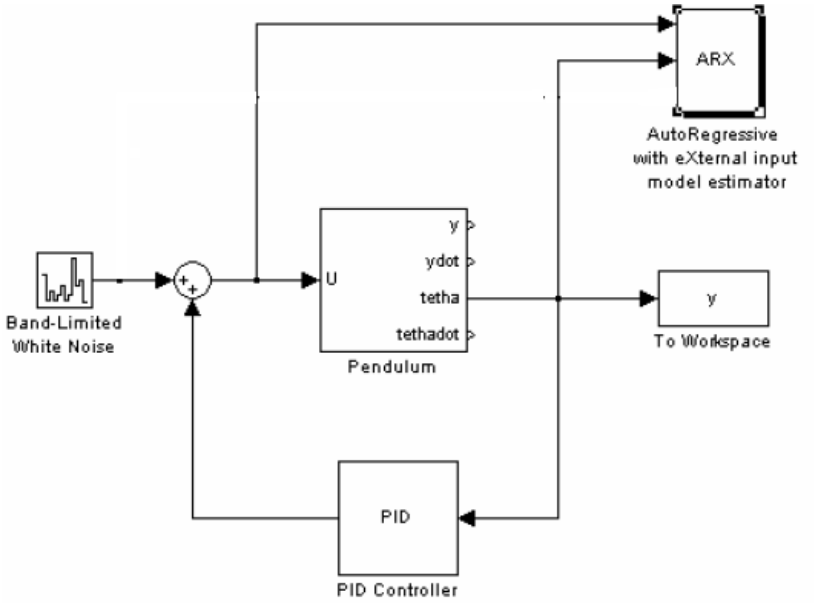


شکل(۱-۲۷) بلوک‌های شناسایی خطی سیستم

روی بلوک ARX دوبار کلیک کرده و در پنجره ظاهر شده مقادیر na، nb،و nc را تغییر دهید. برای رسیدن به بهترین مدل ARX باید na و nbهای مختلفی تست شوند. توجه کنید که ARX221 به معنی na=2، nb=2، و nc=1 است.



شکل(۱-۲۸) تنظیات بلوک ARX

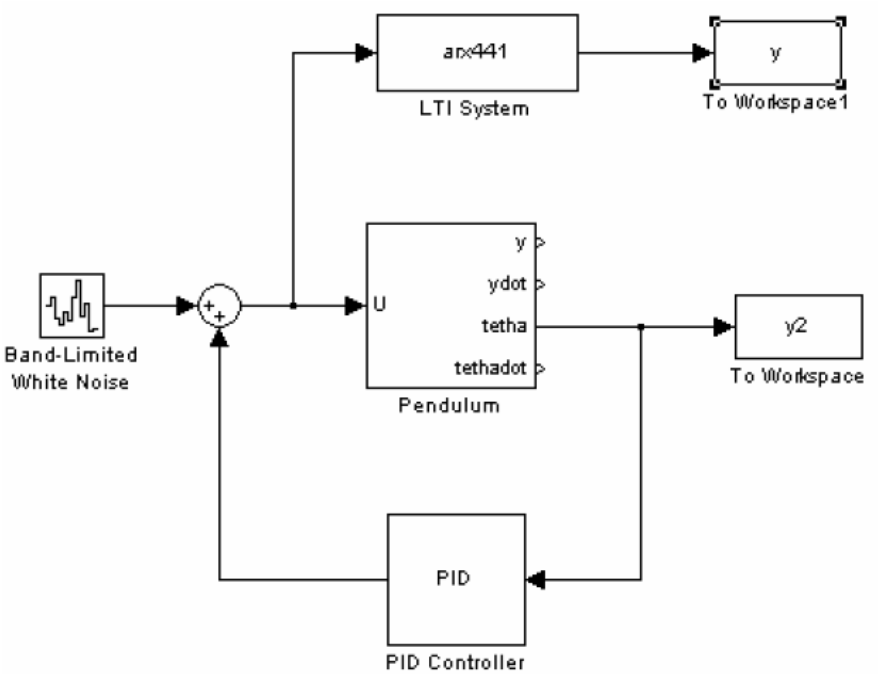


شکل(۱-۲۹) شناسایی سیستم با استفاده از مدل ARX

جدول (۱-۱) مقادیر مختلف na وnb را نشان می‌دهد. کمترین MSE مربوط به ARX221 وARX421 است. برای تولید Data2 روی بلوک Band limited white Noise دوبار کلیک کرده و مقدار seed را تغییر دهید و طبق برنامه‌ای که در بالا آمد، مقدار z1 و z2 را دوباره محاسبه کنید.

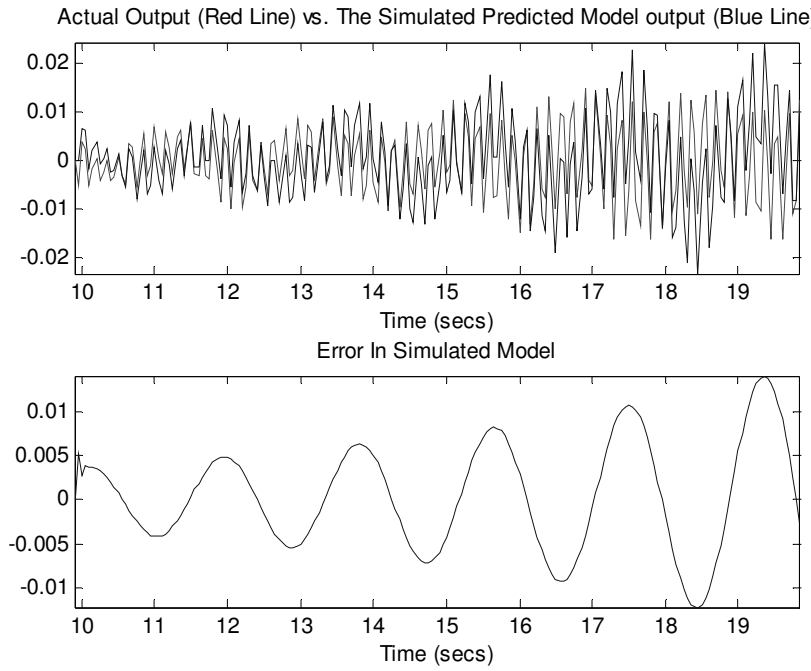
جدول(۱-۱) مقادیر MSE برای داده‌های ارزیابی ARX

ARX [na nb nk]	Data 1	Data2
1 1 1	0.0053	0.0057
2 2 1	0.0040	0.0054
3 2 1	0.0042	0.0057
3 3 1	0.1105	0.1158
4 2 1	0.0041	0.0052
4 3 1	0.0113	0.0175
4 4 1	0.0104	0.0167
5 2 1	0.0043	0.0058
5 3 1	0.0124	0.0152
5 5 1	0.3143	0.3249



شکل(۱-۳۰) استفاده از تابع تبدیل ARX در شناسایی سیستم

نمودارهای زیر خروجی مدل و خروجی پروسه واقعی را نمایش می‌دهند و به نظر می‌رسد خروجی مدل، خروجی واقعی سیستم را ردگیری می‌کند. برای یافتن بهترین پاسخ بهتر است که تابع تبدیل ARX به‌دست آمده و با seedهای مختلف تست شود.

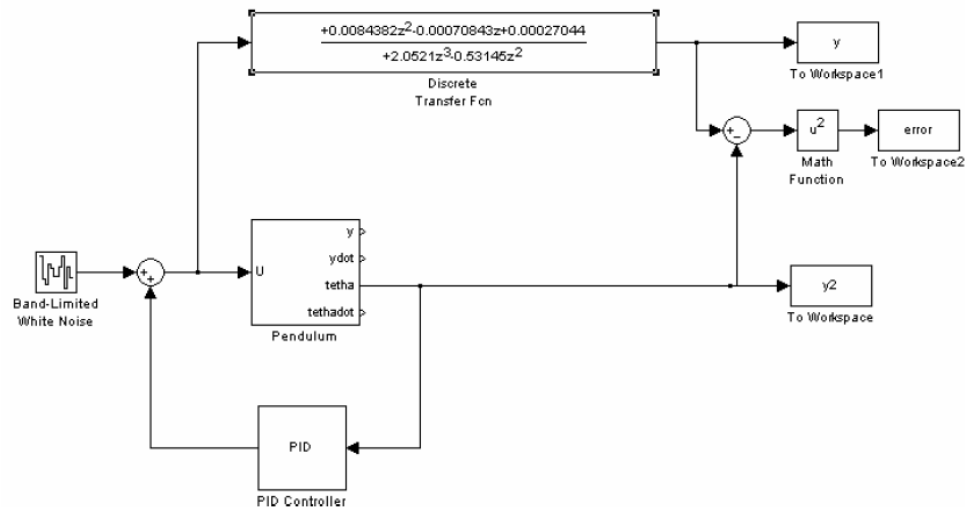


شکل(۱-۳۱) مدل ARX421 همراه با داده‌های ارزیابی۲

انتظار داشتیم که خروجی مدل مشابه خروجی واقعی باشد، ولی با توجه به شبیه‌سازی به این نتیجه می‌رسیم که مدل‌های ARX در پیش‌گویی خروجی واقعی مدل توانمند نیستند.

مدل ARMAX نیز به خوبی نمی‌تواند خروجی پاندول معکوس را پیش‌بینی کند، چون در هر دو حالت فرض می‌شود که مدل طیف نویزی، مدل، و خروجی خصوصیت دینامیکی یکسانی دارند. در بسیاری از مقالات در زمینهٔ شناسایی حلقه بسته اشاره می‌شود که بهترین روش‌ها برای مدل‌سازی سیستم‌ها روش BJ وOE می باشند. همان‌طور که در شکل(۱-۳۲) نشان داده شده مدل‌های تولید شده با استفاده از بلوک‌های BJ وOE به صورت تابع تبدیل نوشته شده و شبیه سازی می‌شوند.

جدول‌های (۱-۲) و (۱-۳) نتایج به‌دست آمده از مدل‌های BJ و OE را نمایش می‌دهند. پارامترهای BJ/OE در هر شبیه‌سازی تغییر می‌کنند تا بهترین مدل را تعیین کنند. مدل BJ53331 کمترین MSE را دارد. مدل OE تا حدودی خطای بزرگتری نسبت به BJ دارد. توجه شود که مدل‌های خطی قادر به شناسایی سیستم‌های غیرخطی نیستند.



شکل(۱-۳۲) بررسی دقت BJ و OE

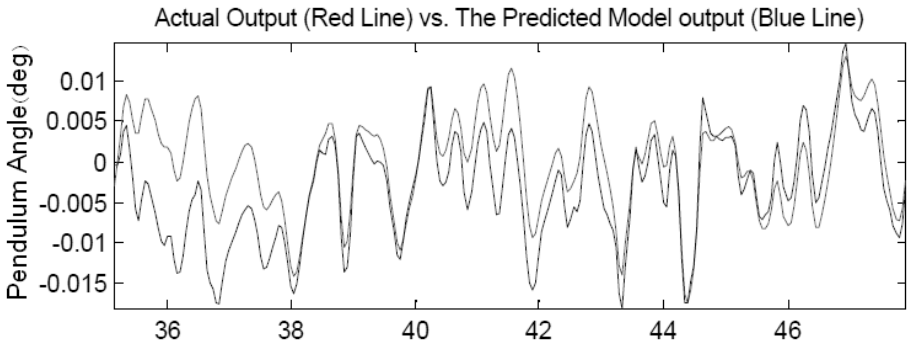
جدول (۱-۲) خطای مدل Box Jenkins

BJ [na nc nd nf nk]	MSE
2 1 1 1 0	3.6214e-004
2 1 1 1 1	4.1615e-004
2 1 2 1 0	4.7409e-004
2 1 2 2 0	8.7144e-004
3 1 1 1 0	2.8155e-004
3 1 2 2 0	1.2681e-004
3 2 2 2 0	1.2606e-004
4 3 3 3 1	5.3353e-005
5 3 3 3 1	4.6758e-005

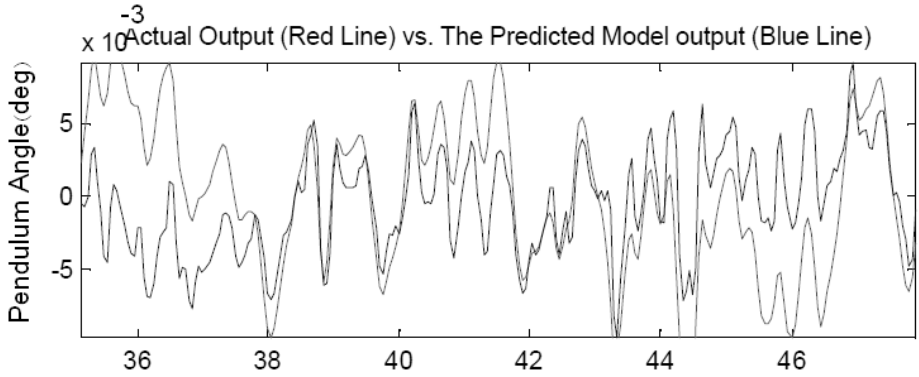
جدول(۱-۳) خطای مدل Output Error

OE [nb nf nk]	MSE
2 1 1	0.0014
2 2 1	0.0069
2 2 2	0.0104
3 1 1	0.0151
3 2 1	0.0074
3 2 2	0.0128
4 1 1	0.0039
4 2 1	0.0084
4 2 2	0.0126
5 1 1	0.0089

نتایج بهترین نمودارهای حاصل از مدل‌های BJ و OE در نمودارهای زیر آورده شده است.



شکل(۱-۳۳) تست مدل BJ[5 3 3 3 1] با داده‌های ارزیابی

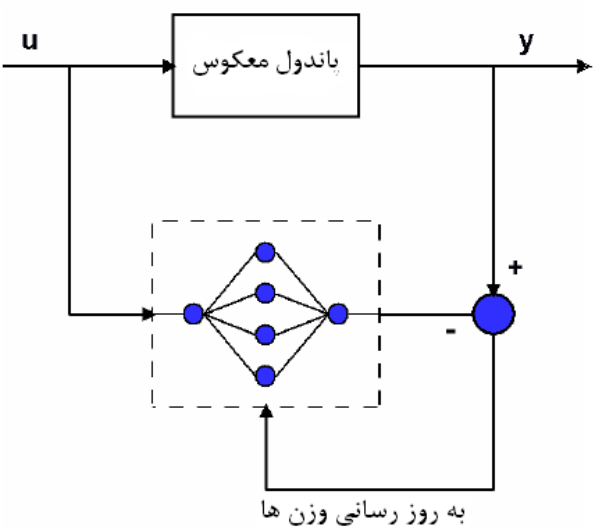


شکل(۱-۳۴) تست مدل OE[2 2 1] با داده‌های ارزیابی

۱- ۱۳ شناسایی غیرخطی مدل سیستم با استفاده از شبکه عصبی

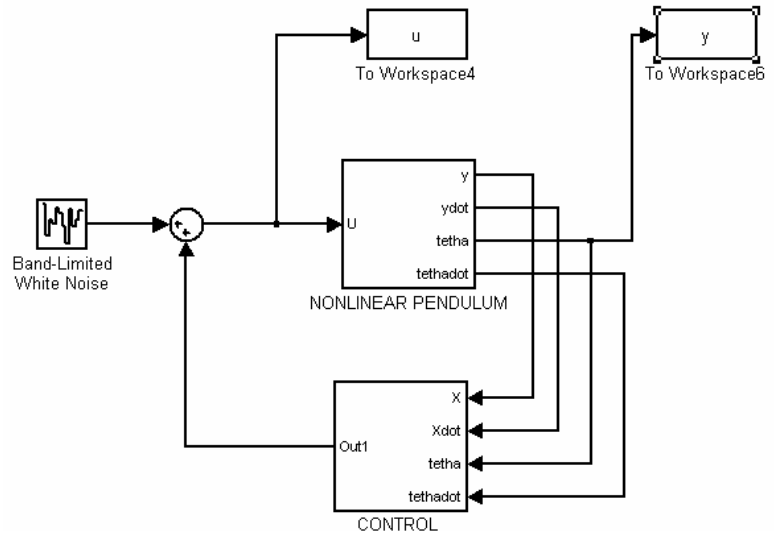
یکی از رایج‌ترین روش‌هایی شناسایی سیستم توسط شبکه عصبی ، مدل‌سازی پیشخور[1] است. در طول آموزش، سیستم و شبکه عصبی یک ورودی یکسان را دریافت می‌کند. سپس خروجی شبکه و خروجی واقعی مقایسه شده و خطای محاسبه شده برای به روز رسانی وزن‌های شبکه عصبی استفاده می‌شوند. این حالت یک نمونه، یادگیری نظارت شده[2] است.

[1] Feed Forward
[2] Supervised



شکل(۱-۳۵) روش مدل‌سازی پیشخور شبکه عصبی

به منظور اینکه یک مجموعه داده برای شبکه فراهم شود تا عمل یادگیری انجام شود، مدل سیمولینک به‌همراه کنترل فیدبک استفاده می‌شود.



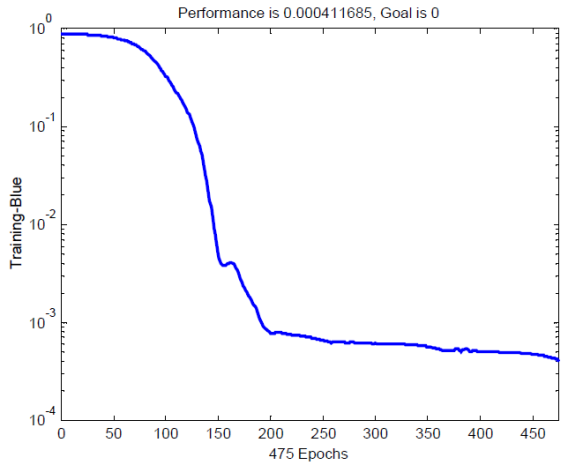
شکل(۱-۳۶) استفاده از داده‌های مدل غیرخطی به‌عنوان اهداف شبکه عصبی

در ابتدا یک مدل تک ورودی و تک خروجی شبکه عصبی توسعه پیدا می‌کند. ورودی همان نیروی کنترلی ، و خروجی زاویه پاندول است. کدهای نوشته شده یک شبکه عصبی پیشخور را ایجاد می‌کند.

```
net = newff([-10 10],[4 1],{'tansig' 'purelin'},'trainlm');
net.trainParam.epochs = 400;
net.trainParam.lr = 0.001;
```

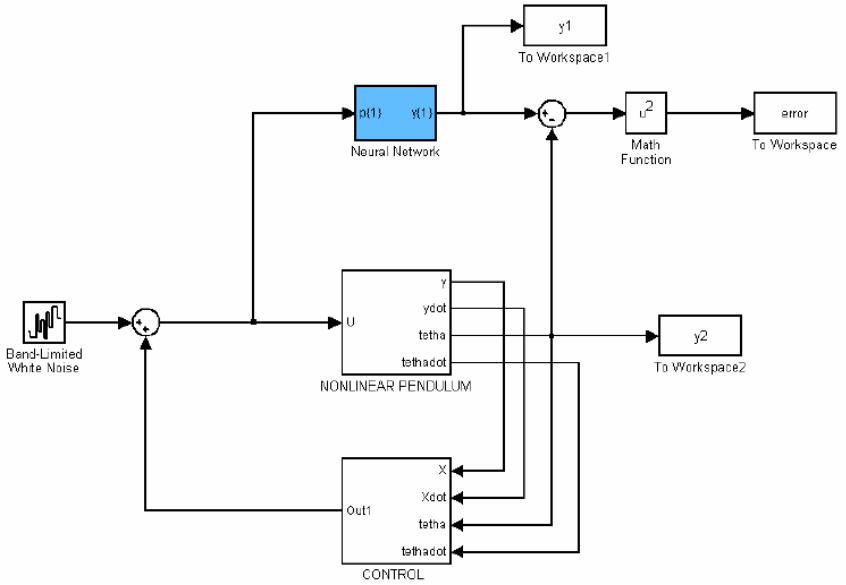
```
net = train(net,u(1:1000)',y(1:1000)');
gensim(net)% Generate Simulink block for neural network
simulation
```

شکل (۱-۳۷) روند آموزش شبکه MLP را نشان می‌دهد. در ابتدا خطای بین خروجی شبکه و پاندول زیاد است و هرچه تعداد سیکل‌ها افزایش یابد، این خطا کاهش می‌یابد.



شکل(۱-۳۷) تغییر مقدار خطا در حین یادگیری شبکه

پس از انجام آموزش، بلوک Neural Network توسط دستور gensim ساخته می‌شود، سپس باید آن را وارد محیط شبیه سازی سیستم کنیم، تا پاسخ شبکه عصبی به ورودی‌های سیستم را به‌دست آوریم.



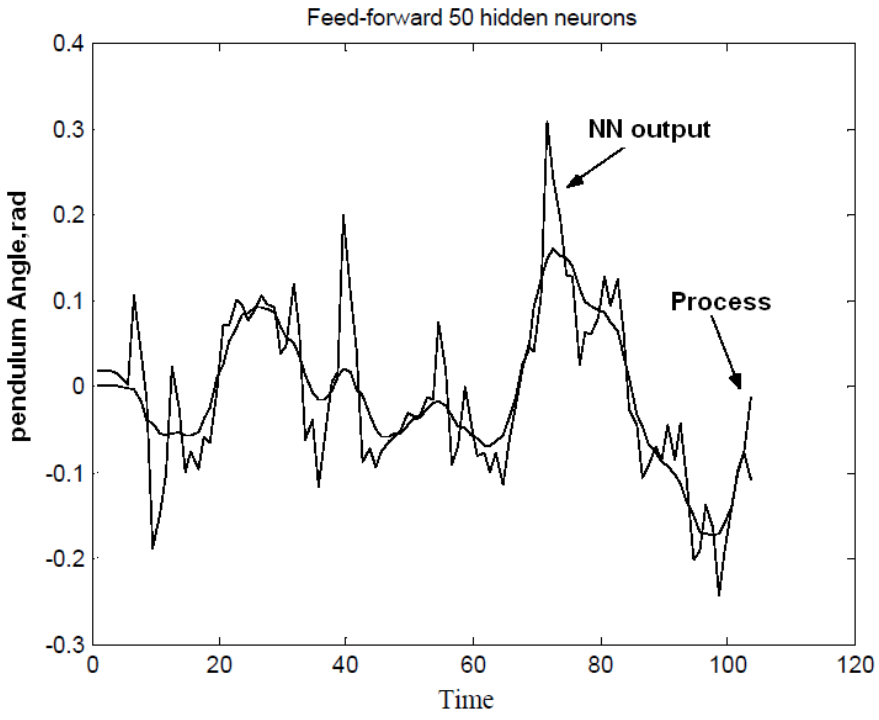
شکل(۱-۳۸) بررسی عملکرد مدل شبکه عصبی

برای تست عملکرد مدل مقادیر اولیه سیگنال ورودی باید تغییر کنند. جدول (۱-۴) نتایج شبیه‌سازی برای چند شبکه با نرون‌های متفاوت درلایه‌های مخفی را نشان می‌دهد. باید توجه کرد که در همه شبیه‌سازی‌ها مقدار seed اولیه باید ثابت نگه‌داشته شود.

جدول(۱-۴) نتایج شبکه پیشخور (Feed Forward)

نوع ANN	تعداد نرون ها در لایه مخفی	سیکل های آموزش	سرعت یادگیری	MSE
FF	50	500	0.0001	3.2622-6
FF	20	500	0.0001	3.4465e-6
FF	10	500	0.0001	1.1103e-5
FF	4	500	0.0001	1.54e-5

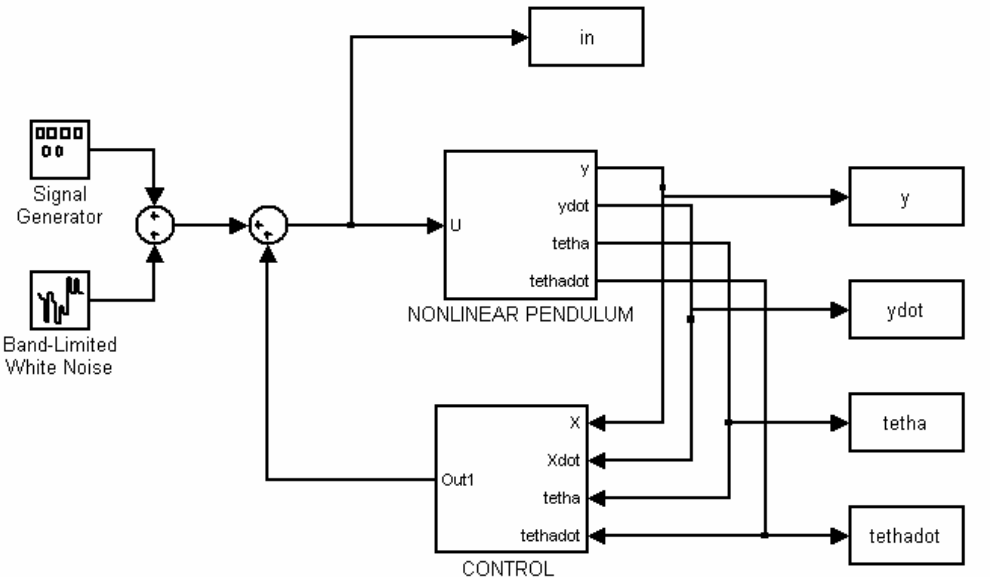
از پاسخ‌ها مشخص است که مدل شبکه‌ای پروسه را به خوبی شبیه‌سازی می‌کند و مقدار MSE کم است. مدل عصبی نیز زاویه پاندول را به خوبی پیش بینی می‌کند.



شکل(۱-۳۹) شبکه عصبی پیشخور با یک لایه مخفی و ۵۰ نرون مخفی

۱۴-۱ شناسایی چند خروجی

شکل (۱-۴۰) پاندول معکوس غیرخطی به همراه قانون کنترلی را نمایش می‌دهد. در این قسمت می‌خواهیم از یک شبکه پیشخور برای مدل سازی سیستم با چند خروجی استفاده کنیم. ورودی شبکه همانند قبل است ولی تعداد خروجی‌ها برابر ۴ می‌باشد.

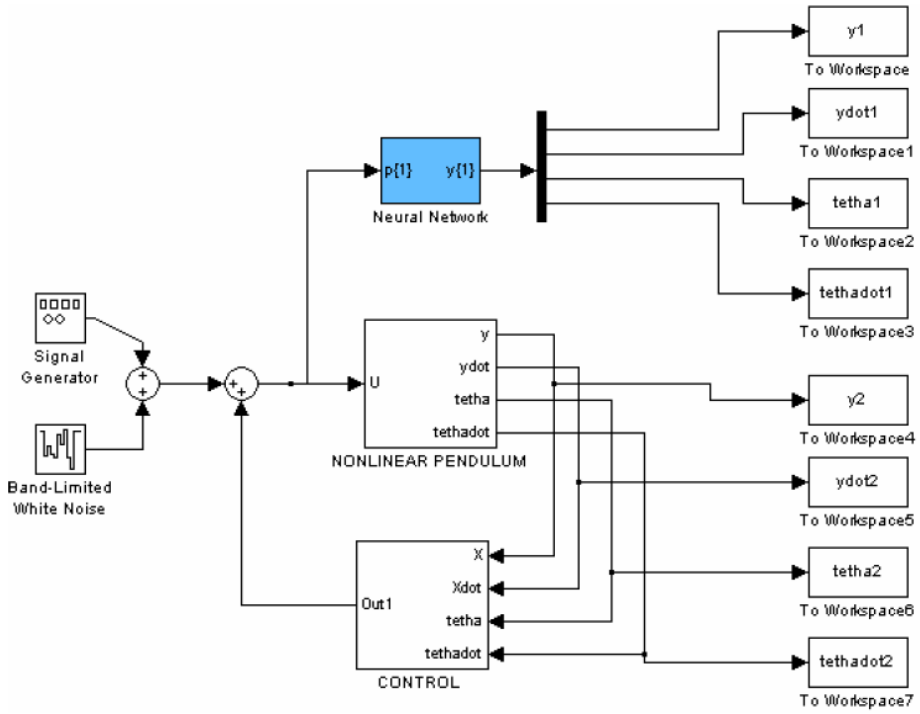


شکل(۱-۴۰) مدل غیرخطی پاندول معکوس برای آموزش شبکه عصبی

این شبکه را به صورت زیر تعریف می‌کنیم:

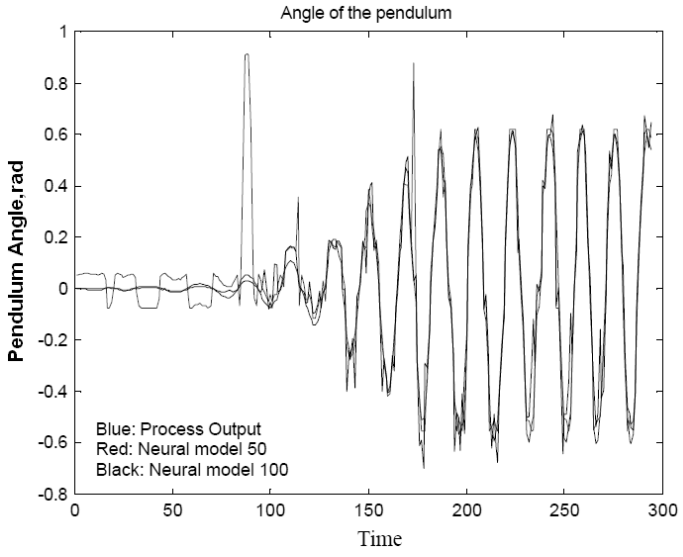
```
clear tempP
for k = 1:200,
 P = [y(k);ydot(k);tetha(k);tethadot(k)];
 tempP = [tempP P];
end
net = newff([-10 10],[50 4],{'tansig' 'purelin'},'trainlm');
net.trainParam.epochs = 500;
net = train(net,in(1:200)',tempP);
```

بعد از آموزش شبکه بلوک شبکه عصبی با دستور gensim ایجاد می‌شود. سپس بلوک تولید شده را در سیستم قرار می‌دهیم تا خروجی‌های شبکه و خروجی‌های واقعی را با هم مقایسه کنیم.

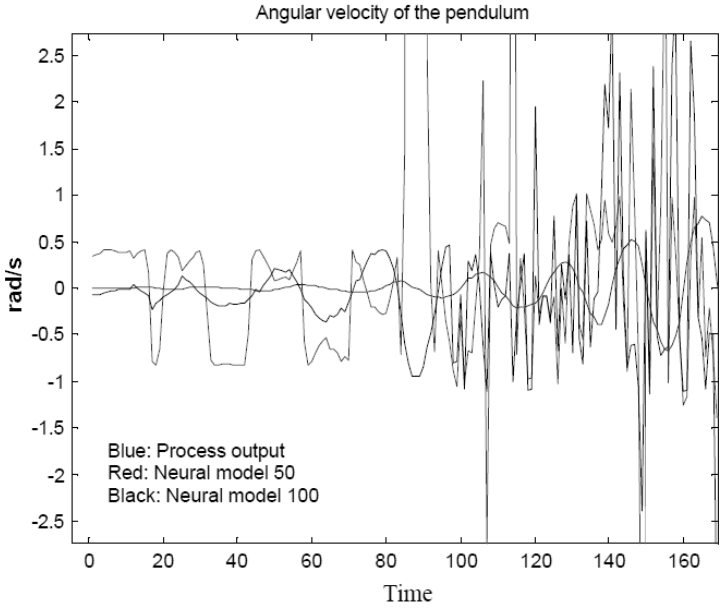


شکل(۱-۴۱) مقایسه خروجی‌های مدل شبکه عصبی و پاندول معکوس

شکل‌های زیر خروجی‌های شبکه و خروجی‌های واقعی را با هم در یک نمودار رسم کرده است. همان‌طور که از شکل‌ها بر می‌آید شبکه عصبی قادر نیست زاویه و سرعت ارابه را به خوبی مدل‌سازی کند.



شکل(۱-۴۲) زاویه پاندول



شکل(۱-۴۳) سرعت زاویه‌ای پاندول

کنترلر عصبی برای پاندول معکوس

تا اینجا روش‌های مختلف شناسایی سیستم به مدل پاندول معکوس اعمال شده‌اند. نتایج شناسایی خطی نشان می‌دهد که استفاده از روش‌های حلقه بسته، این امکان را به‌وجود می‌آورد که مدل‌های دقیق از سیستم ایجاد کنیم و همان‌طور که دیدیم مقدار MSE به‌دست آمده از مدل‌های BJ و OE بسیار کوچک بودند. در ادامه به دلیل ناکارآمد بودن روش شناسایی خطی در مدل‌سازی پاندول معکوس غیرخطی، از شبکه عصبی برای مدل‌سازی سیستم غیرخطی استفاده می‌شود. قبل از اینکه پاندول غیرخطی توسط شبکه عصبی شناسایی شود، این سیستم توسط کنترل فیدبک پایدار شد.

سایزهای مختلف شبکه عصبی پیشخور (FF) با استفاده از داده‌های ورودی و خروجی پاندول معکوس غیرخطی آموزش داده شدند و با استفاده از دستور gensim بلوک شبکه عصبی در سیمولینک برای تست کردن ایجاد شد. همانطور که در نتایج به‌دست آمده مشاهده کردیم، شبکه پیشخور، دینامیکی مشابه پاندول معکوس را از خود نشان داد و مقدار MSE کوچکی به‌دست آمد.

هدف اصلی در این قسمت طراحی کنترلر برای معکوس نگه‌داشتن پاندول معکوس است. در ابتدا مطالبی درباره طراحی کنترلر برای پاندول معکوس یادآور می‌شویم. پاندول در حالت حلقه باز ناپایدار است. از طرفی سیستم غیرخطی و با چند خروجی است. در ابتدا برای نشان دادن فواید کنترل عصبی، مقایسه‌ای بین کنترل عصبی و PID انجام می‌دهیم.

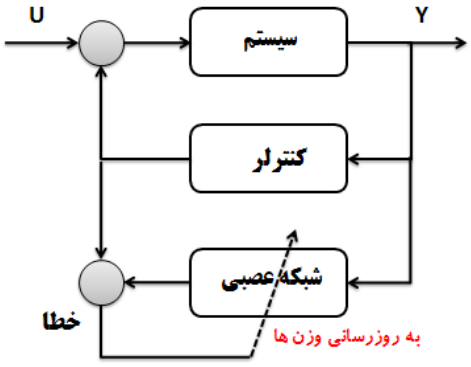
سیستم غیرخطی: کنترلرهای PID خطی استاندارد به‌دلیل غیرخطی‌های پیچیده‌ای که در مدل پاندول معکوس وجود دارد، قادر به کنترل این سیستم نیستند.

سیستم چند خروجی: پاندول معکوس چهار خروجی دارد. به منظور اینکه فیدبک کلیه متغیرهای حالت را داشته باشیم، باید از چهار کنترلر PID استفاده کنیم. شبکه‌های عصبی به‌دلیل ماهیت موازی بودن فواید بیشتری داشته و می‌تواند جایگزین چهار کنترلر PID شود.

ناپایداری در حالت حلقه باز: پاندول معکوس در حالت حلقه باز ناپایدار است. به محض اینکه سیستم شبیه سازی شود، پاندول خواهد افتاد. آموزش شبکه عصبی زمان می‌برد، بنابراین پاندول تا زمانی‌که شبکه آموزش ببیند، در این مدت باید پایدار باقی بماند.

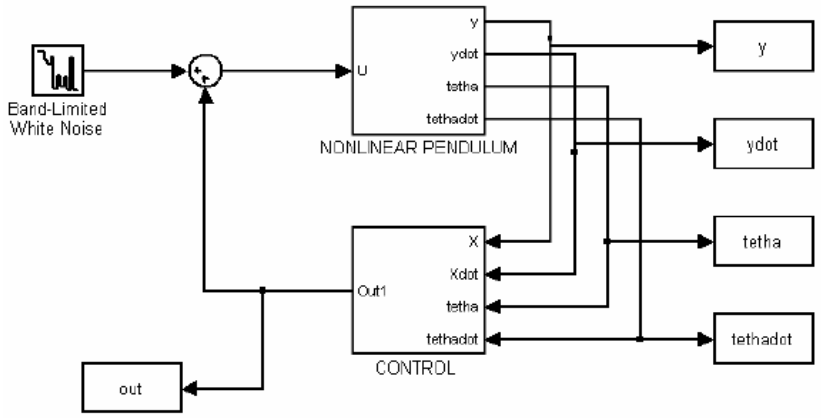
۱-۱۲ کنترل نظارت شده

در این نوع رویکرد کنترلی، یک شبکه عصبی طوری آموزش می‌بیند، که عملکردی مشابه کنترلر موجود داشته باشد. اما این سوال پیش می‌آید که چرا می‌خواهیم از یک کنترلر که در حال کار کردن است کپی برداری کرده و مدل شبکه عصبی کنترلر را بسازیم. بیشتر کنترلرهای قدیمی مثل خطی‌سازی به کمک فیدبک براساس نقطه کاری سیستم کار می‌کنند و این بدان معنی است که کنترلر در صورتی می‌تواند به خوبی عمل کند که سیستم مورد بررسی حول این نقطه کاری مشخص کار کند. بنابراین در این سیستم‌ها در صورتی‌که در سیستم عدم قطعیت وجود داشته باشد و یا تغییرات ناشناخته‌ای به سیستم اعمال شود، کنترلر به خوبی نمی‌تواند جوابگو باشد.



شکل(۱-۴۴) یادگیری نظارت شده با استفاده از کنترلر موجود

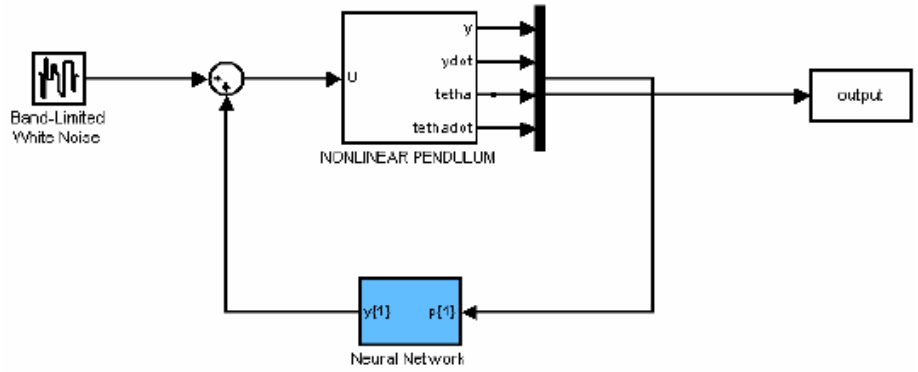
یکی از مزیت‌های شبکه عصبی این است که می‌تواند در صورت وجود عدم قطعیت در سیستم، پارامترهای خود را با شرایط فعلی تطبیق داده و عمل کنترل را به خوبی انجام دهد. در روش کنترل نظارت شده، ناظر داده‌های مطلوب برای یادگیری شبکه عصبی را فراهم می‌کند. در آموزش off-line اهداف شبکه با استفاده از کنترلر موجود فراهم می‌شود. شبکه عصبی وزن‌های خود را طوری تطبیق می‌دهد که خروجی‌ای مشابه کنترلر فعلی داشته باشد. به این منظور چهار ورودی شبکه عصبی یعنی y، ydot، tetha،و tethadot و یک خروجی شبکه یعنی out در فضای کاری MATLAB ذخیره می‌شوند.



شکل(۱-۴۵) آموزش نظارت شده با استفاده از قانون کنترلی

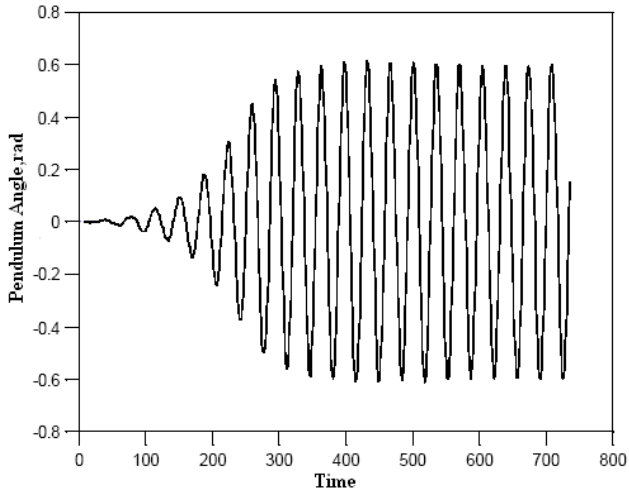
دستورات زیر برای آموزش شبکه عصبی استفاده می‌شوند. قسمت اول برنامه یک آرایه سلولی ایجاد می‌کند. این آرایه سلولی چهار ورودی مختلف را در یک بردار ورودی ترکیب می‌کند. شبکه پیشخور FF نیز دارای ۵۰ نرون در لایه مخفی است. تابع تبدیل لایه مخفی tan-sigmoid و لایه خروجی ، خطی در نظر گرفته می‌شود.

```
clear tempP
for k = 1:500,
 P = [y(k);ydot(k);tetha(k);tethadot(k)];
 tempP = [tempP P];
end
net= newff([-2 2;-2 2;-2 2;-2 2],[501],{'tansig','purelin'},
'trainlm');
net.trainParam.epochs = 500;
net = train(net,tempP,out(1:500)');
```



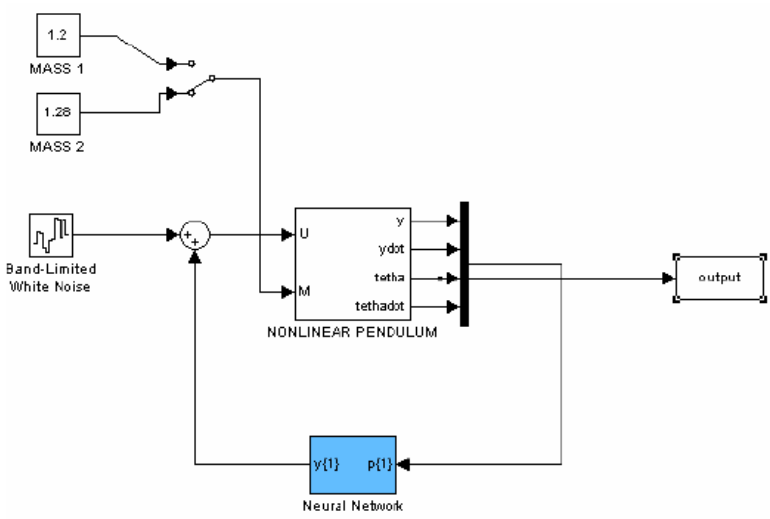
شکل(۱-۴۶) جایگزینی کنترلر اصلی با شبکه عصبی

شکل (۱-۴۷) خروجی به‌دست آمده با استفاده از این کنترلر عصبی را نشان می‌دهد. مقدار خطا حدود 10^{-7} است. بنابراین می‌توان گفت کنترلر تقریباً دقیق می‌باشد.

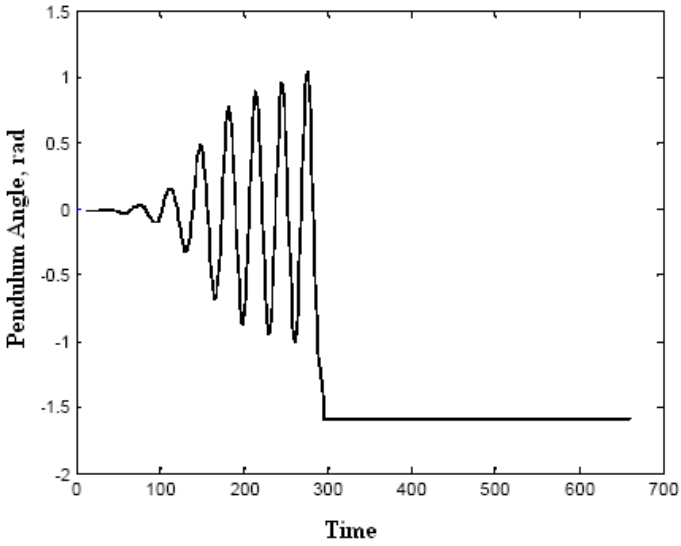


شکل(۱-۴۷) پاسخ حلقه بسته با استفاده از شبکه عصبی

در شبیه سازی بالا وزن‌های شبکه در ابتدا در محیط سیمولینک تنظیم شدند، اما در محیط سیمولینک این وزن‌ها ثابت باقی مانده و یادگیری on-line امکان پذیر نیست. در صورتی‌که پارامترهای پاندول ثابت باقی مانده و هیچ اغتشاشی به سیستم وارد نشود، مشکل خاصی برای سیستم کنترلی به‌وجود نمی‌آید. برای اعمال تغییر در سیستم، جرم ارابه را با استفاده از یک سوئیچ از 1.2kg به 1.28kg تغییر دهید و نتیجه شبیه سازی را مشاهد کنید.



شکل(۱-۴۸) اعمال اغتشاش به سیستم



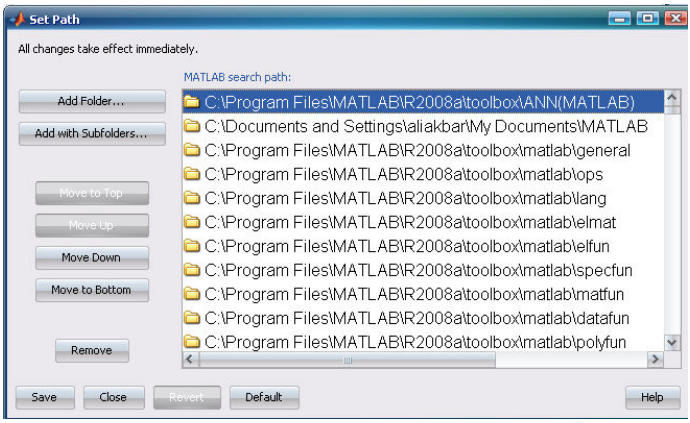
شکل(۱-۴۹) نمودار زاویه پاندول معکوس

۱-۱۵ اضافه کردن Toolbox به MATLAB

کاملاً واضح است که سیستم ناپایدار خواهد بود. این مشکل را می‌توان با استفاده از جعبه ابزار Adaptive Neural در سیمولینک برطرف کرد. پوشه این Toolbox در DVD همراه کتاب موجود است. فایل ANN را از پوشه Toolboxes انتخاب کرده و در مسیر زیر کپی کنید:

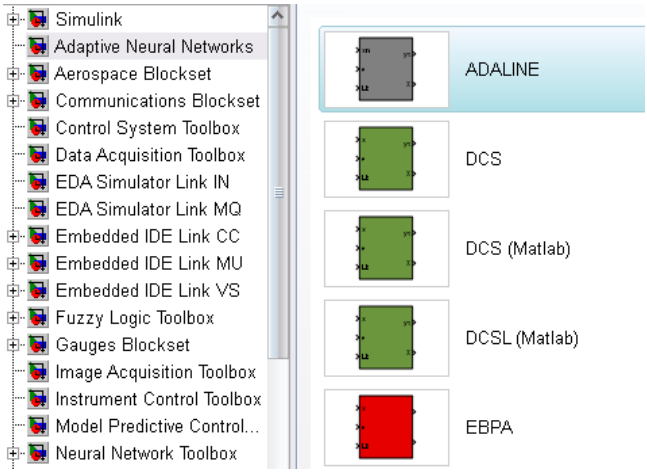
C:\Program Files\MATLAB\R2010a\toolbox

سپس از منوی فایل MATLAB گزینه Set path را انتخاب کنید. پوشه ANN را از مسیری که کپی کرده اید، پیداکرده و دکمه save را بزنید تا این پوشه در کتابخانه سیمولینک نمایش داده شود.



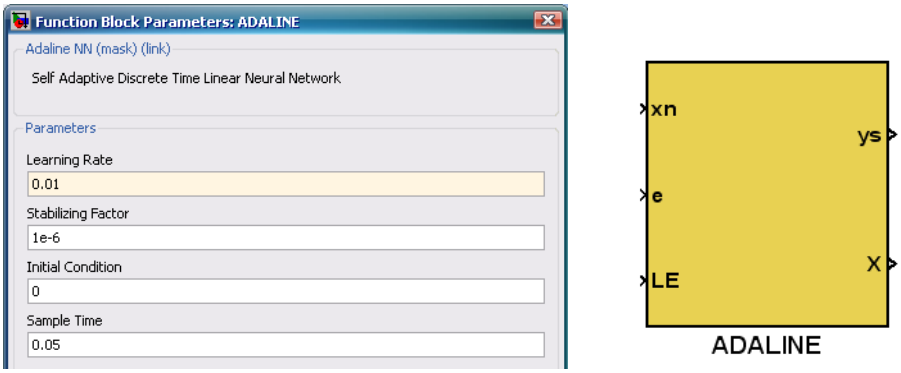
شکل(۱-۵۰)پنجره Set Path برای اضافه کردن Toolbox به MATLAB

بلوک‌های این جعبه ابزار در شکل (۱-۵۱) نمایش داده شده اند.



شکل(۱-۵۱) نمایش بلوک‌های Adaptive Neural Network در کتابخانه سیمولینک

نحوه اتصال این بلوک‌ها به بلوک‌های دیگر سیمولینک یکسان است. ورودی این بلوک‌ها عبارتند از:



شکل(۱-۵۲) بلوک ADALINE به همراه پنجره پارامترها

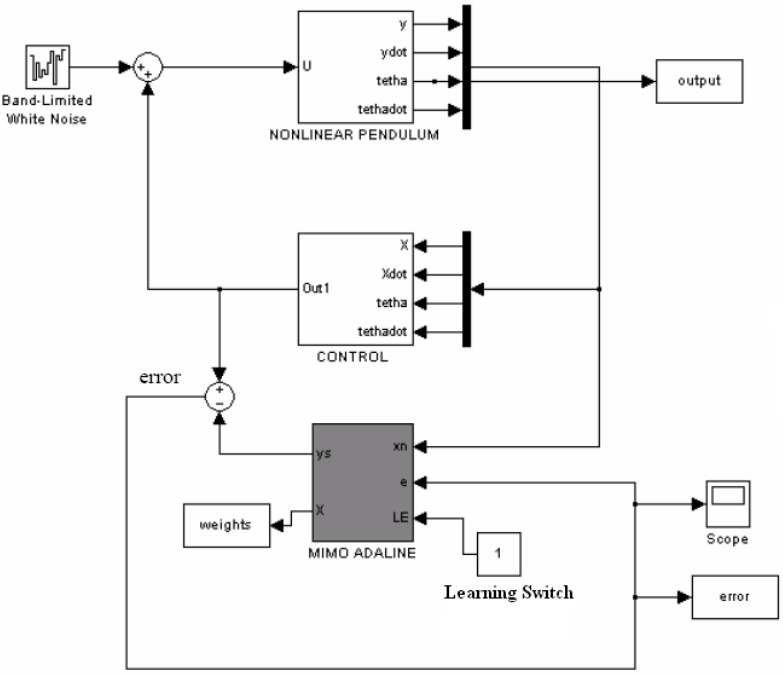
- x: ورودی اعمالی به شبکه عصبی
- E: خطای بین خروجی واقعی و تقریب شبکه عصبی
- LE: یک سیگنال منطقی که آموزش شبکه را فعال یا غیرفعال می‌کند.

و خروجی این بلوک‌ها عبارتند از:

- Ys: مقدار تابع تقریب زده شده
- X: تمام حالتهای شبکه عصبی ، شامل وزنها و همه پارامترهایی که در طول یادگیری تغییر می‌کنند.

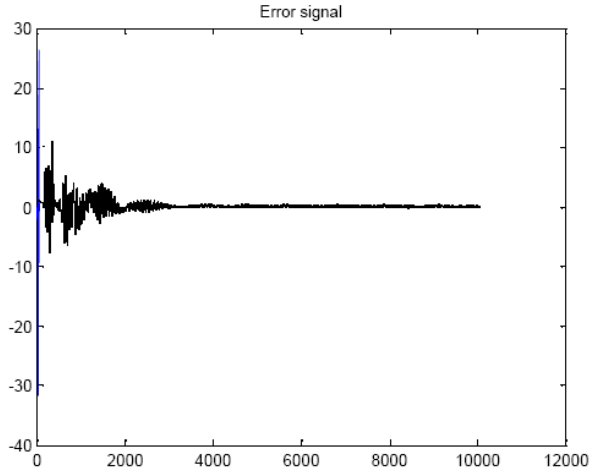
اولین شبکه مورد استفاده ADALINE می‌باشد. این بلوک برای تقریب توابع تقریباً خطی استفاده می‌شود. ADALINE به‌صورت off-line با استفاده از سیستم کنترلی آموزش داده می‌شود. سیگنال خطا برابر اختلاف خروجی کنترلر و خروجی شبکه عصبی می‌باشد. سپس این سیگنال خطا به شبکه عصبی فیدبک شده و

وزن‌های شبکه توسط گرادیان کاهشی به روز می‌شوند. سرعت یادگیری ADALINE را روی 0.01 و سرعت نمونه گیری را روی 0.05 قرار دهید.



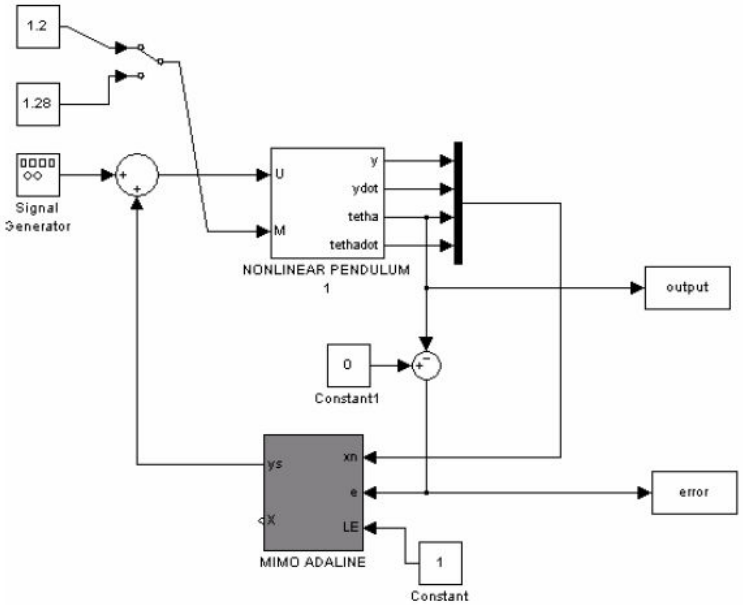
شکل(۱-۵۳) شبکه عصبی تطبیقی که توسط کنترلر اصلی آموزش داده می‌شود

شکل (۱-۵۴) خطای بین خروجی کنترلر و خروجی شبکه عصبی را نشان می‌دهد. ملاحظه می‌شود که مقدار این خطا به سمت صفر می‌رود و وزن‌های شبکه به سمت مقادیر نهایی همگرا می‌شوند.



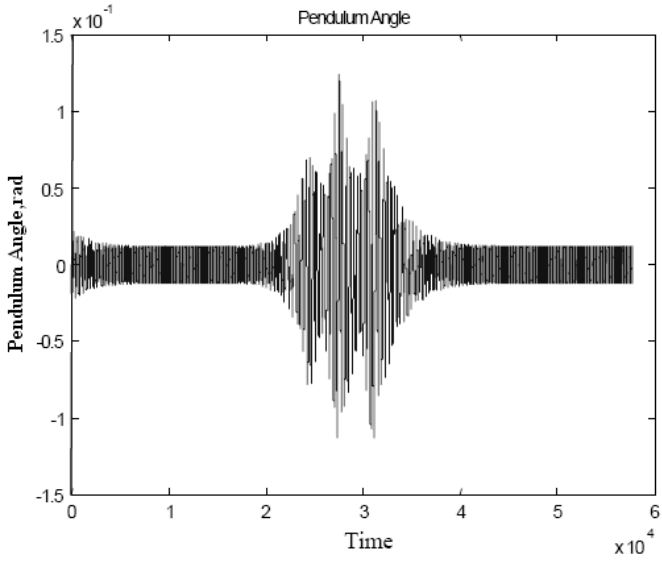
شکل(۱-۵۴) سیگنال خطا

حال شبکه می‌تواند جایگزین کنترلر اصلی شود. برای اعمال اغتشاش خارجی به سیستم همانند حالت قبل عمل کرده و جرم ارابه را با یک سوئیچ تغییر دهید تا تاثیر آن را در خروجی مشاهده کنید.



شکل(۱-۵۵) اعمال اغتشاش به سیستم

نوسانات زاویه پاندول در حین شبیه سازی، با تغییرجرم ، به $0.1rad$ افزایش می‌یابد ولی بعد از سپری‌شدن مدتی، مقدار زاویه پاندول به مقدار طبیعی خود کاهش پیدا می‌کند.



شکل(۱-۵۶) تاثیر اغتشاشات روی زاویه پاندول معکوس

فصل دوم

فهرست دستورهای جعبه ابزار سیستم‌های کنترلی در نرم‌افزار MATLAB

۲ – ۱ فهرست دستورهای نرم افزار MATLAB

Abs	محاسبه قدر مطلق
Acker	محاسبه ماتریس k از روی جایگذاری قطب‌های A-BK در مکان مورد نظر
Axis	تنظیم مقیاس شکل یا نمودار موجود
Bode	رسم نمودار بود
c2dm	تبدیل سیستم‌های پیوسته به سیستم‌های گسسته با روش zoh
Care	حل معادلات جبری پیوسته ریکاتی
Clf	پاک کردن نمودار موجود
Conv	ضرب چند جمله‌ای‌ها در هم (دستور deconv را هم ببینید)
Ctrb	ماتریس کنترل پذیری را محاسبه می‌کند (دستور obsv را هم ببینید)
Dare	حل معادلات جبری گسسته ریکاتی
Deconv	تقسیم چند جمله‌ای ها(دستور conv را هم ببینید)
Det	محاسبه دترمینان یک ماتریس
Dimpulse	محاسبه پاسخ ضربه سیستم‌های خطی زمان گسسته (دستورdstep را هم ببینید)
Dlqr	طراحیLQR برای سیستم‌های خطی زمان گسسته (دستورlqr را هم ببینید)
Dlsim	شبیه سازی سیستم‌های خطی زمان گسسته (دستورlsim را هم ببینید)
Dlyap	حل معادلات لیاپانوف گسسته
Dstep	پاسخ پله سیستم‌های خطی زمان گسسته (دستورstairs را هم ببینید)
Eig	محاسبه مقادیر ویژه ماتریس
Eps	عددی بسیار کوچک
Feedback	ارتباط دو سیستم در ساختار بازخوردی (feedback)
Figure	رسم یک شکل جدید (دستورات subplot, axis را ببینید)
Filt	مشخص کردن فیلتر دیجیتال
For	ایجاد حلقه در برنامه
format	فرمت عدد را مشخص می‌کند
function	ایجاد یک تابع در یک M-file
grid	شبکه کردن صفحه نمودار

gtext	افزودن متن به یک نمودار
help	HELP!
hold	نگه داشتن یک نمودار برای نمایش با دیگر نمودار ها
if	انجام دادن یک دستور العمل
imag	نمایش قسمت موهومی یک عدد مختلط (دستور real را هم ببینید)
impulse	پاسخ ضربه سیستم‌های پیوسته خطی و ثابت با زمان (دستور step, lsim, dlsim را هم نگاه کنید)
input	دستور ورود یک پارامتر به برنامه توسط کاربر
inv	یافتن معکوس یک ماتریس
jgrid	نمایش خطوط شبکه‌ای ثابت‌های ضریب میرایی و زمان نشست (دستورات sgrid, sigrid, zgrid را هم نگاه کنید)
kalman	طراحی تخمین زننده کالمن
kalmdemo	طراحی و شبیه سازی فیلتر کالمن
legend	ایجاد راهنما برای یک گراف
length	طول رشته یا بردار (دستور size را هم نگاه کنید)
linspace	ایجاد یک بردار با بازه‌های مساوی
lnyquist1	نمودار نایکوئیست در مقیاس لگاریتمی (دستور nyquist1 را هم ببینید)
log	محاسبه لگاریتم طبیعی
loglog	رسم نمودار با مقیاس log-log
logspace	ایجاد یک بردار با فواصل لگاریتمی
lqr	طراحی LQR برای سیستم‌های خطی (دستور dlqr را هم ببینید)
Lqrd	طراحی LQ گسسته برای سیستم‌های پیوسته
lqry	طراحی LQ با توزین خروجی
lsim	شبیه سازی سیستم‌های خطی
lyap	حل معادلات لیاپانوف پیوسته
margin	محاسبه حد فاز وحد بهره
norm	محاسبه نرم یک بردار
nyquist	رسم نمودار نایکوئیست
obsv	محاسبه ماتریس رویت پذیری
ones	ایجاد بردار یا ماتریس با درایه‌های واحد
place	محاسبه ماتریس K با استفاده از جایابی قطب‌ها در محل مورد نظر
plot	رسم نمودار(دستورات figure, axis, subplot را ببینید)
poly	محاسبه چند جمله‌ای مشخصه برای یک ماتریس

polyadd	جمع کردن دو چند جمله‌ای با هم
polyval	محاسبه مقدار چند جمله‌ای
pzmap	رسم محل صفر و قطب سیستم‌های خطی در صفحه s.
rank	محاسبه رتبه یک ماتریس
real	محاسبه قسمت حقیقی یک عدد موهومی
rlocfind	یافتن مقدار k در محل مشخص شده بر روی ترسیمه مسیر ریشه‌ها
rlocus	رسم مکان هندسی ریشه ها
roots	پیدا کردن ریشه‌های یک چند جمله‌ای
rscale	پیدا کردن ضریب مقیاس در سیستم‌های فیدبک کامل متغیرهای حالت
set	تنظیم تعداد فاصله‌های خالی روی هر یک از محور ها
series	اتصال داخلی سری سیستم‌های خطی مستقل از زمان
sgrid	نمایش خطوط شبکه‌ای ثابت‌های ضریب میرایی و فرکانس طبیعی (دستورsigrid, zgrid را هم ببینید)
sigrid	نمایش خطوط شبکه‌ای با زمان نشست ثابت (دستور sigrid, zgrid را هم ببینید)
size	محاسبه ابعاد یک بردار یا یک ماتریس
sqrt	محاسبه جذر
ss	ایجاد مدل فضای حالت یا تبدیل سیستم LTI به فضای حالت
ss2tf	تبدیل مدل فضای حالت به تابع تبدیل
ss2zp	تبدیل مدل فضای حالت به فرم نمایش صفر و قطب
stairs	رسم پاسخ پله سیستم گسسته
step	رسم پاسخ پله
subplot	تقسیم پنجره نمایش منحنی به چند قسمت
text	افزودن یک متن به نمودار
tf	ایجاد تابع تبدیل
tf2ss	تبدیل نمایش تابع تبدیلی به فرم فضای حالت
tf2zp	تبدیل نمایش تابع تبدیل به فرم نمایشی صفر و قطب
Title	افزودن عنوان به یک نمودار
Bandwidth	محاسبه فرکانس پهنای باند
xlabel/ylabel	افزودن یک متن به محورهای x,y
Zeros	ایجاد بردار یا ماتریس صفر
Zgrid	نمایش خطوط شبکه‌ای ثابت‌های ضریب میرایی و فرکانس طبیعی (دستورsigrid, zgrid را هم ببینید)
zp2ss	تبدیل نمایش صفر و قطب به مدل فضای حالت
zp2tf	تبدیل نمایش صفر و قطب به تابع تبدیل

۲ – ۲ دستور acker

هدف: طراحی جایابی قطب ها برای سیستمهای تک ورودی **K=acker(A,B,P)**

توضیح: سیستم تک ورودی زیر مفروض است:

$$\dot{X} = AX + Bu$$

می‌خواهیم بردار فیدبک حالت K را چنان طراحی کنیم که قطبهای سیستم حلقه بسته در محلهای تعیین شده در بردار P قرار گیرند. دستور k=acker(A,B,P) با استفاده از فرمول آکرمن بردار بهره فیدبک (K) را چنان مشخص می‌کند که قانون فیدبک u=-Kx، قطبهای سیستم حلقه بسته را در محل‌های تعیین شده در بردار P قرار می‌دهد. به عبارت دیگر مقادیر ویژه ماتریس A-BK در محل اعضای بردار P قرار می‌گیرند. به عنوان نمونه در تکه برنامه زیر با طراحی فیدبک حالت K قطبهای سیستم ناپایدار به P=[-2,-3] منتقل می‌شوند.

علاوه براین چنانچه سیستم تک خروجی باشد، می‌توان از دستور acker برای محاسبه بهره رویتگر لیونبرگر نیز استفاده کرد. اگر Y=Cx معادله خروجی سیستم باشد، دستور زیر بردار L را به عنوان بهره رویتگر محاسبه خواهد کرد و قطبهای رویتگر در محلهای مشخص شده توسط بردار P قرار می‌گیرند.

```
A=[2 0;-1 -1];
B=[1;1]
P=[-2,-3 ]
K=acker(A,B,P)
```

```
K =
   6.5000   -0.5000
```

محدودیت‌ها

۱. دستور acker فقط در سیستمهای تک ورودی قابل استفاده است.

۲. (A,B) باید کنترل پذیر باشد.

۳. این روش از قابلیت اطمینان بالایی برخوردار نیست و چنانچه درجه سیستم بیش از ۵ باشد یا کنترل پذیری سیستم ضعیف باشد دچار اشکال خواهد شد.

توجه: دستور place عمل جایابی قطب را در سیستم‌های چند ورودی انجام می‌دهد، اما اگر محل قطب‌ها یکسان باشد نمی‌توان از این دستور استفاده کرد.

۲ – ۳ دستور canon

هدف: محاسبه تحقق‌های استاندارد (کانونیکال) در فضای حالت

CSYS= canon(SYS,'TYPE')
[CSYS,T]= canon(SYS,'TYPE')

توضیح: این دستور یک تحقق کانونیکال در فضای حالت برای سیستم‌های LTI پیوسته یا زمان گسسته که با SYS مشخص شده است، ارائه می‌دهد. دو نوع تحقق کانونیکال در این دستورالعمل در نظر گرفته شده است که توسط متغیر TYPE انتخاب می‌گردد.

الف- فرم MODAL

دستور CSYS=canon (sys,'modal') یک تحقق MODAL از سیستم sys را به‌دست آورده و در csys می‌ریزد. در این تحقق مقادیر ویژه حقیقی sys روی قطرهای ماتریس A قرار می‌گیرند و به ازاء هر زوج قطب موهومی یک بلوک ۲ در۲ در ماتریس A ظاهر می‌شود.

حالت‌های زیر در محاسبه ماتریس مدال وجود دارد:

۱. اگر ماتریس حالت سیستم A دارای n مقدار ویژه متمایز حقیقی و غیر تکراری $\lambda_1, \lambda_2, \ldots, \lambda_n$ باشد.

۲. اگر ماتریس A مقادیر ویژه مختلط داشته باشد.

۳. اگر ماتریس حالت سیستم A دارای مقادیر ویژه تکراری باشد.

به عنوان مثال، برای سیستمی با مقادیر ویژه $(\lambda_1, \sigma \pm \omega j, \lambda_2)$ ماتریس A در تحقق فرم MODAL به صورت زیر خواهد بود:

$$Modal\ Form: \begin{bmatrix} \lambda_1 & 0 & 0 & 0 \\ 0 & \sigma & \omega & 0 \\ 0 & -\omega & \sigma & 0 \\ 0 & 0 & 0 & \lambda_2 \end{bmatrix}$$

ب- فرم COMPANION

دستور CSYS=canon (SYS,'modal') یک تحقق COMPANION از sys ارائه می‌دهد. اگر معادله مشخصه سیستم به صورت زیر باشد:

$$P(s) = s^n + a_1 s^{n-1} + \ldots + a_{n-1} s + a_n$$

آنگاه ماتریس A در تحقق COMPANION به صورت زیر خواهد بود:

$$A_{Com} = \begin{bmatrix} 0 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \ddots & 1 \\ -a_0 & -a_1 & -a_2 & -a_3 & \cdots & -a_{n-1} \end{bmatrix}$$

اگر sys به صورت مدل فضای حالت وجود داشته باشد آنگاه دستور

[CSYS,T]=canon(SYS,'TYPE')

ماتریس تبدیل Tرا، که توسط آن حالت‌های سیستم اصلی به حالت‌های سیستم کانونیکال مرتبط می‌شوند، نیز ارائه می‌دهد.

$$Z = T.X$$

چنانچه سیستم SYS به فرم فضای حالت نباشد، ماتریس T تهی خواهد بود.

روند انجام عملیات در این دستور به این صورت است که ابتدا چنانچه سیستم SYS به فرم تابع تبدیل یا قطب و صفر داده شده باشد با استفاده از دستور SS فرم فضای حالت آن به‌دست می‌آید. در تحقق modal، ماتریس p که مجموعه بردارهای ویژه ماتریس A است، محاسبه می‌شود و سپس با معادلات زیر تبدیل تشابهی $T=p^{-1}$ به سیستم اعمال شده و فرم کانونیکال به‌دست می‌آید.

$$\begin{cases} \dot{Z} = P^{-1}APZ + P^{-1}Bu \\ Y = CPZ + Du \end{cases}$$

به این ترتیب، ماتریس تبدیل حالت T که از نتایج اجرای دستورالعمل است همان P^{-1} خواهد بود. تحقق فرم COMPANION نیز با استفاده از تبدیل تشابهی از روی ماتریس کنترل پذیری به‌دست می‌آید.

مثال: سیستم زیر مفروض است:

$$\dot{X} = \begin{bmatrix} 1 & 2 & 0 \\ -1 & 5 & 6 \\ 3 & 0.5 & 2 \end{bmatrix} X + \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} U$$

$$Y = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} X$$

مقادیر ویژه ماتریس A عبارتند از:

```
Eig(A) =
        0.6715 + 2.5709i
        0.6715 - 2.5709i
        6.6571
```

برای به‌دست آوردن تحقق کانونیکال MODAL سیستم، برنامه زیر را داریم:

```
A=[1 2 0;-1 5 6;3 0.5 2];
B=[1;1;1];
C=[1 0 0];
D=0;
SYS=ss(A,B,C,D);
[CSYS,T]=canon(SYS,'modal');
[Ac,Bc,Cc,Dc]=ssdata(SYS);
```

که در آن Ac، Bc، Cc، وDc ماتریس‌های مربوط به تحقق کانونیکال و T تبدیل تشابهی مربوطه است. تحقق کانونیکال به صورت زیر خواهد بود:

```
T =
   -0.1097   -0.2167   -0.2792
   -0.4301    0.1507    0.0040
```

```
   -0.1089   -0.0801    0.3538
```

فرم مدال فضای حالت به صورت زیر خواهد بود:

```
a =
           x1       x2       x3
   x1   6.657        0        0
   x2       0   0.6715    2.571
   x3       0   -2.571   0.6715

b =
           u1
   x1  -0.6056
   x2  -0.2754
   x3   0.1648

c =
           x1       x2       x3
   y1  -1.013   -1.869   -0.778

d =
       u1
   y1   0
```

محدودیتها

۱. این دستور زمانی پاسخ صحیح می‌دهد که ماتریس A مقادیر ویژه تکراری نداشته باشد.

۲. در تحقق COMPANION سیستم باید از اولین ورودی کنترل پذیر باشد.

۲ – ۴ دستور ctrb

هدف: محاسبه ماتریس کنترل پذیری

Co=ctrb(A,B)
Co=ctrb(SYS)
Co=ctrb(SYS.a,SYS.b)

توضیح: چنانچه معادله حالت سیستم به فرم $\dot{X} = AX + Bu$ باشد، ماتریس کنترل پذیری سیستم (که با دستور ctrb محاسبه می‌شود) به صورت زیر خواهد بود:

$$C = \begin{bmatrix} B & AB & A^2B & .... & A^{n-1}B \end{bmatrix}$$

واضح است که سیستم، زمانی کنترل‌پذیر کامل است که رتبه ماتریس C کامل باشد.

مثال: ماتریس‌های A و B را به صورت زیر در نظر می‌گیریم:

$$A = \begin{pmatrix} 1 & 1 \\ 4 & -2 \end{pmatrix} \quad B = \begin{pmatrix} 1 & -1 \\ 1 & -1 \end{pmatrix}$$

برنامه زیر رتبه ماتریس کنترل پذیری را محاسبه می‌کند.

```
A=[1 1;4 -2];
B=[1 -1;1 -1];
C=ctrb(A,B)
Rank(C);
```

```
Co =
   1   -1    2   -2
   1   -1    2   -2
Rank = 1
```

چون رتبه C کامل نیست، بنابراین سیستم کنترل‌پذیر کامل نمی‌باشد. تعداد حالت‌های کنترل‌ناپذیر سیستم به صورت زیر محاسبه می‌شود:

```
unco=length(A)-rank(Co)
>> unco = 1
```

و از این‌رو سیستم یک حالت کنترل ناپذیر دارد.

۲ – ۵ دستور eig

هدف: یافتن مقادیر و بردارهای ویژه

```
d=eig(A)
d=eig(A,B)
[V,D]=eig(A)
[V,D]=eig(A,B)
```

توضیح: در آغاز به مرور پاره‌ای تعاریف مقدماتی می‌پردازیم. مساله مقادیر ویژه در واقع حل معادله زیر است:

$$AX = \lambda X$$

که در آن A یک ماتریس $n \times n$، x یک بردار n عضوی، و λ یک اسکالر است.

مقادیری از λ که معادله فوق را ارضاء می‌کنند مقادیر ویژه ماتریس A هستند. تعداد مقادیر ویژه برابر با تعداد ابعاد ماتریس A (یعنی n) است.

بردار X نیز به نام بردار ویژه سمت راست ماتریس A شناخته می‌شود. در نرم افزار MATLAB دستور eig برای محاسبه مقادیر و بردارهای ویژه در نظر گرفته شده است.

دستور d=eig(A) مقادیر ویژه ماتریس A را بر می‌گرداند.

دستور d=eig(A,B) که در آن A و B دو ماتریس مربعی هستند، مقادیر ویژه تعمیم یافته را محاسبه می‌کند.

[V,D]=eig(A) مقادیر ویژه ماتریس A را در روی ماتریس قطری D و بردارهای ویژه آن را در ماتریس V قرار می‌دهد.

بنابراین خواهیم داشت A*V=V*D که V، ماتریس بردارهای ویژه راست A است. ماتریس W ماتریس بردارهای چپ A نامیده می‌شود.

دستور [V,D]=eig(A,B) ماتریس قطری D شامل مقادیر ویژه تعمیم یافته و ماتریس V که ستون‌های آن بردارهای ویژه تعمیم یافته هستند را تولید می‌کند.

۲ – ۶ دستور estim

هدف: تعیین معادلات دینامیکی رویتگر با دریافت معادلات حالت سیستم و بهره رویتگر

est=estim(sys,L)
est=estim(sys,L,sensors,known)

توضیح: دستور est=estim(sys,L) با دریافت سیستم sys و بهره رویتگر L معادلات حالت رویتگر est را نشان می‌دهد. همه ورودی‌های سیستم u تصادفی و همه خروجی‌های y قابل اندازه گیری فرض می‌شوند. رویتگر est در فرم فضای حالت ارائه می‌شود. اگر سیستم اصلی sys دارای معادلات زیر باشد:

$$\dot{X} = AX + Bw$$
$$Y = CX + Dw$$

آنگاه دستور estim تخمین‌های خروجی و حالت سیستم $(\hat{x}, \hat{y})$ را با معادلات زیر تولید می‌کند:

$$\dot{\hat{x}}(t) = A\hat{x}(t) + L(y(t) - \hat{y}(t))$$
$$\hat{y}(t) = C\hat{x}(t)$$

دستور est=estim(sys,l,sensors,known) برای سیستم‌های عمومی‌تر به کار می‌رود. در اینجا فرض می‌شود که سیستم sys دارای ورودی‌های معلوم u و ورودیهای تصادفی w باشد. همچنین خروجی‌های سیستم نیز به دو گروه قابل اندازه گیری y و غیر قابل اندازه گیری z تقسیم می‌شوند.

معادلات سیستم به صورت زیر خواهد بود:

$$\dot{x} = Ax + B_1 w + B_2 u$$
$$\begin{bmatrix} z \\ y \end{bmatrix} = \begin{bmatrix} C_1 \\ C_2 \end{bmatrix} x + \begin{bmatrix} D_{11} \\ D_{21} \end{bmatrix} w + \begin{bmatrix} D_{12} \\ D_{22} \end{bmatrix} u$$

بردارهای اندیس sensors و known به ترتیب مشخص می‌کنند که کدام‌یک از خروجی ها قابل اندازه‌گیری هستند (y) و کدام‌یک از ورودی‌ها مقدار غیرتصادفی دارند (u). رویتگر est از ورودی‌های u و خروجی‌های y و تخمین حالت‌های سیستم استفاده می‌کند.

$$\dot{\hat{x}} = A\hat{x} + B_2 u + L(y - C_2\hat{x} - D_{22}u)$$
$$\begin{bmatrix} \hat{y} \\ \hat{x} \end{bmatrix} = \begin{bmatrix} C_2 \\ I \end{bmatrix} \hat{x} + \begin{bmatrix} D_{22} \\ 0 \end{bmatrix} u$$

شایان ذکر است که طراحی بهره رویتگر L با دستورات acker، kalman، یا place امکان‌پذیر است از سوی دیگر به منظور تضمین صحت تخمین، لازم است که دینامیک‌های رویتگر از دینامیک‌های سیستم اصلی سریعتر باشند.

۲ – ۷ دستور initial

هدف: محاسبه پاسخ سیستم به شرایط اولیه (پاسخ ورودی صفر[1])

```
initial(sys,x0)
initial(sys,x0,t)
initial(sys1,sys2,...,sysN,x0)
initial(sys1,sys2,...,sysN,x0,t)
[y,x,t]=initial(sys,x0)
```

توضیح: دستور initial پاسخ سیستمی را که هیچ ورودی خارجی به آن اعمال نمی‌شود به ازای شرایط اولیه در حالت‌های سیستم محاسبه می‌کند.

این تابع به مدلهای زمان پیوسته و گسسته قابل اعمال است. به جز در حالت [y,t,x]=initial(sys,xo) در سایر موارد اجرای دستور initial با رسم پاسخهای شرایط اولیه سیستم همراه است.

دستور initial(sys,x) پاسخ سیستم sys (که معادلات آن به فرم فضای حالت است) را به شرایط اولیه x0 رسم می‌کند. $x(t) = e^{A(t-t_0)}.x(t_0)$

سیستم sys می‌تواند پیوسته یا گسسته SISO یا MIMO با ورودی یا بدون ورودی باشد. بازه زمانی نمایش منحنی‌های شبیه‌سازی به طور خودکار چنان تعیین می‌شود که عملکرد پاسخ‌ها را در حالت گذرا به‌طورکامل نشان دهد.

نتیجه دستور initial(sys,xo,t) مانند حالت قبل است با این تفاوت که در این‌حالت بازه زمانی مطلوب برای محاسبه شرایط سیستم نیز مشخص شده است. می‌توان با انتخاب t، لحظه انتهای شبیه‌سازی را مشخص کرد یا t را به صورت بردار زیر تعریف نمود.

```
t=0:dt:tfinal
t=0:0.1:10;
```

در این حالت علاوه بر بازه زمانی شبیه‌سازی گام‌های شبیه‌سازی نیز مشخص شده است. در سیستم‌های گسسته dt باید با زمان نمونه برداری هماهنگ باشد. در سیستمهای پیوسته dt باید آنقدر کوچک اختیار شود تا رفتار حالت گذرا به خوبی مشخص گردد.

برای رسم همزمان پاسخ‌های چند سیستم LTI به یک شرایط اولیه دستورات زیر به کار می‌رود:

```
initial(sys1,sys2,...,sysN,x0)
initial(sys1,sys2,...,sysN,x0,t)
```

[1] Zero Input or Free Response

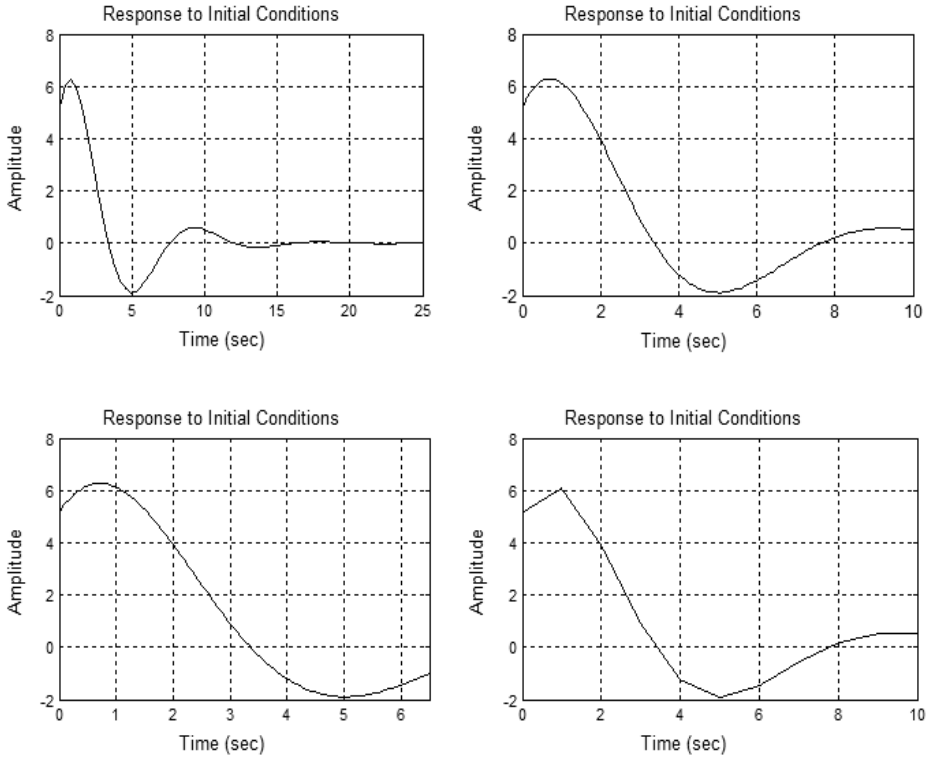
مثال: معادلات حالت سیستم را به فرم زیر در نظر می‌گیریم :

$$\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\end{bmatrix}=\begin{bmatrix}-0.557 & -0.781\\ 0.781 & 0\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix}$$

$$y=\begin{bmatrix}1.96 & 6.44\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix}$$

پاسخ سیستم با شرایط اولیه $X_0=[1;0.5]$ توسط برنامه زیر قابل حصول است.

```
A = [-0.5572 -0.7814;0.7814 0];
C = [1.9691 6.4493];
x0 = [1 ; 0.5];
t1=0:0.1:10;
t2=0:1:10;
sys = ss(A,[],C,[]);
subplot(2,2,1);initial(sys,x0);
subplot(2,2,2);initial(sys,x0,t1);
subplot(2,2,3);initial(sys,x0,6.5);
subplot(2,2,4);initial(sys,x0,t2);
```



شکل(۱-۱) پاسخ سیستم به شرایط اولیه

۲ – ۸ دستور Obsv

هدف: محاسبه ماتریس رویت پذیری

نگارش:

Ob=obsv(A,C)
Ob=obsv(sys)

توضیح: دستور obsv ماتریس رویت پذیری یک سیستم با نمایش فضای حالت را محاسبه می‌کند. اگر A یک ماتریس $n\times n$ و P یک ماتریس $p\times n$ باشد دستور Ob=obsv(A,C) ماتریس رویت‌پذیری زیر را نتیجه خواهد داد:

$$O=\left[C\ \ CA\ \ CA^2....\ CA^{n-1}\right]$$

واضح است که با دستور rank[obsv(A,C)] می‌توان رویت‌پذیری کامل سیستم را بررسی کرد. اگر SYS مدل فضای حالت یک سیستم باشد دستور obsv(sys) نیز ماتریس رویت پذیری سیستم را محاسبه خواهد کرد.

۲ – ۹ دستور obsvf

هدف: جدا کردن حالت‌های رویت ناپذیر و رویت پذیر سیستم

[Abar,Bbar,Cbar,T,k]=obsvf(A,B,C)

توضیح: معادلات حالت سیستم را به صورت زیر در نظر می‌گیریم:

$$\dot{X}=A_{n\times n}\,X+Bu$$
$$Y=C_{p\times n}\,X+Du$$

اگر رتبه ماتریس رویت پذیری سیستم r باشد و r < n باشد، آنگاه سیستم رویت‌پذیر کامل نیست. در این حالت تبدیل تشابهی T وجود دارد که حالت‌های رویت پذیر و رویت ناپذیر سیستم را از هم جدا می‌کند. با اعمال این تبدیل حالت‌های رویت ناپذیر سیستم در گوشه بالا و سمت چپ ماتریس A قرار می‌گیرند.

$$\overline{A}=\begin{bmatrix}A_{no} & A_{12}\\ 0 & A_o\end{bmatrix},\ \overline{B}=\begin{bmatrix}B_{no}\\ B_o\end{bmatrix},\ \overline{C}=\begin{bmatrix}0 & C_o\end{bmatrix}$$

به این ترتیب زوج [Abar,Bbar,Cbar,k]=obsvf(A,B,C) رویت پذیر کامل است و مقادیر ویژه A_{no} حالتهای رویت ناپذیر سیستم هستند.

مثال:

```
A=[0 1 0;0 2 1;0 -2 -1];
B=[1;2;1];
C=[0 0 1];
D=0;
```

```
Abar,Bbar,Cbar,T,k]=obsvf(A,B,C)
```

این سیستم یک مود رویت ناپذیر در صفر دارد.

۲ – ۱۰ دستور place

هدف: طراحی جایاب قطب برای سیستم‌های چند ورودی

```
K=place(A,B,P)
[K,prec,message]=place(A,B,P)
```

توضیح: سیستم چند ورودی (یا تک ورودی) زیر مفروض است:

$$\dot{X} = AX + Bu$$
$$Y = CX + Du$$

فرض می‌شود که بردار p حاوی محل قطب‌های مطلوب برای سیستم حلقه بسته باشد. دستورK=place(A,B,P) ماتریس فیدبک حالت k را چنان طراحی می‌کند که قانون U=-Kx قطب‌های سیستم حلقه بسته را در محل‌های مشخص شده در بردار p قرار دهد. واضح است که در سیستم‌های چند ورودی، ماتریس k منحصر به فرد نیست. دستور place از الگوریتم‌هایی استفاده می‌کند که حساسیت سیستم حلقه بسته را در برابر اغتشاشات در A وB کاهش می‌دهد.

دستور[K,prec,message]=place(A,B,P) علاوه بر تعیین k، prec را بر می‌گرداند که نشان می‌دهد که قطب‌های سیستم حلقه بسته تا چه اندازه به محل‌های تعیین شده در p نزدیک هستند. prec تعداد قطب هایی را که دقیقاً در محل‌های تعیین شده در p قرار گرفته اند نشان می‌دهد. بعلاوه، اگر یکی از قطب‌های سیستم حلقه بسته بیش از ۱۰٪ از محل‌های مطلوب دور باشد، یک پیغام هشدار در message صادر می‌شود.

با استفاده از دستور place می‌توان ماتریس بهره رویتگر را به صورت زیر محاسبه کرد.

```
L=place(A',C',P)'
```

۲ – ۱۱ دستور ss

هدف: تعریف مدل فضای حالت یا تبدیل یک مدل LTI به فضای حالت

```
sys=ss(A,B,C,D);
sys=ss(d);
sys=ss(A,B,C,D,ltisys);
sys=ss(A,B,C,D,'property1',value1,…,'propertyN',valueN);
sys_ss=ss(sys,'minimal')
```

توضیح: دستور SS برای نمایش سیستم به فرم فضای حالت یا تبدیل یک مدل تابع تبدیل یا مدل صفر-قطب-بهره به مدل فضای حالت به کار می‌رود.

دستورSYS=ss(A,B,C,D) مدل فضای حالت برای سیستم زمان پیوسته زیر را تولید می‌کند.

$$\dot{X} = AX + Bu$$
$$Y = CX + Du$$

اگر D=0، کافیست مقدار آن به صورت اسکالر صفر(بدون توجه به ابعاد آن) وارد شود.

دستورsys=ss(d) سیستم با ماتریس بهره ثابت d را نمایش می‌دهد و معادل است با:

```
sys=ss([],[],[],d);
```

دستور sys=ss(A,B,C,D,ltisys) یک مدل فضای حالت با توجه به ویژگی مشخص شده در LTI SYS تولید می‌کند. این ویژگی می‌تواند یکی از موارد ذکر شده در جدول (۱-۱) باشد.

جدول(۱-۱) ویژگی‌های سیستم LTI

نوع ویژگی	معرفی ویژگی	نام ویژگی
ماتریس	تأخیرهای ورودی و خروجی	ioDelayMatrix
بردار	تأخیر در ورودی	inputDelay
Cell array	گروه کانالهای ورودی	Input Group
Cell vector of string	اسامی کانالهای ورودی	InputName
Text	یادداشتهایی برای معرفی مدل	Notes
بردار	تأخیرهای خروجی	outputDelay
Cell array	گروه کانالهای خروجی	Output Group
Cell vector of string	اسامی کانالهای ورودی	InputName
اسکالر	زمان نمونه برداری	T_s
دلخواه	اطلاعات اضافی	userData

هریک از این ویژگی‌ها به اختصار توضیح داده می‌شود:

۱- زمان نمونه برداری T_s (برحسب ثانیه) برای تعریف مدل فضای حالت سیستمهای زمان-گسسته به کار می‌رود. T_s=0 سیستم زمان پیوسته و T_s= -1 معرف سیستم زمان گسسته با زمان نمونه‌برداری نامعلوم است.

۲- ویژگی IoDelayMatrix، OutputDelay ، و InputDelay امکان وارد کردن تأخیر در مدل‌های فضای حالت را ایجاد می‌کند و مقادیر پیش فرض این ویژگی‌ها صفر است (بدون تأخیر).

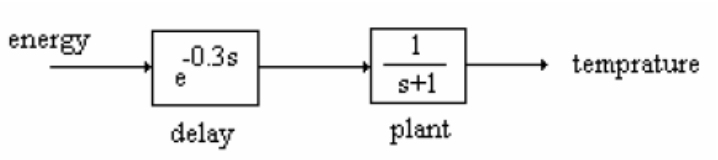
۳- ویژگی‌های OutputName وInputName امکان نام‌گذاری تک تک ورودی‌ها و خروجی‌ها را به کاربر می‌دهد.

۴- InputGroup و Output Group امکان گروه بندی ورودی‌ها و خروجی‌های سیستم را به برنامه‌نویس می‌دهد. به عنوان مثال در یک سیستم پنج ورودی برنامه نویس می‌تواند سه ورودی را در گروه Control و دو ورودی را در گروه Noise قرار دهد.

۵- ویژگی‌های Notes و Userdata نیز برای تعریف موارد اضافی توسط برنامه نویس در نظر گرفته شده‌اند. Notes تنها یک text بوده و معمولاً حاوی توضیحاتی درباره سیستم است. در ویژگی User data برنامه‌نویس می‌تواند داده‌های مورد نظر خود را تعریف کند.

دستور(sys_ss=ss(sys,'minimal' یک تحقق فضای حالت از سیستم sys را چنان ارائه می‌دهد که در آن هیچ حالت رویت ناپذیر یا کنترل ناپذیری موجود نباشد.

مثال۱: سیستم تک ورودی /تک خروجی تأخیردار زیر را در نظر می‌گیریم.



شکل(۱-۲) سیستم تک ورودی/تک خروجی

دستورات زیر plant را در فضای حالت معرفی می‌کند.

```
sys=tf(1,[1 1]);
sys_ss=ss(sys);
```

برای افزودن تأخیر به plant ویژگی input delay را مقدار دهی می‌کنیم. این‌کار با دستور set انجام می‌شود.

```
set(sys_ss,'InputDelay',0.3)
```

برای نام گذاری ورودی و خروجی سیستم و نیز افزودن یک یادداشت به آن دستور زیر را به کار می‌بریم.

```
set(sys_ss,'InputName','energy',Outputname','temprature','Not
es','A Sample Heater Mode')
```

به این ترتیب sys_ss در واقع مدل فضای حالت برای سیستم تأخیردار است و رسم پاسخ پله آن به خوبی 0.3 ثانیه زمان مرده را در ابتدای پاسخ مشخص خواهد کرد.

مثال۲: برنامه زیر علاوه بر قرار دادن مدل فضای حالت سیستم در متغیر system، حالتها و خروجیهای سیستم را نیز نام گذاری می‌کند.

```
A=[0 1;-1 -2];
B=[1;1];
C=[1 0;0 2];
D=[0;0];
system=ss(A,B,C,D,'statename',{'position';'velocity'},'Output
name',{'out1' ;'out2'});
```

۲ – ۱۲ دستور ssdata

هدف: دسترسی به ماتریسهای a،b،c، و d فضای حالت از روی مدل سیستم

[A,B,C,D]= ssdata(SYS) سیستم‌های زمان پیوسته

[A,B,C,D,Ts]=ssdata(SYS) سیستم‌های زمان گسسته

توضیح: چنانچه مدل سیستم به فرم تابع تبدیل یا قطب /صفر/بهره/باشد این دستور ابتدا آن را به مدل فضای حالت تبدیل کرده سپس A،B،C، وD را استخراج می‌کند.

۲ – ۱۳ دستور ss2ss

هدف: تبدیل محورهای حالت در نمایش فضای حالت (اعمال تبدیل تشابهی)

SYST=ss2ss(SYS,T)

توضیح: سیستم sys با نمایش فضای حالت زیر مفروض است:

$$\dot{X} = AX + Bu$$
$$Y = CX + Du$$

دستور ss2ss تبدیل تشابهی $Z = T^{-1}X$ را روی بردار x اعمال کرده و سیستم syst با مدل فضای حالت زیر را نتیجه می‌دهد:

$$\begin{cases} \dot{Z} = T^{-1}ATZ + T^{-1}Bu \\ Y = CTZ + Du \end{cases}$$

۲ – ۱۴ دستور ss2tf

هدف: تبدیل مدل فضای حالت سیستم به تابع تبدیل

[Num,Den]=ss2tf(A,B,C,D,iu)

توضیح: تابع تبدیل سیستم $\begin{cases} \dot{X} = AX + Bu \\ Y = CX + Du \end{cases}$ را نسبت به iuاُمین ورودی محاسبه می‌کند. محاسبه تابع تبدیل طبق رابطه زیر است:

$$G(s) = C(sI - A)^{-1}B + D$$

بردار Den حاوی ضرایب مخرج تابع تبدیل بر حسب توانهای نزولی s است. تعداد سطرهای ماتریس Num برابر با تعداد خروجی‌های سیستم بوده و هر سطر به صورت تابع تبدیل خروجی متناظر است.

۲ – ۱۵ دستور tf2ss

هدف: به‌دست‌آوردن مدل فضای حالت از روی تابع تبدیل سیستم

[A,B,C,D]=tf2ss(Num,Den)

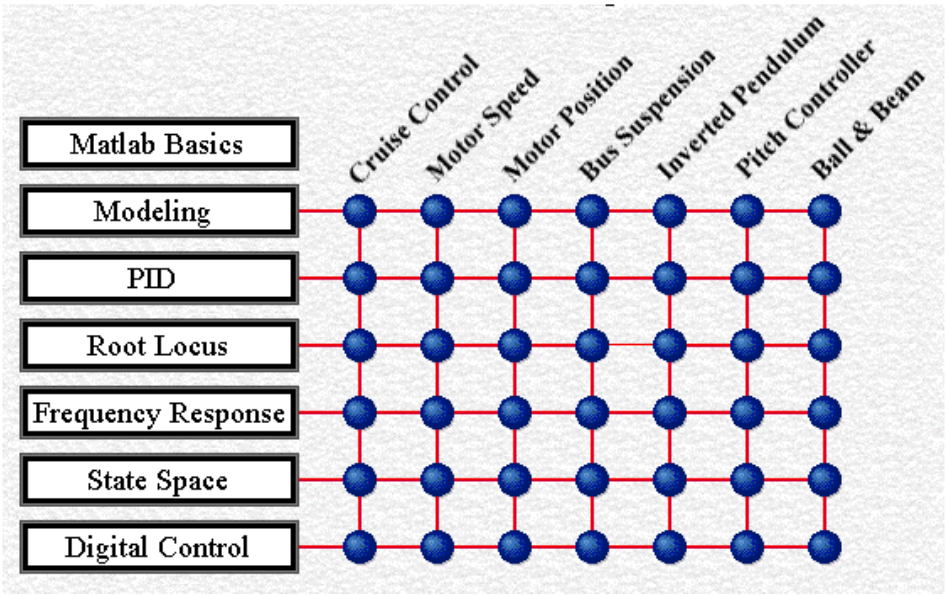
توضیح: تابع تبدیل سیستم را به صورت مدل فضای حالت زیر تبدیل می‌کند.

$$\dot{X} = AX + Bu$$
$$Y = CX + Du$$

مدل فضای حالت به فرم کانونیکال می‌باشد.

قسمت چهارم

- ✓ **پروژه اول** سیستم تثبیت سرعت خودرو
- ✓ **پروژه دوم** کنترل سرعت موتور DC
- ✓ **پروژه سوم** کنترل موقعیت موتور DC
- ✓ **پروژه چهارم** کنترل سیستم تعلیق خودرو
- ✓ **پروژه پنجم** سیستم کنترل و تعادل پاندول وارونه
- ✓ **پروژه ششم** سیستم کنترل موقعیت هواپیما در صفحه قائم
- ✓ **پروژه هفتم** سیستم کنترل موقعیت گوی بر روی تیر



بعد از مطالعه کامل دستورات MATLAB و آموزش طراحی سیستم‌های کنترلی، آمادگی لازم در طراحی کنترلر برای سیستم‌های صنعتی و کاربردی را خواهید داشت. این قسمت شامل هفت سیستم صنعتی است که هر یک با هفت روش و رویکرد کنترلی، طراحی و تحلیل می‌شوند، تا در پایان به یک پاسخ مناسب برسیم. می‌توان به جرأت گفت که در پایان انجام این ۷ پروژه کنترلی با ۷ روش مختلف (۴۹ پروژه کنترلی) قادر خواهید بود هر سیستم کنترلی دیگری را طراحی و تحلیل کنید .

پروژه اول

سیستم تثبیت سرعت خودرو (Cruise Control)

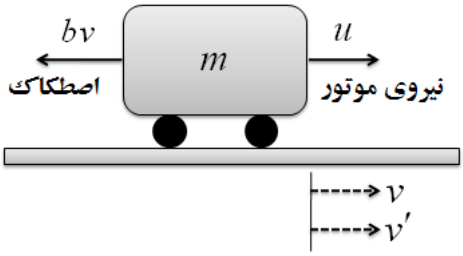
اهداف این پروژه

- مدل سازی کروز
- طراحی کنترلر PID
- رسم مکان هندسی ریشه‌ها (Root Locus)
- پاسخ فرکانسی
- طراحی فضای حالت
- طراحی کنترلر دیجیتالی
- طراحی کنترلرPI با سیمولینک

۱– ۱ مدل‌سازی

۱-۱-۱ مدل‌سازی سیستم براساس اصول فیزیکی و استخراج معادلات سیستم

مدل‌سازی سیستم کنترل کروز بسیار ساده می‌باشد. اگر در این مساله از اینرسی چرخ‌ها چشم‌پوشی کنیم و فرض کنیم که اصطکاک با سرعت رابطه مستقیم داشته و به بدنه با جرم m اعمال می‌شود، در آن صورت مساله به سیستم جرم و دمپر تبدیل می‌شود.



شکل (۱-۱) مدل ساده شده کروز

براساس قانون نیوتن معادله‌های سیستم را می‌توانیم به صورت زیر بنویسیم:

$$\sum F = mv' \Rightarrow u - bv = mv'$$
$$\begin{cases} mv' + bv = u \\ y = v \end{cases} \qquad (۱-۱)$$

در این معادلات، u نیروی موتور و y خروجی سیستم می‌باشد. در این مثال اجازه بدهید فرض کنیم:

$m = 1000 kg$

$u = 500 N$

$b = 50 N\,sec/m$

۱-۱-۲ تابع تبدیل سیستم

برای به‌دست آوردن تابع تبدیل سیستم، نیاز داریم از معادله (۱-۱) تبدیل لاپلاس بگیریم. توجه کنید زمانی که در حال پیدا کردن تابع تبدیل سیستم هستیم، شرایط اولیه را برابر صفر در نظر می‌گیریم.

$$\begin{cases} ms.V(s) + bV(s) = U(s) \\ Y(s) = V(s) \end{cases} \Rightarrow ms.Y(s) + b.Y(s) = U(s)$$
$$\Rightarrow \frac{Y(s)}{U(s)} = \frac{1}{ms + b} \qquad (۲-۱)$$

معادله حاضر، یک معادله مرتبه اول است.

۱-۱-۳ مدل فضای حالت سیستم

می‌توان معادله مرتبه اول 1^{st} را به فرم فضای حالت نوشت:

$$\begin{cases} x' = Ax + Bu \\ y = Cx \end{cases} \Rightarrow \begin{cases} v' = [-\frac{b}{m}]v + [\frac{1}{m}]u \\ y = [1]v \end{cases} \qquad (۳-۱)$$

۱-۱-۴ قیدهای حاکم بر طراحی

قدم بعدی در مدل‌سازی سیستم این است که بدانیم این سیستم قرار است در انتها چه شرایطی داشته باشد؛ یعنی قیدهای حاکم بر سیستم باید مشخص شوند.

- وقتی که نیروی موتور 500N است، حداکثر سرعت باید برابر 10m/s باشد.
- جسم متحرک باید قادر باشدکمتر از 5 ثانیه شتاب بگیرد و به این سرعت برسد پس زمان خیز[1] باید کمتر از 5 ثانیه باشد ($Rise time < 5 \sec$).
- در این سیستم، جهش[2] حدود 10% آسیب جدی به سیستم اعمال نمی‌کند ($Overshoot < 10\%$).
- حدود 2% خطای حالت ماندگار برای این سیستم قابل قبول است($Steady State error < 2\%$).

۱-۱-۵ پاسخ سیستم حلقه باز

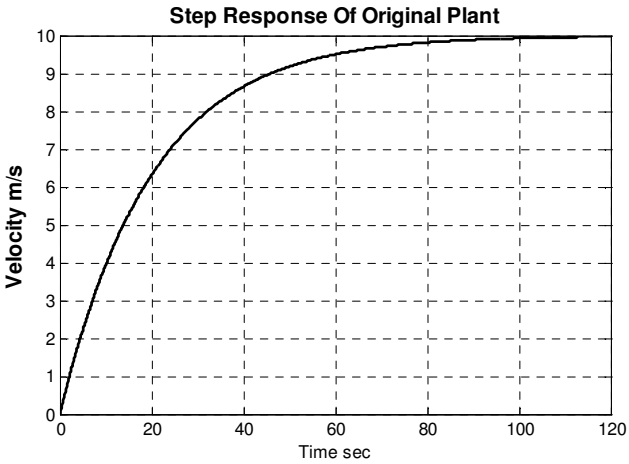
حال بهتر است اجازه دهید پاسخ سیستم حلقه باز را به ورودی پله بررسی کنیم. دستورات زیر را در یک m-file واردکنید:

```
        %==========  Open loop system  ==============
m=1000;b=50;u=500;
```

[1] Rise time
[2] Overshoot

```
num=[1];
den=[m b];
sys=tf(num,den)   % Calculate transfer function
step(u*sys)           % Calculate step response
ylabel('Velocity m/s'), xlabel('Time sec')
title('Step Response Of Original Plant')
```

با اجرای برنامه، نمودار(۱-۲) به‌دست می‌آید. از روی شکل مشخص است که جسم متحرک در مدت 120 ثانیه به سرعت 10m/s می‌رسد و این موضوع، معیار کمتر از 5 ثانیه بودن زمان خیز را برآورده نمی‌کند.



شکل (۱-۲) پاسخ سیستم به ورودی پله

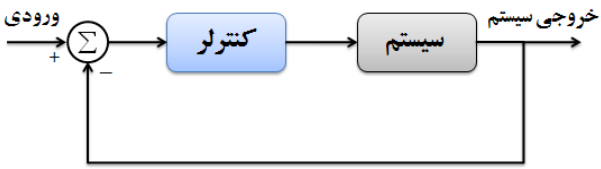
۱- ۲ طراحی کنترلر PID

از معادله (۱-۲) تابع تبدیل سیستم به صورت زیر به‌دست می‌آید:

$$\frac{Y(s)}{U(s)}=\frac{1}{ms+b} \qquad (۱-۴)$$

بلوک نمودار سیستم کنترلی با فیدبک واحد در شکل (۱-۳) نمایش داده شده است. می‌دانیم تابع تبدیل کنترلر PID به صورت زیر است:

$$PID\ Controller:\ k_p+k_D s+\frac{k_I}{s}=\frac{k_D s^2+k_p s+k_I}{s} \qquad (۱-۵)$$



شکل (۱- ۳) بلوک نمودار سیستم کنترلی با فیدبک واحد

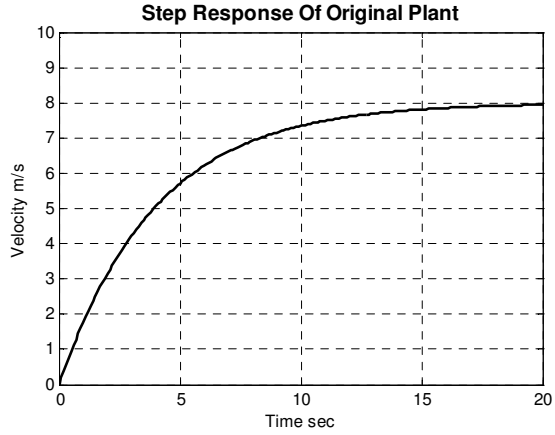
از مبحث کنترل PID به یاد داریم که با افزایش بهره تناسبی می‌توان زمان خیز را کاهش داد، بنابراین در ابتدا یک کنترلر تناسبی[1] قرار می‌دهیم. اولین‌کاری که باید انجام دهیم این است که تابع تبدیل سیستم حلقه بسته با کنترلر تناسبی(k_p) را به‌دست آوریم. با کاهش بلوک نمودار بالا، تابع تبدیل سیستم حلقه بسته با کنترلر تناسبی به صورت زیر به‌دست می‌آید:

$$\frac{Y(s)}{U(s)}=\frac{G}{1+GH}=\frac{\frac{k_p}{ms+b}}{1+\frac{k_p}{ms+b}}=\frac{k_p}{ms+(b+k_p)} \qquad (۶-۱)$$

با توجه به شکل (۱-۲) نیاز داریم که زمان خیز را کاهش دهیم. پس با افزودن کنترلر تناسبی، زمان خیز را کاهش می‌دهیم. در معادله (۱-۶) مقدار $k_p = 200$ را قرار داده و دستورات زیررا به ادامه m-file اضافه کنید.

```
kp=200; u=10;
num1=[kp];den1=[m b+kp];
t=0:0.1:20;
sys2=tf(num1,den1);
step(u*sys2)
axis([0 20 0 10])
```

با اجرای m-file خروجی زیر به‌دست می‌آید.



شکل (۱-۴) پاسخ سیستم در حضور کنترل تناسبی

از روی شکل (۱-۴) مشخص است که خروجی حاضر دوباره معیارهای طراحی مثل زمان خیز و خطای ماندگار را برآورده نمی‌کند. با افزایش $k_p = 10000$ می‌توان زمان خیز را به 0.5 ثانیه نیزکاهش داد ولی این مقدار اصلاً واقعی نیست، یعنی موتور نمی‌تواند در مدت 0.5 ثانیه از سرعت 0 به سرعت 10m/s برسد.

توجه: برای پیدا کردن پاسخ سیستم حلقه بسته می‌توان به طور مستقیم از تابع تبدیل سیستم حلقه باز استفاده کرد. دستور cloop این کار را انجام می‌دهد.

[1] Proportional

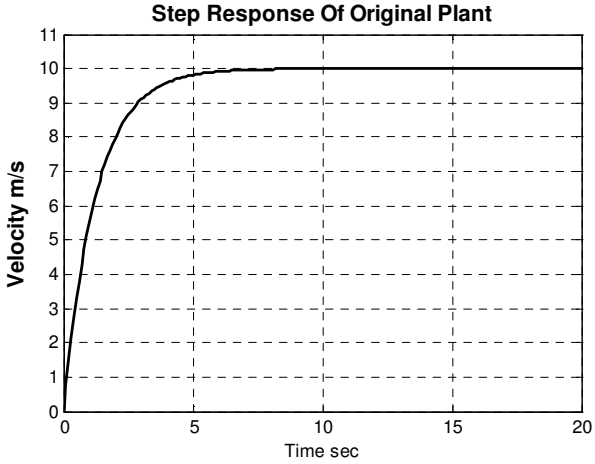
```
kp=200;
num=[1];
den=[m b];
[numc,denc]=cloop(kp*num,den,-1);% Calculate close loop
response
t = 0:0.1:20;
sys3=tf(numc,denc);
figure(3)
step (u*sys3)
```

در هر دو صورت به یک جواب خواهیم رسید. برای حذف خطای حالت ماندگار از کنترلر انتگرالی I استفاده می‌کنیم. تابع تبدیل سیستم حلقه بسته با کنترلر تناسبی– انتگرالی (PI) به صورت زیر خواهد بود:

$$\frac{Y(s)}{U(s)}=\frac{G}{1+GH}=\frac{\dfrac{k_p+\dfrac{k_I}{s}}{ms+b}}{1+\dfrac{k_p+\dfrac{k_I}{s}}{ms+b}}=\frac{k_p s+k_I}{ms^2+(b+k_p)s+k_I} \qquad (۷-۱)$$

با تنظیم بهره انتگرالی و تناسبی $k_p = 800, k_I = 40$ معیارهای طراحی مورد نیاز به‌دست می‌آید.

```
%======= PI Design ==========
kp = 800; ki = 40;
num3=[kp ki];
den3=[m b+kp ki];
sys3=tf(num3,den3);
figure(4)
step(u*sys3);
axis([0 20 0 11]);
```

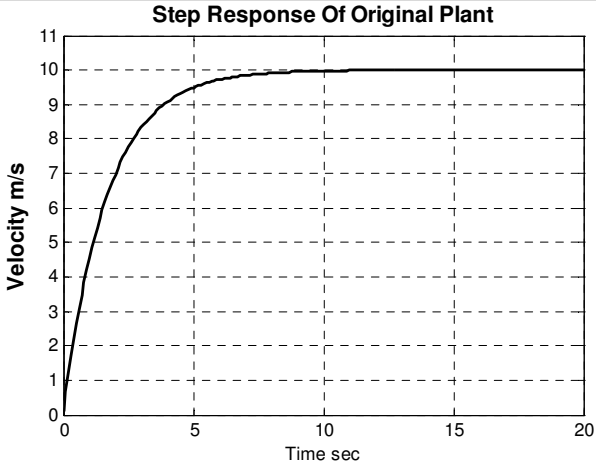


شکل (۱-۵) پاسخ پله در حضور کنترلر PI

در این مثال خاص، نیازی به استفاده از کنترلر PID نمی‌باشد ولی به عنوان تمرین تابع تبدیل سیستم حلقه بسته با کنترلر PID را نیز به‌دست می‌آوریم.

$$\frac{Y(s)}{U(s)}=\frac{G}{1+GH}=\frac{\dfrac{k_p+\dfrac{k_I}{s}+k_Ds}{ms+b}}{1+\dfrac{k_p+\dfrac{k_I}{s}+k_Ds}{ms+b}}=\frac{k_Ds^2+k_ps+k_I}{(m+k_D)s^2+(b+k_p)s+k_I} \qquad (۸-۱)$$

```
%========== PID Design ========
kp=600;ki=30;kd=10;
num4=[kd kp ki];
den4=[m+kd b+kp ki];
sys4=tf(num4,den4);
figure(5)
step(u*sys4)
axis([0 20 0 11])
```



شکل (۱- ۶) پاسخ پله در حضور کنترلر PID

۱– ۳ رسم مکان هندسی ریشه‌ها (Root Locus)

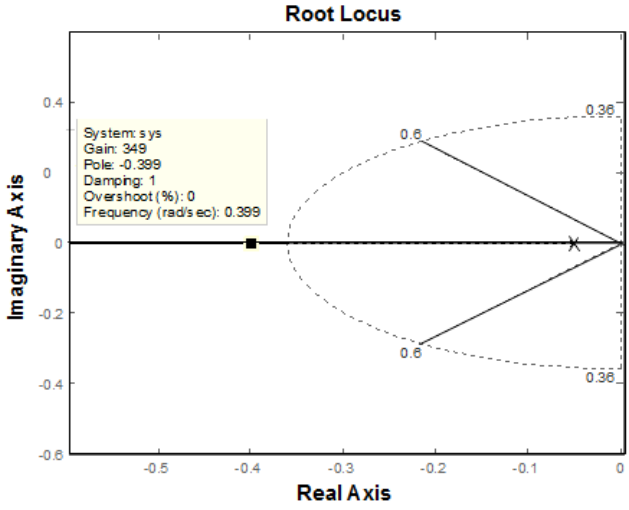
حال می‌خواهیم مساله کنترل کروز را با استفاده از مکان هندسی ریشه‌ها انجام دهیم. همانطور که می‌دانیم مکان هندسی ریشه‌ها موقعیت قطب‌ها را، وقتی‌که بهره k از صفر تا بی‌نهایت تغییر می‌کند، نشان می‌دهد. بنابراین یک کنترلر تناسبی K_P برای حل این مساله در نظر گرفته، تابع تبدیل سیستم حلقه بسته را به صورت زیر به‌دست می‌آوریم:

$$\frac{Y(s)}{U(s)}=\frac{1}{ms+b}\Rightarrow\frac{Y(s)}{U(s)_{closeloop}}=\frac{K_p}{ms+(b+K_p)} \qquad (۹-۱)$$

با استفاده از دستور rlocus و sgrid می‌توان مکان هندسی ریشه‌ها را رسم کرد. در ابتدا نیاز داریم که مقادیر ω_n, ζ مشخص شوند. برای به‌دست آوردن آنها می‌توانیم از فرمول‌های $M_p = e^{-\frac{\pi\zeta}{\sqrt{1-\zeta^2}}}$, $\omega_n = \frac{1.8}{T_r}$ استفاده کنیم. بنابراین باید $\zeta > 0.6$ تا جهش کمتر از 10% و $\omega_n > 0.36$ تا زمان خیز کمتر از 5 ثانیه گردد. حال برای رسم مکان هندسی ریشه‌ها فرامین زیر را به ادامه m-file اضافه کنید:

```
m = 1000;b = 50;u = 10;
numo=[1];deno=[m b];
figure(1)
axis([-0.6 0 -0.6 0.6]);
rlocus (numo,deno)
sgrid(0.6, 0.36)    % Generate s-plane grid of
[Kp, poles]=rlocfind(numo,deno)
figure(2)
numc=[Kp];
denc=[m (b+Kp)];
t=0:0.1:20;
step (u*numc,denc,t)
axis ([0 20 0 10])
```

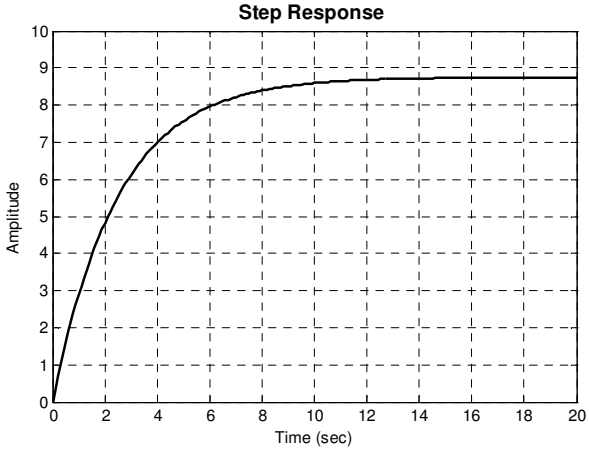
با اجرای دستورات، خروجی زیر حاصل می‌شود.



شکل (۱- ۷) مکان هندسی ریشه‌ها

خط‌چین، نشان دهنده مکان ثابت میرایی $\zeta = 0.6$ است، به‌گونه‌ای که ضریب میرایی بیشتر از 0.6 بین دو خط، و ضریب میرایی کمتر از 0.6 خارج این دو خط قرار می‌گیرد. نیم بیضی نیز نشان دهنده فرکانس طبیعی ثابت $\omega_n = 0.36\ rad/\sec$ است به گونه‌ای که فرکانس طبیعی بیشتر از 0.36 خارج نیم بیضی و فرکانس کمتر

از 0.36 داخل بیضی قرار می‌گیرد. اگر به پنجره فرامین MATLAB توجه کنید می‌بینید که از شما می‌خواهد که یک نقطه روی مکان هندسی ریشه‌ها انتخاب کنید؛ بنابراین چون ما می‌خواهیم $\zeta > 0.6$, $\omega_n > 0.36\ rad/\sec$، پس روی مکان هندسی ریشه‌ها روی 0.4-کلیک کنید. بنابراین مقدار بهره Kp و موقعیت قطب‌ها در پنجره دستورات MATLAB ظاهر شده و پاسخ پله نیز به صورت زیر خواهد بود.



شکل (۱- ۸) پاسخ پله سیستم حلقه بسته

```
selected_point =
  -0.4001 - 0.0000i
```

همانطور که از شکل مشخص است، خطای حالت ماندگار حدود 10% است. برای حل این مشکل از کنترلر lag استفاده می‌کنیم.

۱-۳-۱ طراحی کنترلر Lag

برای کاهش خطای حالت ماندگار از یک جبران ساز Lag استفاده می‌کنیم. تابع تبدیل Lag به صورت زیر می‌باشد:

$$G_C(s) = \frac{s + z_0}{s + p_0} \qquad (۱۰-۱)$$

تابع تبدیل سیستم حلقه باز بدون Kp به صورت زیر می‌باشد:

$$G_{OL}(s) = \frac{s + z_0}{(ms + b)(s + p_0)} \qquad (۱۱-۱)$$

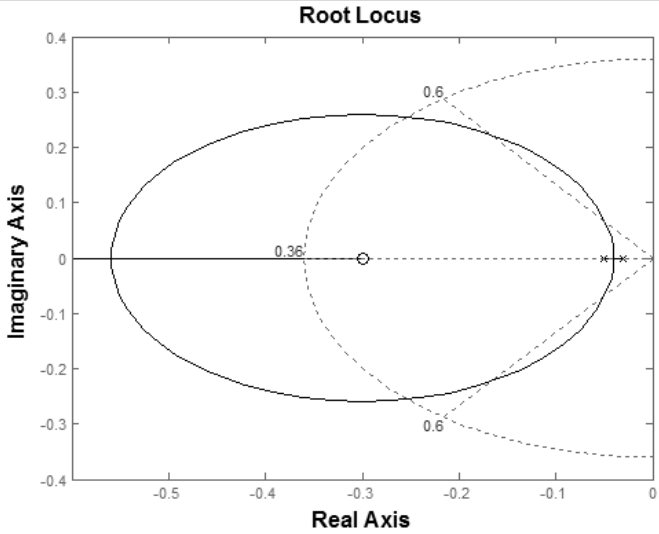
تابع تبدیل سیستم حلقه بسته با فیدبک واحد با بهره K_p نیز به صورت زیر به‌دست می‌آید:

$$G_{CL}(s) = \frac{K_p.s + K_p.z_0}{ms^2 + (b + mp_0 + K_p)s + bp_0 + K_p z_o} \qquad (۱۲-۱)$$

بزرگی Z_0 بزرگتر از p_0 است. جبران کننده Lag باعث کشیده شدن مکان هندسی ریشه‌ها به سمت نیم صفحه راست می‌گردد که نتیجه‌ای نامطلوب است. بنابراین صفر و قطب باید نزدیک هم باشند که معمولاً نزدیک مبدا در نظر گرفته می‌شود. بنابراین خطای حالت ماندگار به نسبت $\frac{z_0}{p_0}$ کاهش پیدا می‌کند.

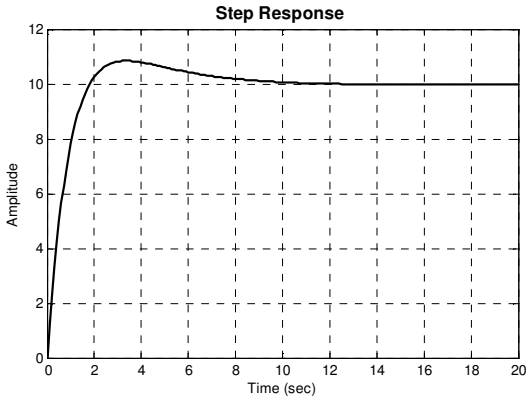
بنابراین اجازه بدهید که $z_0 = -0.3, p_0 = -0.03$ را انتخاب کنیم. درادامه، دستورات زیر را به ادامه m-file اضافه کنید. بعد از اجرای برنامه نقطه 0.4- را روی محور حقیقی انتخاب کنید.

```
m = 1000;b = 50; u = 10;
Zo=0.3; Po=0.03;
numo=[1 Zo];
deno=[m b+m*Po b*Po];
figure(1)
hold on
axis ([-0.6 0 -0.4 0.4])
rlocus(numo,deno)
sgrid(0.6,0.36)
[Kp, poles]=rlocfind(numo,deno)
figure(2)
t=0:0.1:20;
numc=[Kp Kp*Zo];
denc=[m b+m*Po+Kp b*Po+Kp*Zo];
axis ([0 20 0 12])
step (u*numc,denc,t)
```



شکل (۱-۹) نمودار مکان هندسی ریشه‌ها

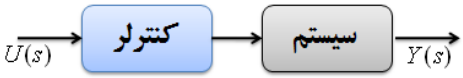
پاسخ زیر حاصل می‌شود؛ خطای حالت ماندگار برابر صفر بوده و جواب مطلوب به‌دست می‌آید.



شکل (۱-۱۰) پاسخ پله در حضور کنترلر Lag

۱- ۴ پاسخ فرکانسی

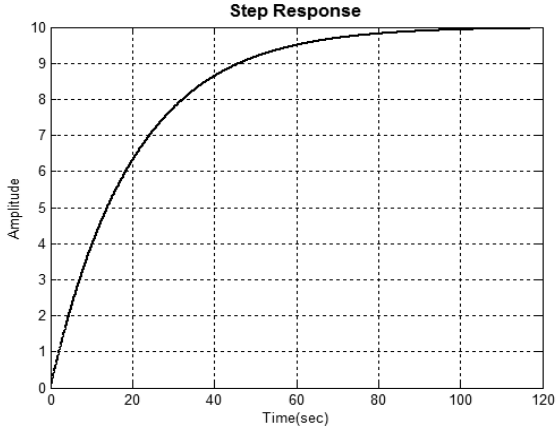
اولین قدم در استفاده از روش پاسخ فرکانسی این است که تعیین کنیم از چه تابع تبدیل حلقه بازی می‌خواهیم استفاده کنیم. مشابه روش مکان هندسی ریشه‌ها از یک کنترلر تناسبی استفاده می‌کنیم.



به منظور کشیدن نمودار بود [1] پاسخ حلقه باز باید پایدار باشد. اکنون بررسی می‌کنیم که پاسخ حلقه باز به چه صورت خواهد بود. پس به ادامه m-file دستورات زیر را اضافه کنید:

```
m = 1000;b = 50;u = 500; Kp=1;
numo=[Kp];deno=[m b];
step(u*numo,deno)
```

بعد از اجرای برنامه پاسخ زیر به‌دست می‌آید.

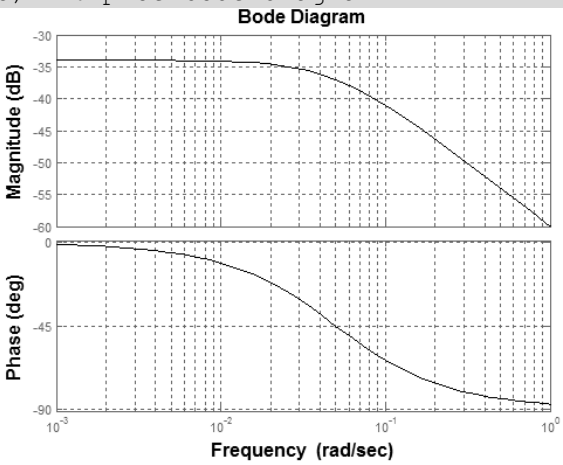


شکل (۱-۱۱) نمودار پاسخ پله

[1] Bode

همانطور که ملاحظه می‌شود پاسخ حلقه باز پایدار است. اکنون خط آخر برنامه را پاک کنید و دستور زیر را اضافه نمائید:

```
bode(numo,deno)    % plot bode diagram
```



شکل (۱-۱۲) نمودار بود برای سیستم حلقه باز

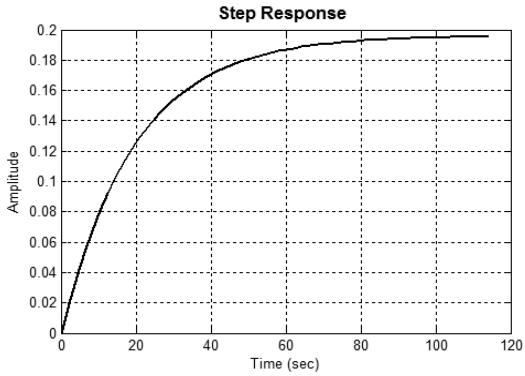
۱-۴-۱ کنترلر تناسبی

در فرکانس $-7.5dB$ ~ -6 پهنای باند فرکانسی برابر ω_n است. می‌دانیم که نسبت میرایی تقریباً برابر $\zeta = PM/100$ است (بر حسب درجه) و در شکل بالا $PM = 155\deg$، پس $\zeta = 1.55 \rightarrow overdamped$.

و خطای حالت ماندگار از رابطه زیر به‌دست می‌آید:

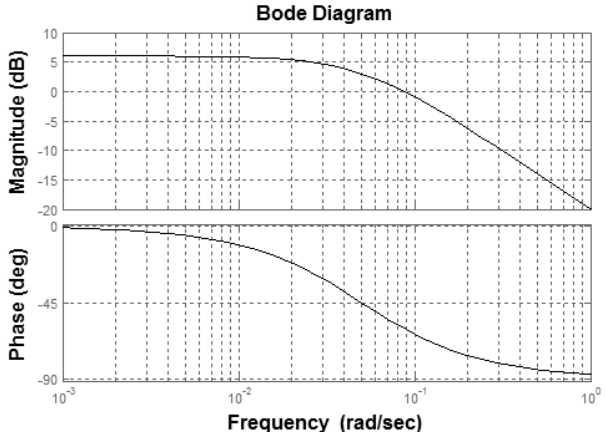
$$Steady\ state\ error = \frac{1}{1+Magnitude} \times 100$$
$$Magnitude = 10^{\frac{M(dB)}{20}} \qquad (۱-۱۳)$$

در این سیستم از آنجا که بهره فرکانس پایین برابر 35dB- است پس خطای حالت ماندگار برابر 98% به‌دست می‌آید. از طرفی با توجه به اینکه نسبت میرایی بزرگتر از 1 است پس سیستم میرای فوق بحرانی است. این مطلب در شکل (۱-۱۳) نمایش داده شده است.



شکل (۱-۱۳) پاسخ پله سیستم حلقه بسته

اگر بتوانیم بهره را به سمت بالا شیفت دهیم تا اینکه هم پهنای باند فرکانسی و هم بهره فرکانس پایین کاهش پیدا کند، در آن صورت، هم زمان خیز و هم خطای حالت ماندگار بهبود پیدا می‌کند. ما این کار را با افزایش دادن بهره Kp انجام می‌دهیم. به طور مثال، مقدار آن را برابر 100 می‌دهیم تا ببینیم چه اتفاقی می‌افتد. نمودار بود (۱-۱۴) حاصل می‌شود: `Kp=100;`

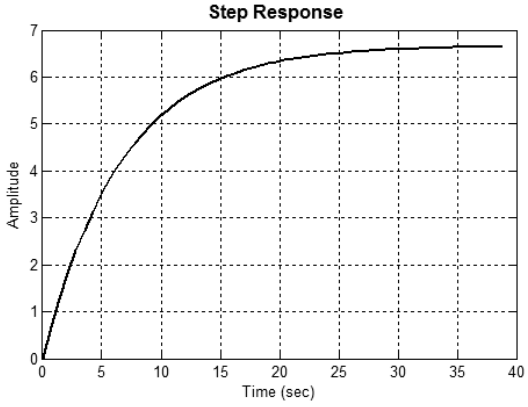


شکل (۱-۱۴) نمودار بود با افزایش بهره

ملاحظه می‌شود که بهره فرکانس پایین حدود 6dB (بزرگی 2) است که این امر نشان دهنده خطای حالت ماندگار 33% است. از طرفی پهنای باند فرکانسی حدود 0.1rad/sec است (با دستور Bandwidth(sys) می‌توان پهنای باند را محاسبه کرد) که نشان دهنده زمان خیز 18 ثانیه است.

```
sys_cl = feedback(sys,1);
step(u*sys_cl)
```

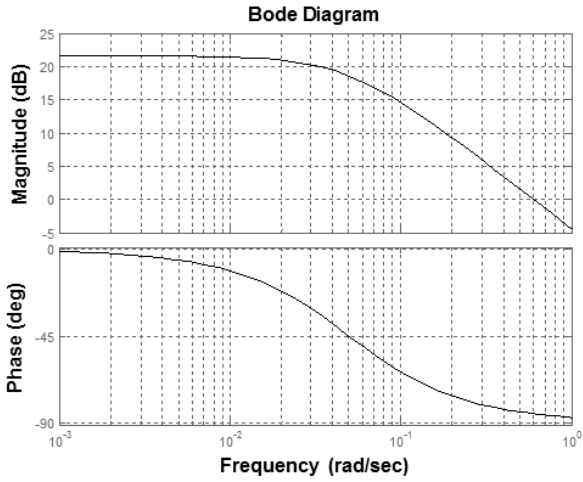
حال پاسخ سیستم حلقه بسته را بررسی می‌کنیم. پیش بینی می‌کنیم که با افزایش Kp هم خطای حالت ماندگار و هم زمان خیز بهبود پیدا می‌کند.



شکل (۱-۱۵) پاسخ پله سیستم حلقه بسته

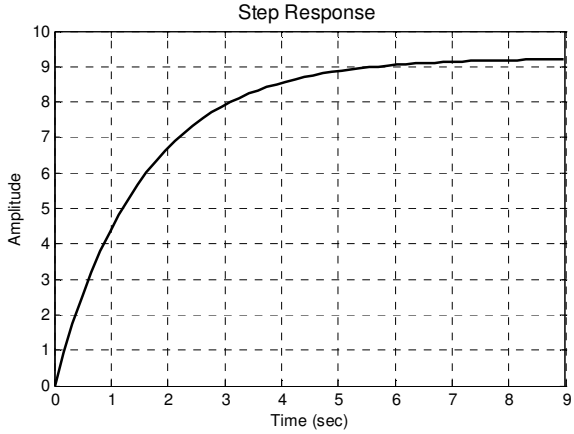
نکته خیلی مهم: پهنای باند برای سیستم حلقه بسته محاسبه می‌شود و برابر اولین فرکانسی است که بهره سیستم به اندازه 70.79%یا 3dB- کمتر از بهره DC آن می‌شود. این مقدار برای سیستم مرتبه اول حدود 1.5 برابر فرکانس گذر از 0 dB می‌باشد. حال دوباره Kp را افزایش می‌دهیم.

```
Kp=600;
numo=[Kp/b];deno=[m/b 1];
bode(numo,deno)
```



شکل (۱-۱۶) نمودار بود با افزایش بهره

در این حالت بهره فرکانس پایین حدود 20dB است (بزرگی 10)، پس خطای حالت ماندگار حدود 9% شده و پهنای باند فرکانسی حدود 0.6 می‌شود. از قبل پیش بینی می‌کنیم که زمان خیز حدود 3 ثانیه خواهد بود، که به مقدار مطلوب خواهد رسید. حال جواب حلقه بسته را ببینیم:



شکل (۱-۱۷) پاسخ پله سیستم حلقه بسته

سرانجام خطای حالت ماندگار به مقدار مطلوب خود می‌رسد ولی با افزایش بیش از حد بهره K، زمان خیز خیلی پایین می‌آید و تا حدودی غیرواقعی می‌شود. پس از یک کنترلر lag استفاده می‌کنیم تا خطای حالت ماندگار را اصلاح کند.

۱-۴-۲ استفاده از کنترلر Lag

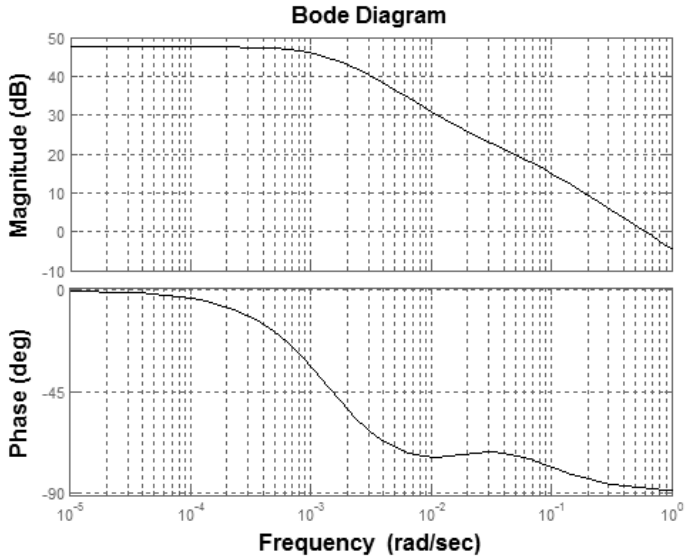
تاثیر اصلی کنترل lag در نمودار بزرگی قابل نمایش است. این کنترلر بهره را در فرکانس‌های پائین اضافه می‌کند. بزرگی این بهره برابر a است و تاثیر این بهره باعث کاهش خطای حالت ماندگار سیستم حلقه بسته با فاکتور a می‌شود؛ زیرا بهره کنترلر در فرکانس‌های بالا و وسط برابر 1 است. بسته به مقدار a، مقادیر مختلف فازی بزرگتر از 90- را می‌توان به سیستم اضافه کرد.

کنترلر Lag بهره فرکانس پایین را کاهش می‌دهد ولی موقعیت پهنای باند فرکانسی را تغییر نمی‌دهد. آنچه مورد نظر است این است که بهره فرکانس پایین، افزایش یابد تا خطای حالت ماندگار را کاهش داده و موقعیت پهنای باند فرکانسی را تغییر ندهد تا همان مقدار زمان خیز را داشته باشیم. تابع تبدیل Lag به صورت زیر است:

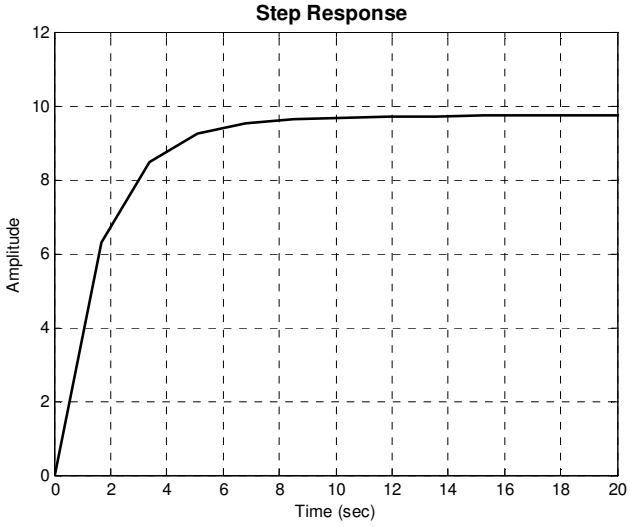
$$G(s)=\frac{1}{a}\left(\frac{1+aTs}{1+Ts}\right) \qquad (۱۴-۱)$$

ما نیاز داریم که مقادیر a,T را تعیین کنیم. خطای حالت ماندگار با افزایش فاکتور a کاهش پیدا می‌کند. از طرفی مقدار T باید طوری انتخاب شود که فرکانس‌های دو گوشه به هم خیلی نزدیک نباشند چون جواب حالت گذرا خیلی نامطلوب می‌شود. حال اجازه دهید که $a=0.05, T=700$ را انتخاب کنیم. فرامین زیر را به ادامه m-file اضافه کنید:

```
m = 1000; b = 50; u = 10;
Kp=600;
numo=[Kp/b];
deno=[m/b 1];
a = 0.05; T=700;
numlag = [a*T 1];
denlag = a*[T 1];
newnum = conv(numo,numlag);
newden = conv(deno,denlag);

bode(newnum,newden)
figure
[numc,denc]=cloop(newnum,newden);
step (u*numc,denc)
```

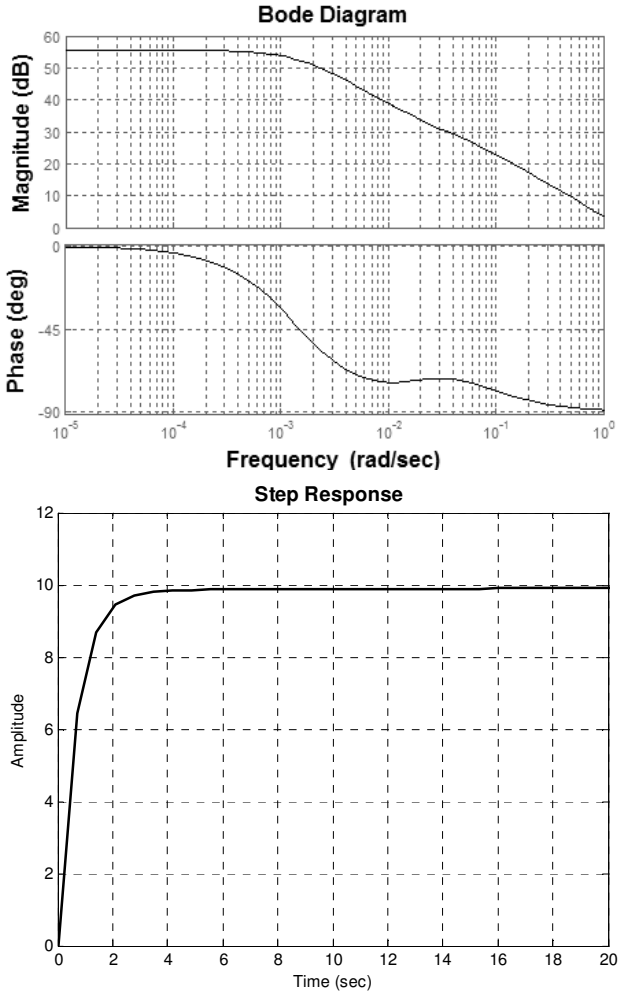


شکل (۱–۱۸) نمودار بود با کنترلر Lag



شکل (۱–۱۹) پاسخ پله سیستم حلقه بسته

همانطور که می‌بینیم خطای حالت ماندگاربه شدت کاهش پیدا می‌کند ولی زمان نشست بیشتر می‌شود. بنابراین با سعی و خطا مقدار بهره را برابر 1500 قرار می‌دهیم و نمودار زیر حاصل می‌شود.



شکل (۱-۲۰) نمودار بود و پاسخ پله سیستم حلقه بسته

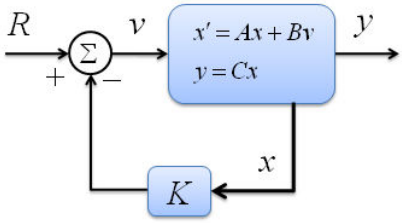
همانطور که از شکل بر می‌آید جهش صفر و خطای حالت ماندگار حدود صفر و زمان خیز حدود 2 ثانیه است و زمان نشست کمتر از 3.5 ثانیه خواهد شد.

۱– ۵ طراحی فضای حالت

همانطور که قبلاً گفته شده بود معادله فضای حالت سیستم به صورت زیر می‌باشد:

$$\begin{cases} v' = [-\frac{b}{m}]v + [\frac{1}{m}]u \\ y = [I]v \end{cases} \qquad (۱-۱۵)$$

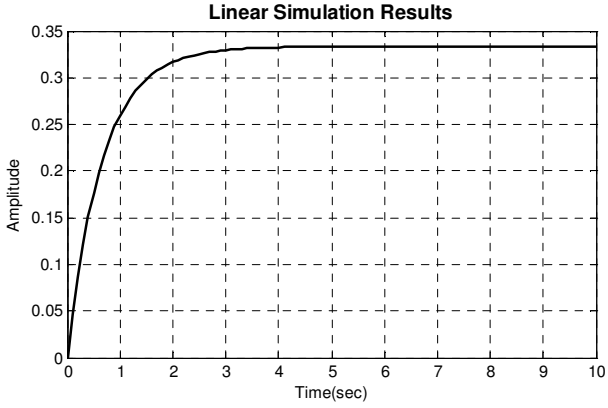
حال می‌خواهیم با استفاده از روش جایابی قطب‌ها سیستم را کنترل کنیم. شکل (۱-۲۱) کنترل به روش فیدبک تمام حالت‌ها [1] را نمایش می‌دهد.



شکل (۱-۲۱) کنترل به روش فیدبک کلیه متغیرهای حالت

در شکل (۱-۲۱) R ورودی رفرنس، K ماتریس کنترل، و $U = -K.X$ ورودی سیستم می‌باشد. از طرفی می دانیم که برای به‌دست آوردن خروجی دلخواه باید مکان قطب‌های مطلوب را تعیین کنیم. محل قطب‌ها نیز از معادله مشخصه $|sI-(A-BK)|$ به‌دست می‌آید. دراینجا، محل قطب‌های مطلوب را خودمان تعیین کرده، سپس با استفاده از دستورات MATLAB ماتریسK را به‌دست می‌آوریم. از آنجایی که ماتریس A یک در یک است پس فقط یک قطب داریم بنابراین به صورت دلخواه محل قطب را در -1.5 قرار می‌دهیم. سپس با استفاده از دستور place مقدار ماتریس K را به‌دست می‌آوریم. دستورات زیر را به ادامه m-file اضافه کنید:

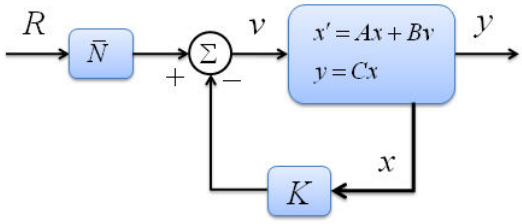
```
m=1000;b=50;
t=0:0.1:10;
u=500*ones(size(t));
A=[-b/m];B=[1/m];C=[1];D=[0];
x0=[0]; p1=-1.5;
K=place(A,B,[p1]);
A1=A-B*K;
lsim(A1,B,C,D,u,t,x0);
```



شکل (۱-۲۲) پاسخ پله سیستم حلقه بسته

[1] Full state feedback

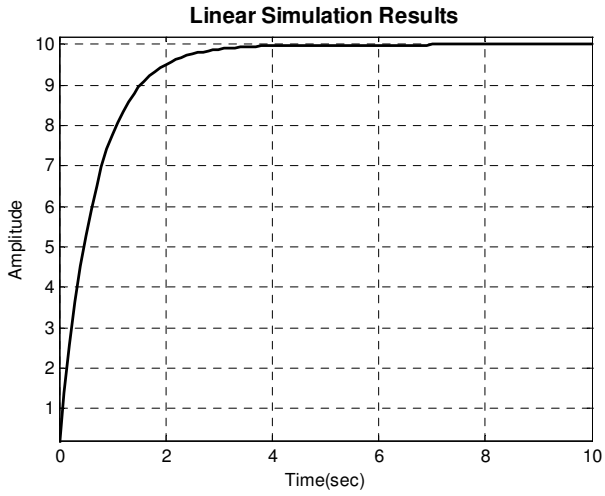
برای حذف خطای ماندگار، از یک ورودی مرجع که در یک ضریب ثابت ضرب می‌شود استفاده می‌کنیم. درشکل شماتیک بالا خروجی سیستم با ورودی مرجع مقایسه نشد، در عوض تمام حالت‌ها اندازه‌گیری گردید و در بردار K ضرب شد و نتیجهٔ به دست آمده را با مقدار مرجع مقایسه کردیم و دلیلی وجود ندارد که انتظار داشته باشیم که Kx برابر مقدار مطلوب باشد. برای جبران این مساله باید ورودی را مقیاس بندی کنیم تا خروجی حاصل برابر ورودی مرجع شود. این فاکتور مقیاس بندی $\bar{N}$ نامیده می‌شود که درشکل شماتیک زیر نمایش داده شده است.



شکل (۱-۲۳) کنترل به روش فیدبک کلیه متغیرهای حالت همراه با ورودی مرجع

با چند بار سعی و خطا کردن برای رسیدن به پاسخ مناسب ، مقدار Nbar را برابر 30 قرار می‌دهیم.

```
x0=[0];
p1=-1.5;
K=place(A,B,[p1]);
Nbar=30;
A1=A-B*K;
lsim(A1,B*Nbar,C,D,u,t,x0);
```



شکل (۱-۲۴) پاسخ پله سیستم حلقه بسته

از شکل(۱-۲۴) واضح است که خطای ماندگار حذف شده است.

۱– ۶ طراحی کنترلر دیجیتالی

همانطور که در قسمت مدل سازی دیدیم، تابع تبدیل مدل کروز به صورت $Y(s)/U(s) = 1/(ms+b)$ به‌دست آمد. اولین قدم در آنالیز دیجیتالی این است که تابع تبدیل گسسته را از روی تابع تبدیل پیوسته $1/(ms+b)$ به‌دست آوریم. برای انجام این کار از دستور c2dm استفاده می‌کنیم. برای استفاده از این دستور باید چهار پارامتر زیر را تنظیم کنیم:

۱- مدت زمان نمونه برداری T_s که برحسب sec/ *sample* می‌باشد. این پارامتر باید کوچکتر از $1/(30\times BW)$ باشدکه در آن BW پهنای باند فرکانسی سیستم حلقه بسته است.

۲- ماتریس مخرج.

۳- ماتریس صورت کسر.

۴- روشی که برای نمونه برداری انتخاب می‌کنیم.

دراینجا روش ZOH[1] (نگهدارنده مرتبه صفر) را انتخاب می‌کنیم. فرض کنید زمان نمونه برداری برابر $\frac{1}{50}$ ثانیه است، یعنی فرض کردیم پهنای باند برابر $1\ rad/\sec$ است. بنابراین دستورات زیر را به ادامه m-file اضافه می‌کنیم:

```
num=[1];
den=[1000 50];
Ts=1/50;
[numDz,denDz] = c2dm (num,den,Ts,'zoh')
```

با اجرای m-file تابع تبدیل دیجیتالی به صورت زیر به‌دست می‌آید:

$$\frac{Y(z)}{U(z)} = \frac{0.0001\times(0.19)}{z-0.99} \qquad (۱۶-۱)$$

۱-۶-۱ ر سم مکان هندسی در صفحه Z

با استفاده از دستور zgrid می‌توانیم مکان هندسی ریشه‌ها را رسم کرده، و سپس ازروی نمودار، بهره k مطلوب را به‌دست آوریم. این دستور به دو آرگومان نیاز دارد:

۱- فرکانس طبیعی ω_n

۲- نسبت میرایی ζ که این دو پارامتر از روی زمان خیز T_r وجهش M_p به‌دست می‌آیند.

$$\omega_n = \frac{1.8}{T_r},\ M_p = e^{-\frac{\pi\zeta}{\sqrt{1-\zeta^2}}} \qquad (۱۷-۱)$$

از آنجایی که خیز 5 ثانیه و جهش 10% است، بنابراین مقدار $\omega_n = 0.36\ rad/\sec$ ، $\zeta = 0.6$ به‌دست می‌آید. مقدار ω_n, ζ باید بیشتر از این مقادیر باشد.

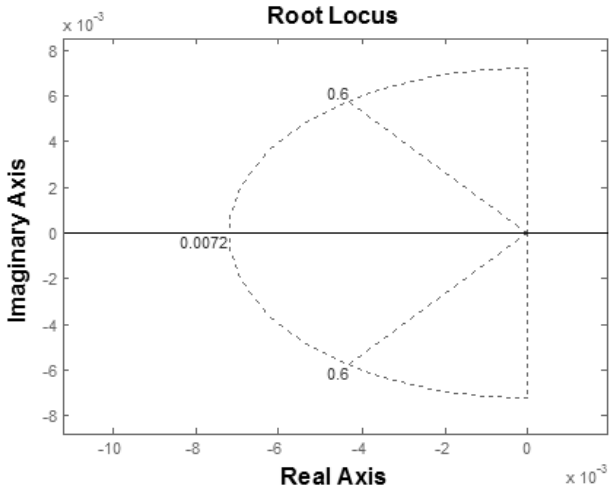
[1] Zero order Hold

آرگومان ω_n باید برحسب $rad/sample$ باشد، بنابراین:

$$\omega_n = 0.36\frac{rad}{\sec}\times\frac{1}{50}\frac{\sec}{sample} = 0.0072\frac{rad}{sample}$$

با اجرای فرامین زیر نمودار زیر حاصل می‌شود:

```
Wn=0.0072;
zeta=0.6;
rlocus (numDz,denDz)
zgrid (zeta, Wn)
axis ([-1 1 -1 1])
```

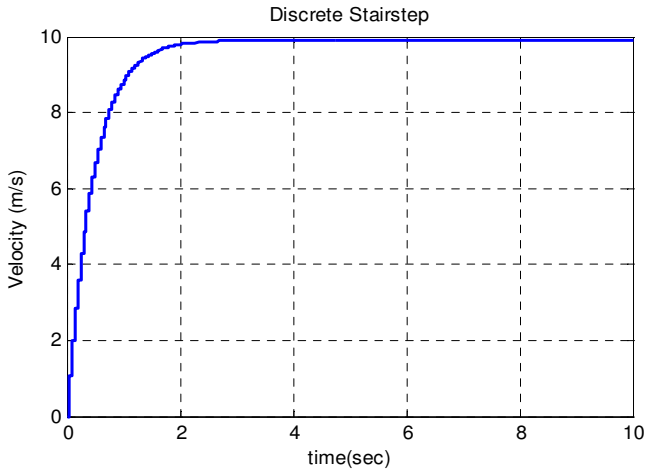


شکل (۱-۲۵) مکان هندسی ریشه‌ها

خط نقطه چین نشان دهنده مکان ثابت میرایی $\zeta = 0.6$ است، به طوری که ثابت میرایی بیشتر از 0.6 بین دو خط و ثابت میرایی کمتر از 0.6 خارج این دو خط قرار می‌گیرد. نیم بیضی نیز نشان دهنده فرکانس طبیعی ثابت $\omega_n = 0.36\ rad/\sec$ است، به طوری که فرکانس طبیعی بیشتر از 0.36 خارج نیم بیضی و فرکانس کمتر از 0.36 داخل بیضی قرار می‌گیرد. اجازه دهید تا بهره K را با استفاده از دستور rlocfind پیدا کرده و پاسخ پله را به‌دست آوریم. دستورات زیر را به ادامه m-file اضافه کنید:

```
[K,poles]=rlocfind (numDz,denDz)
[numcDz,dencDz] = cloop (K*numDz,denDz);
U=10;
[x] = dstep (U*numcDz,dencDz,201);
figure(1)
t=0:0.05:10;
stairs (t,x)
```

حال روی مکان هندسی ریشه‌ها یک نقطه نزدیک نقطه 0.9+ انتخاب کنید. مقدار K و موقعیت قطب‌ها داده می‌شود ($k = 4500,\ Pole = 0.9$) از طرفی برای سیستم حلقه بسته چنین پاسخی حاصل شود:



شکل (۱-۲۶) پاسخ پله سیستم حلقه بسته با کنترلر دیجیتالی

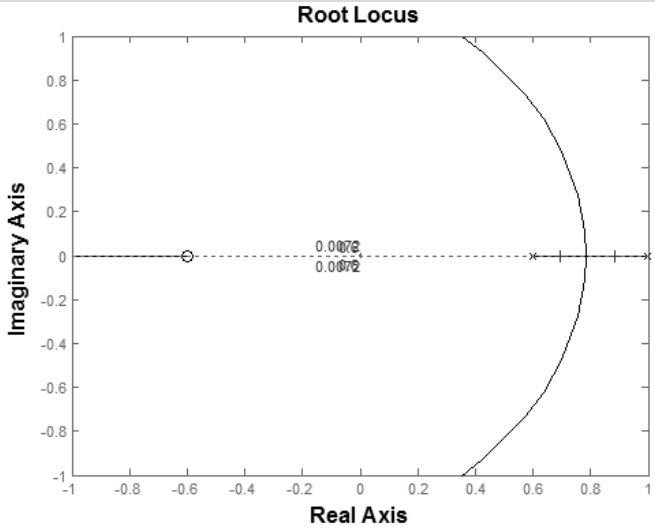
از روی نمودار واضح است که این طراحی تمامی نیازهای ما را برآورده کرده است اما مقدار بهره به‌دست آمده حدود 4500 است که خیلی بالابوده و تلاش کنترلر به مراتب زیاد می‌شود و ممکن است در سیستم‌های واقعی این مقدار قابل دستیابی نباشد. برای رسیدن به یک بهره مطلوب و واقعی باید کنترلر گسسته خود را تصحیح کنیم. برای حل این مشکل یک جبران کننده Lead به سیستم اضافه می‌کنیم تا پاسخ مطلوبی به‌دست آید:

$$G_{Lead}(z) = K.\frac{z+a}{z+b} \qquad (۱-۱۸)$$

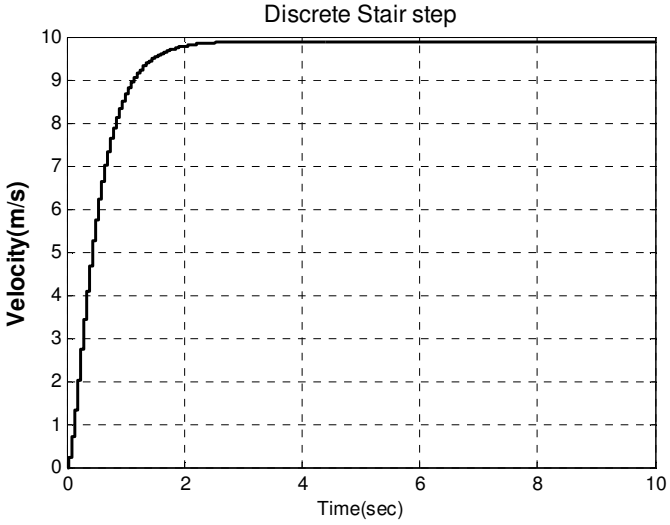
در ابتدا ما نیاز داریم که بهره K را کاهش دهیم تا پاسخ منطقی‌ای داشته باشیم. از طرفی می‌دانیم که در قطبها مقدار بهره k صفر و در صفرها این مقدار به بینهایت می‌رسد. تمام قطب‌ها باید در یک دایره به شعاع 1 قرار بگیرند تا سیستم پایدار باشد. بنابراین نقطه مورد نظر باید نزدیک قطب‌ها باشد. با در نظر گرفتن این دو مورد یک قطب در 0.6+ و یک صفر در 0.6- در نظر می‌گیریم. دستورات زیر به را به ادامه m-file اضافه کنید:

```
num=[1]; den=[1000 50];
Ts=1/50;
[numDz,denDz] = c2dm (num,den,Ts,'zoh');
numleadDz=[1 0.6]; denleadDz=[1 -0.6];
numDnew=conv (numDz,numleadDz);
denDnew=conv (denDz,denleadDz);
Wn=0.0072; zeta=0.6;
rlocus (numDnew,denDnew)
zgrid (zeta, Wn)
axis ([-1 1 -1 1])
[K,poles] = rlocfind (numDnew,denDnew)
[numcDnew,dencDnew] = cloop (K*numDnew,denDnew);
U=10;
```

```
[x] = dstep (U*numcDnew,dencDnew,201);
figure
t=0:0.05:10;
stairs (t,x)
```



شکل (۱-۲۷) مکان هندسی ریشه‌ها با جبران کننده Lead



شکل (۱-۲۸) پاسخ پله سیستم حلقه بسته با کنترلر Lead دیجیتالی

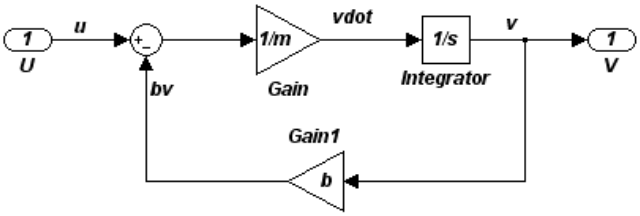
با اجرای m-file از شما یک نقطه روی مکان هندسی ریشه‌ها درخواست می‌شود. نقطه‌ای نزدیک 0.9+ انتخاب کنید. جواب حاصل مشابه حالت قبل است ولی مقدار بهره به 1000 کاهش پیدا کرده و تلاش کنترلی به مراتب کمتر می‌شود و جواب معقولانه تری به دست می‌آید.

۱– ۷ طراحی کنترلر PI با سیمولینک

همانطور که قبلاً گفته شده بود مدل‌سازی سیستم کنترل کروز بسیار ساده می‌باشد. اگر در این مساله از اینرسی چرخ‌ها چشم‌پوشی کنیم و فرض کنیم که اصطکاک با سرعت رابطه مستقیم داشته و به بدنه جرم m اعمال می‌شود، در آن صورت مساله به سیستم جرم و فنر ساده تبدیل می‌شود. برای مدل‌سازی کروز در محیط سیمولینک بلوک مورد نیاز را از جدول زیر به پنجره مدل‌سازی انتقال می‌دهیم .

تعداد	مسیر بلوک
1	Simulink>>Sources>>step
1	Simulink>>Source>>In1
1	Simulink>>Sink>>Out1
1	Simulink>>Sink>>Scope
1	Simulink>>Continuous>>Integrator
1	Simulink>>Continuous>>TransferFcn
2	Simulink>>Math operation>>Gain
2	Simulink>> Math operation >> Sum

بعد از انتقال بلوک‌ها مطابق شکل(۱-۲۹) بلوک‌ها را به هم متصل می‌کنیم تا مدل دینامیکی کروز ساخته شود.



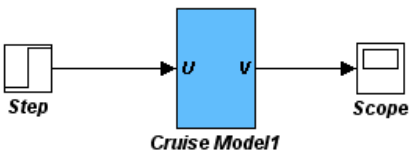
شکل (۱-۲۹) مدل بلوک نمودار کروز

سپس تمامی بلوک‌ها را انتخاب کرده و آنها را در یک زیر سیستم (Subsystem) قرار می‌دهیم. از این به بعد این زیر سیستم به عنوان مدل کروز شناخته می‌شود. در مرحله بعد یک ورودی پله به سیستم اعمال می‌کنیم تا خروجی سیستم حلقه باز را مشاهده کنیم. روی بلوک ورودی پله دوبار کلیک کرده و مقادیر زیر را وارد نمایید:

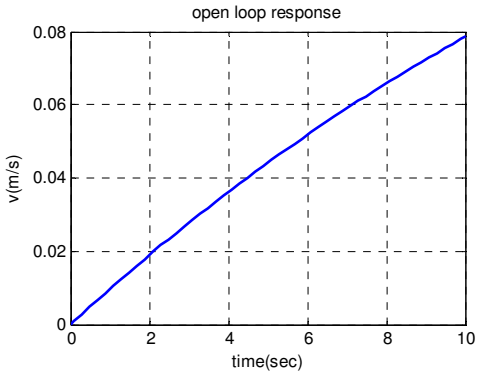
Step time=0; final value = u;

زمان شبیه سازی را برابر 10 ثانیه قرار دهید و مقادیر زیر را در فضای کاری MATLAB وارد نموده، و دکمه Run را بزنید تا شبیه سازی شروع شود.

```
m=1000;b=50;u=10;
```



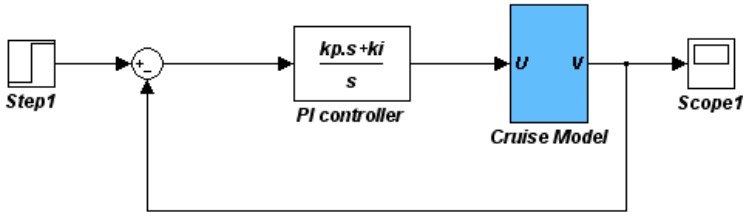
شکل (۱-۳۰) مدل کروز با ورودی پله



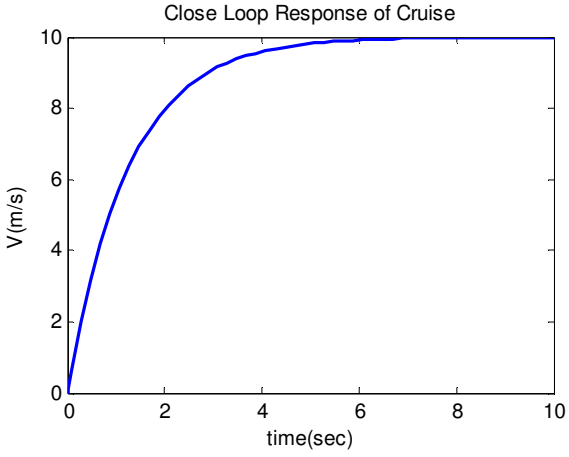
شکل (۱-۳۱) پاسخ پله سیستم

برای طراحی کنترلر PI بر روی بلوک Transfer Fcn دوبار کلیک کنید و تنظیمات زیر را انجام دهید:

Num=[kp ki]; den=[1 0]; ودر محیط فضای کاری MATLAB مقادیر $kp = 800, ki = 40$ را وارد کنید.



شکل(۱-۳۲) سیستم با ورودی PI



شکل (۱-۳۳) پاسخ سیستم به ورودی PI

پروژه دوم

کنترل سرعت موتور DC

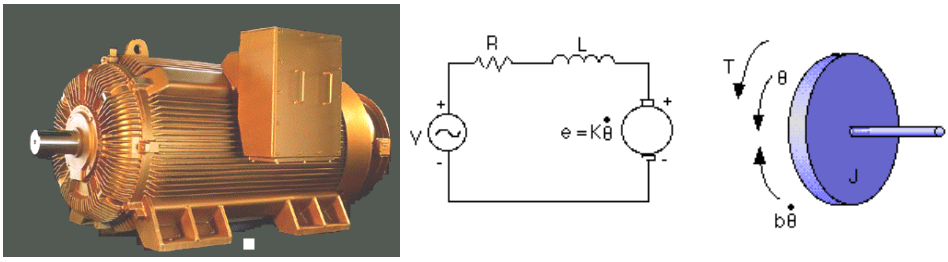
اهداف این پروژه

- مدل سازی
- طراحی کنترلر PID
- رسم مکان هندسی ریشه‌ها (Root Locus)
- پاسخ فرکانسی
- طراحی فضای حالت
- طراحی کنترلر دیجیتالی
- طراحی جبران کننده Lag در محیط سیمولینک

۲ – ۱ مدل سازی

۲-۱-۱ مدل سازی سیستم براساس اصول فیزیکی و استخراج معادلات سیستم

یکی از محرک‌های رایج در سیستم‌های کنترلی، موتور DC می‌باشد که به طور مستقیم می‌تواند حرکت دورانی، و به طور غیرمستقیم حرکت خطی تولید کند. مدار الکتریکی آرمیچر و نمودار آزاد روتور در شکل (۲-۱) نمایش داده شده است.



شکل (۲- ۱) موتور DC

برای این مساله فرض کنید مقادیر زیر را برای پارامترهای فیزیکی انتخاب کرده‌ایم؛ این مقادیر از یک آزمایش واقعی استخراج شده اند.

ممان اینرسی روتور: $J = 0.01 \frac{kg.m^2}{s^2}$

ضریب میرایی سیستم مکانیکی: $C_m = 0.1Nm.s$

ثابت نیروی الکتروموتوری: $K = K_e = K_t = 0.01 \frac{N.m}{Amp}$

مقاومت الکتریکی : $R = 1\Omega$

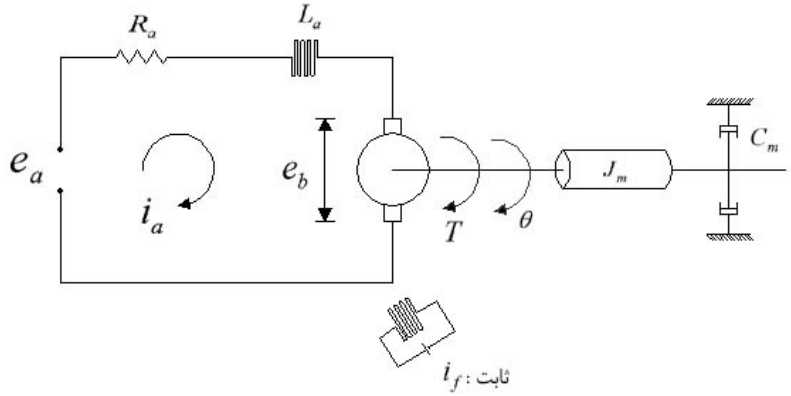
اندوکتانس الکتریکی: $L = 0.5H$

ورودی(V): ولتاژ منبع تغذیه

خروجی$\dot{\theta}$: سرعت شفت

در این قسمت فرض بر این است که شفت و روتور صلب می‌باشند.

با توجه به شکل (۲-۲) گشتاور موتور با جریان ورودی موتور رابطه مسقیم دارد.

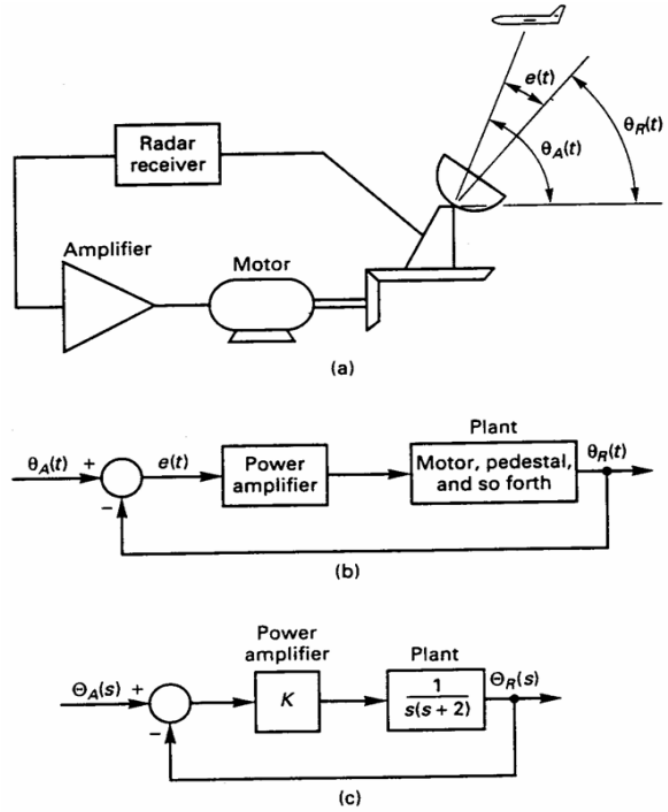


شکل (۲- ۲) مدار الکتریکی موتور به‌همراه قسمت مکانیکی

$$\varphi_f \propto i_f \Rightarrow T \propto i_a \times i_f \Rightarrow T = k_t i_a \quad (۱-۲)$$

و نیروی پس زن الکترموتوری با سرعت موتور رابطه مستقیم دارد ($e_b = k_b \dot{\theta} = EMF$). در سیستم SI دو ضریب K_e ،K_t با هم برابرند ($K_e = K_t = K$).

یکی از کاربردهای کنترل سرعت و موقعیت موتور DC، کنترل موقعیت و سرعت حرکت رادار است که در شکل (۲-۳) نمایش داده شده است.



شکل (۲- ۳) کنترل حرکت رادار با استفاده از موتور DC

اگر قوانین کیرشهف و قوانین نیوتن را برای مدار شکل روبرو بنویسیم، داریم:

$$\begin{cases} e_a - Ri - L\dfrac{di}{dt} - e_b = 0 \\ T = J\ddot{\theta} + C_m\dot{\theta} \end{cases} \Rightarrow \begin{cases} V - K\dot{\theta} = Ri + L\dfrac{di}{dt} \\ Ki = J\ddot{\theta} + C_m\dot{\theta} \end{cases} \qquad (۲-۲)$$

۲-۱-۲ تابع تبدیل سیستم

برای به‌دست آوردن تابع تبدیل سیستم نیاز دارید از معادله (۲-۲) تبدیل لاپلاس بگیرید. توجه کنید زمانی که در حال پیدا کردن تابع تبدیل سیستم هستید شرایط اولیه را برابر صفر در نظر بگیرید.

$$\begin{cases} V - K.s.\theta(s) = R\,I(s) + L\,s.I(s) \\ K\,I(s) = Js^2\theta(s) + C_m s.\theta(s) \end{cases} \qquad (۳-۲)$$

$$\xrightarrow{by\ ELiminating\ I(s)} V - Ks\theta(s) = \frac{Js^2\theta(s) + C_m s.\theta(s)}{K}(R + Ls)$$

$$\frac{\omega(s)}{V(s)} = \frac{K}{(Js + C_m)(R + Ls) + K^2} \qquad (۴-۲)$$

۲-۱-۳ مدل فضای حالت سیستم

دراین مساله سرعت دورانی روتور و جریان موتور را به عنوان متغیرهای حالت در نظر بگیرید. بنابراین معادله بالا را می‌توان به صورت زیر نوشت:

$$\begin{cases} \dot{x}(t) = Ax(t) + Bu(t) \\ y(t) = Cx(t) + Du(t) \end{cases} \Rightarrow \begin{cases} \dfrac{d}{dt}\begin{bmatrix} \dot{\theta} \\ i \end{bmatrix} = \begin{pmatrix} \dfrac{-C_m}{J} & \dfrac{K}{J} \\ \dfrac{-K}{L} & \dfrac{-R}{L} \end{pmatrix}\begin{bmatrix} \dot{\theta} \\ i \end{bmatrix} + \begin{bmatrix} 0 \\ \dfrac{1}{L} \end{bmatrix} V \\ \dot{\theta} = \begin{bmatrix} 1 & 0 \end{bmatrix}\begin{bmatrix} \dot{\theta} \\ i \end{bmatrix} \end{cases} \quad (۵-۲)$$

۲-۱-۴ قیدهای حاکم بر طراحی

قدم بعدی در مدل سازی این سیستم، این است که بدانیم این سیستم قرار است چه شرایطی را داشته باشد؛ یعنی قیدهای حاکم بر سیستم باید مشخص شود.

- از آنجایی که سیستم قرار است با سرعت مشخصی بچرخد پس خطای حالت ماندگار 1% را برای آن در نظر می‌گیریم.
- زمان رسیدن به سرعت ماندگار باید کمتراز 2 ثانیه باشد پس $Settling\ time < 2s$.
- از آنجایی که سرعت بیش از اندازه باعث خراب شدن تجهیزات می‌شود پس مقدار حداکثر جهش سرعت را برابر 5% در نظر می‌گیریم پس $Overshoot < 5\%$.

۲-۱-۵ پاسخ سیستم حلقه باز

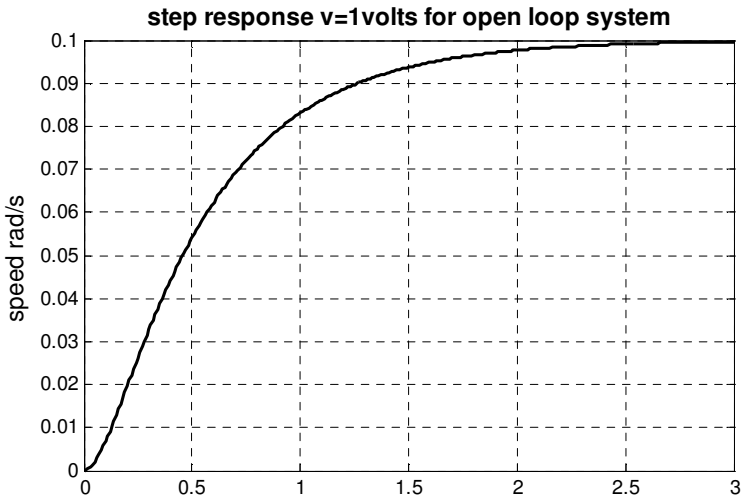
حال بهتر است اجازه دهید پاسخ سیستم حلقه باز به ورودی پله را بررسی کنیم.

$$num = K$$

$$den = (Js + C_m)(R + Ls) + K^2$$

دستورات زیر را در یک m-file وارد کنید:

```
%==========  Open loop system  ========
j=0.01;Cm=0.1;R=1;L=0.5;k=0.01;
num=k;
den=[(j*L)  ((j*R)+(Cm*L))  ((R*Cm)+k^2)];
sys=tf(num,den);
figure(1)
t=0:0.01:3;
step(sys,t);
title('step response v=1volts for open loop system')
ylabel('speed rad/s')
```



شکل (۲-۴) پاسخ پله سیستم حلقه باز

از روی شکل مشخص است که زمان نشست حدود 3 ثانیه است و همانطور که قبلاً گفته شده بود، زمان نشست باید کمتر از 2 ثانیه باشد.

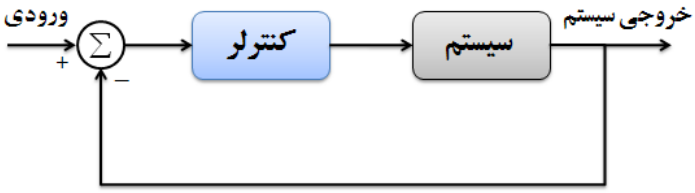
۲ – ۲ طراحی کنترلر PID

از معادله (۲-۴) تابع تبدیل سیستم به صورت زیر به‌دست آمد:

$$\frac{\omega(s)}{V(s)}=\frac{K}{(Js+C_m)(R+Ls)+K^2}$$

بلوک نمودار سیستم کنترلی با فیدبک واحد در شکل(۲-۵) نمایش داده شده است. تابع تبدیل کنترلر PID به صورت زیر است :

$$PID: k_p+k_D s+\frac{k_I}{s}=\frac{k_D s^2+k_p s+k_I}{s} \qquad (۲-۶)$$



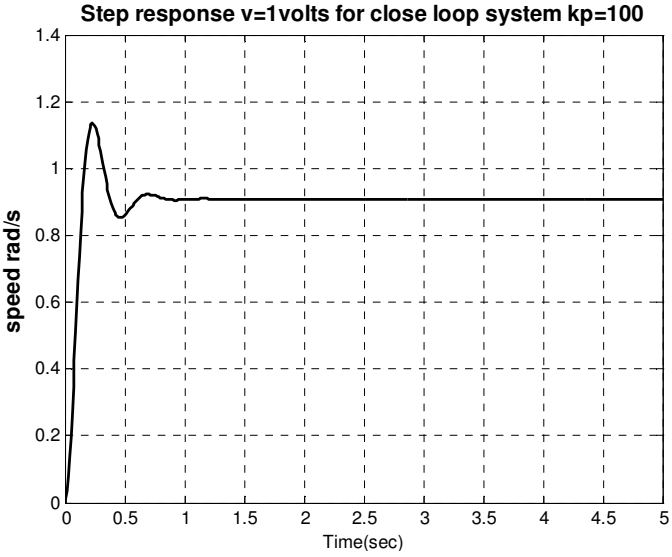
شکل (۲-۵) بلوک نمودار سیستم کنترلی با فیدبک واحد

۲-۲-۱ کنترلر تناسبی

در ابتدا یک کنترلر تناسبی (پروپرشنال) قرار دهید. اولین کاری که باید انجام دهید این است که تابع تبدیل سیستم حلقه بسته با کنترلر تناسبی (k_p) را به‌دست آورده، سپس در برنامه مقدار $k_p = 100$ را قرار دهید.

```
%============ P controller ==================
kp=100;
numa=kp*num;
dena=den;
[numc,denc]=cloop(numa,dena);
sys_p=tf(numc,denc);
step(sys_p,0:0.01:4);
title('step response v=1volts for close loop system kp=100')
```

خروجی به صورت زیر خواهد بود:



شکل (۲-۶) پاسخ پله سیستم همراه با کنترل تناسبی

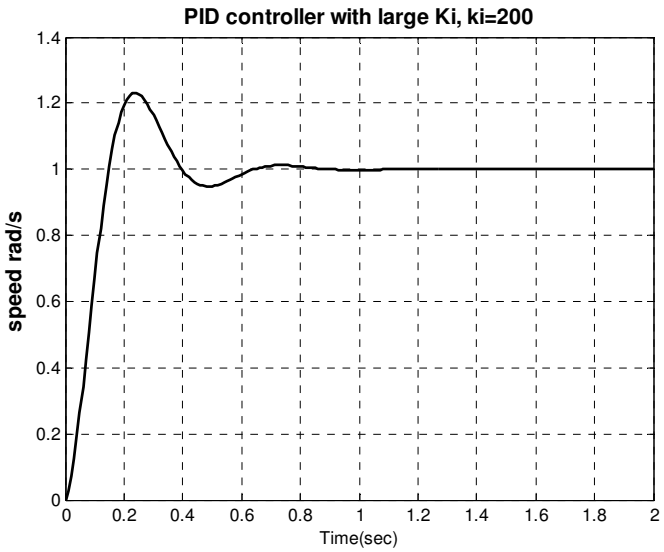
۲-۲-۲ کنترلر PID و تنظیم بهره‌ها

همانطور که از شکل بر می‌آید، مقدار جهش حدود 20% و خطای حالت ماندگار بیشتر از5% می‌باشد. بنابراین برای کاهش خطای ماندگار از کنترل انتگرالی و برای کاهش مقدار جهش از کنترلر مشتق‌گیر استفاده کنید.

```
%===== PID Controller ========
ki=200;kp=100;kd=1;
numco=[kd kp ki];
denco=[1 0];
numa1=conv(num,numco);
dena1=conv(den,denco);
[numcl,dencl]=cloop(numa1,dena1);
sys_pid=tf(numcl,dencl);
figure(3)
step(sys_pid,0:0.01:1);
```

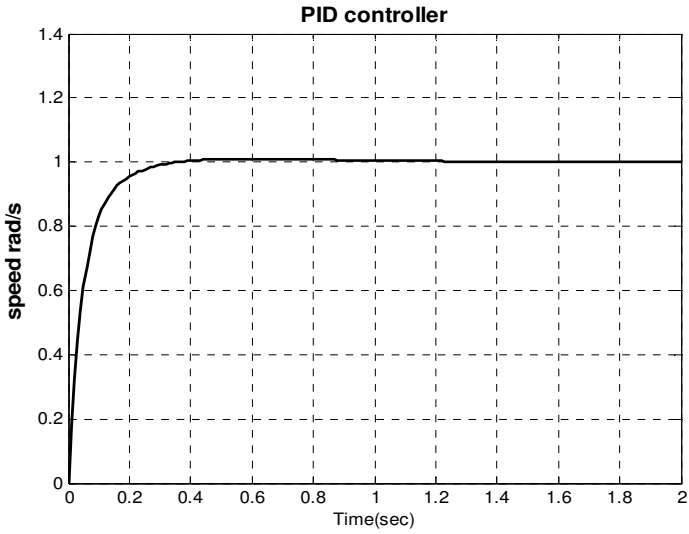
```
title('step response v=1volts for close loop system PID controller')
```

خروجی به صورت زیر به‌دست می‌آید:



شکل (۲-۷) پاسخ پله سیستم با کنترلر PID با Ki بزرگ

همانطورکه در شکل مشاهده می‌شود سرعت بیشتر از قبل است، ولی افزایش Ki باعث بدتر شدن پاسخ حالت گذرا نیز شده است. بنابراین با افزایش Kd مقدار جهش را کاهش می‌دهیم تا پاسخ حالت گذرا بهبود پیدا کند. پس به m-file برگشته و مقدار kd را برابر 10 قرار دهید.



شکل (۲-۸) پاسخ پله سیستم با کنترلر PID

۲ – ۳ رسم مکان هندسی ریشه‌ها (Root Locus)

از قسمت مدل‌سازی به یاد داریم که تابع تبدیل سیستم حلقه باز به صورت زیر به‌دست می‌آید:

$$\frac{\omega(s)}{V(s)} = \frac{K}{(Js + C_m)(R + Ls) + K^2}$$

در این مسأله هدف ما بر این است که زمان نشستی کمتر از 2 ثانیه، مقدار خیز کمتر از 5%، و خطای حالت ماندگار کمتر از 1% باشد. حال با استفاده از روش مکان هندسی ریشه‌ها کنترلری برای این سیستم طراحی می‌کنیم. ایده اصلی در طراحی به کمک مکان هندسی ریشه‌ها پیدا کردن پاسخ سیستم حلقه بسته از روی نمودار مکان هندسی ریشه‌های سیستم حلقه باز می‌باشد، به طوری که با اضافه کردن یک سری صفر و قطب به سیستم اصلی پاسخ سیستم حلقه بسته تصحیح شود. پس فرامین زیر را در یک m-file جدید بنویسید:

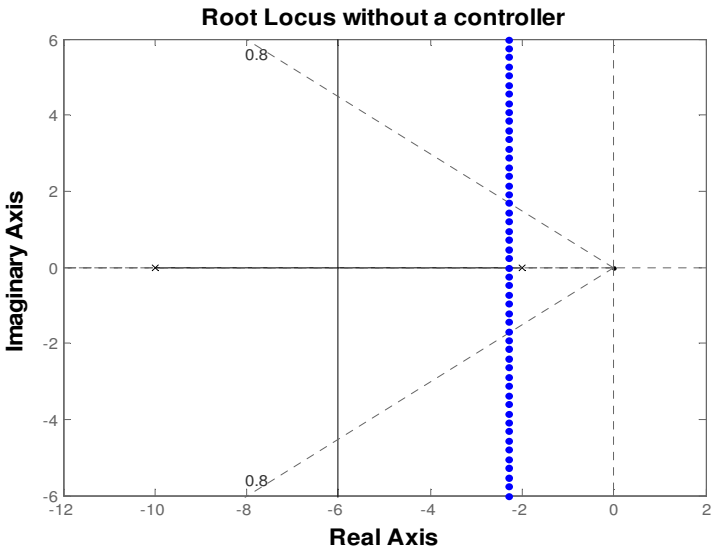
```
J=0.01;Cm=0.1;K=0.01;R=1;L=0.5;
num=K;
den=[(J*L) ((J*R)+(L*Cm)) ((Cm*R)+K^2)];
rlocus(num,den)
sgrid(.8,0)
sigrid(2.3)
title('Root Locus without a controller')
```

برای استفاده از دستور sigrid باید فرامین زیر را در یک m-file بنویسید و آن را save کنید.

```
function[ ] = sigrid(sig)
%SIGRID   Generate s-plane grid lines for a root locus or
pole-zero map.
error(nargchk(1,1,nargin));

hold on
%Plot sigma line
    limits = axis;
    mx=limits(1,4);
    mn=limits(1,3);
    stz=abs(mx)+abs(mn);
    st=stz/50;
    im=mn:st:mx;
    lim=length(im);
    for i=1:lim
        re(i)=-sig;
    end
    re(:);
    plot(re,im,'.')
hold off
return
```

دستور sgrid به دو آرگومان نیاز دارد: نسبت میرایی $\zeta = 0.8$ که متناسب با جهش 5% است و فرکانس طبیعی $\omega_n = 0$ (متناظر با حالت بدون در نظر گرفتن زمان خیز). اما، دستور Sigrid نیاز به یک آرگومان دارد.

($\zeta\omega_n = \frac{4.6}{T_s} = \frac{4.6}{2} = 2.3$)

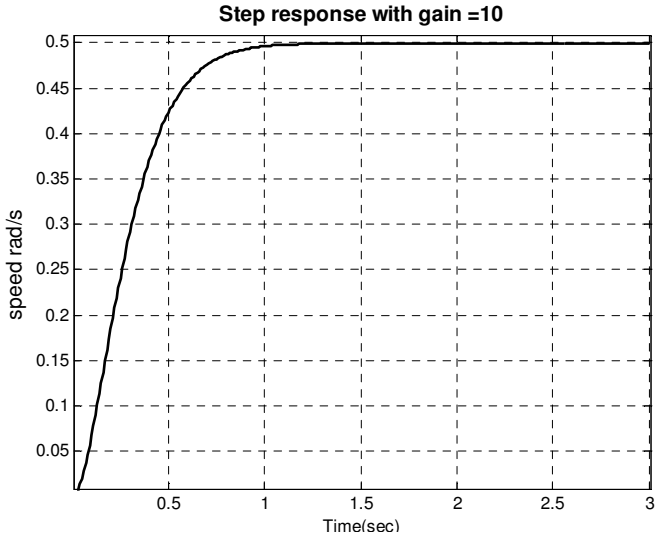


شکل (۲-۹) مکان هندسی ریشه‌ها برای سیستم حلقه باز

پیدا کردن بهره با استفاده از دستور rlocfind: نیاز است که زمان نشست و جهش تا آنجا که امکان دارد کوچک باشد. نسبت میرایی بزرگ متناظر با نقاطی است که نزدیک محور حقیقی هستند و پاسخ سریع تر متناظر با نقاطی است که از محور موهومی دور باشند. برای پیدا کردن این نقاط می‌توانیم از دستور rlocfind استفاده کنیم. با استفاده از این دستور می‌توانید بهره K را پیدا کرده و پاسخ پله سیستم را با این بهره به‌دست آورید. بنابراین دستورات زیر را به ادامه m-file اضافه کنید.

```
[k,poles] = rlocfind(num,den)
[numc,denc]=cloop(k*num,den,-1);
t=0:0.01:3;
step(numc,denc,t)
title('Step response with gain =10')
```

در ادامه از شما یک نقطه روی مکان هندسی ریشه‌ها خواسته می‌شود. برای این منظور نقطه $-6+2i$ را انتخاب کنید. بنابراین پاسخی مشابه زیر به‌دست می‌آید.



شکل (۲-۱۰) پاسخ پله سیستم حلقه بسته با بهره انتخابی

از روی شکل مشخص است که سیستم میرای فوق بحرانی[1] بوده و زمان نشست حدود 1 ثانیه است. بنابراین مقدار جهش و زمان نشست راضی‌کننده است. تنها مشکلی که در شکل هم دیده می‌شود خطای حالت ماندگار است که حدود 50% می‌باشد. اگر بهره را افزایش دهید تا خطای حالت ماندگار کاهش یابد، مقدار جهش بیشتر می‌شود. برای حل این مشکل از یک کنترلر Lag استفاده می‌شود.

۲-۳-۱ اضافه کردن کنترلر Lag

نمودار مکان هندسی ریشه‌ها برای این مثال بسیار ساده است. معیارهای زمان نشست و میرایی با کنترلر تناسبی متناسب است. فقط خطای حالت ماندگار با کنترلر تناسبی متناسب نیست. جبران کننده Lag می‌تواند این خطای حالت ماندگار را کاهش دهد اما با انجام دادن این کار زمان نشست بیشتر می‌شود.

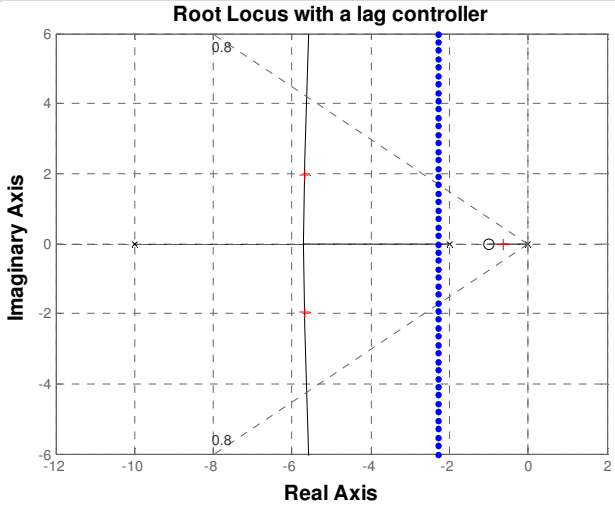
$$G_{Lag}(s)=\frac{s+1}{s+0.01} \qquad (۲-۷)$$

بنابراین m-file خود را به صورت زیر اصلاح کنید:

```
J=0.01;Cm=0.1;K=0.01;R=1;L=0.5;
num=K; den=[(J*L) ((J*R)+(L*Cm)) ((Cm*R)+K^2)];
z1=1;p1=0.01;
numa = [1 z1]; dena = [1 p1];
numb=conv(num,numa);
denb=conv(den,dena);
rlocus(numb,denb)
sgrid(.8,0)
```

[1] Over damped

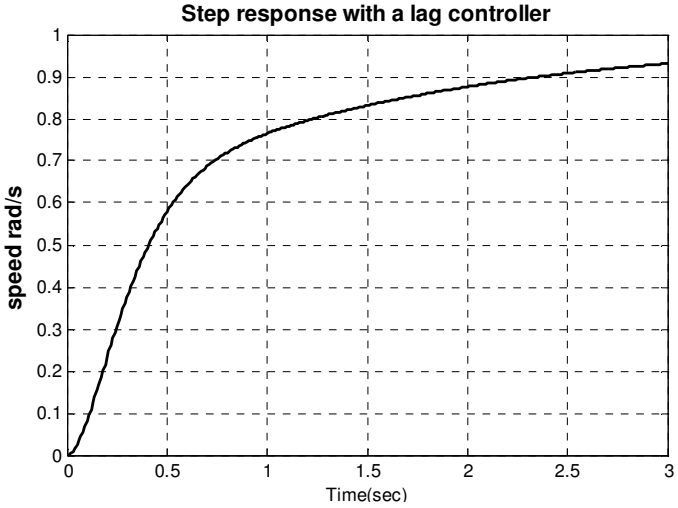
```
sigrid(2.3)
title('Root Locus with a lag controller')
```



شکل (۲-۱۱) مکان هندسی ریشه‌های سیستم به همراه کنترلر Lag

Numa,dena صورت و مخرج کنترلر و numb , denb صورت مخرج تابع تبدیل حلقه باز است. حال پاسخ سیستم حلقه بسته را رسم می‌کنیم. دوباره همان نقطه $-6+2i$ را انتخاب می‌کنیم.

```
[k,poles]=rlocfind(numb,denb)
[numc,denc]=cloop(k*numb,denb,-1);
t=0:0.01:3;
step(numc,denc,t)
title('Step response with a lag controller')
```



شکل (۲-۱۲) پاسخ پله به همراه کنترلر Lag با بهره ۱۲

بهره حدود 12 است ولی این جواب مناسب نیست، زیرا قطب Lag نزدیک محور موهومی است؛ پس تا حدودی آرام است و زمان نشست بیشتر می‌شود. حال مقدار بهره را افزایش داده و نقطه دورتری را انتخاب می‌کنیم ($gain = 50$).

شکل (۲-۱۳) پاسخ پله به همراه کنترلر Lag با بهره ۵۰

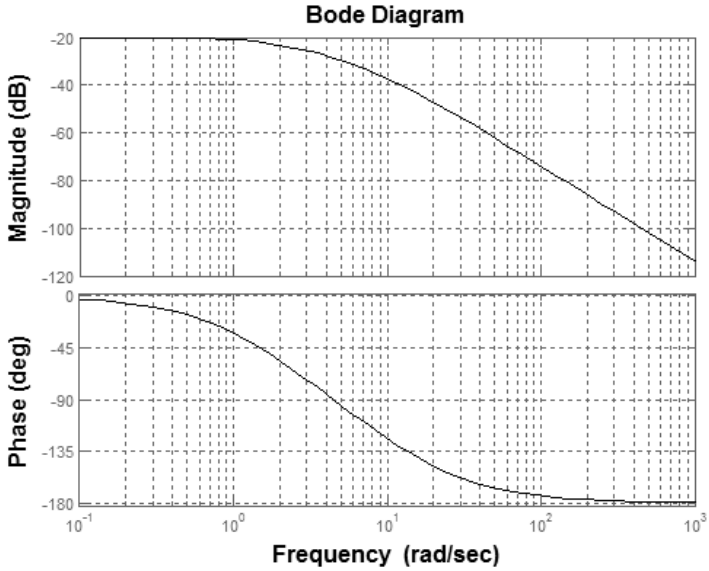
با انجام این کار به جواب مطلوب تری خواهید رسید و تمام معیارهای طراحی برآورده می‌شود، یعنی زمان نشست کمتر از 2 ثانیه، جهش کمتر از 5% ، وخطای حالت ماندگار کمتر از 1% است.

۲ – ۴ پاسخ فرکانسی

ایده اصلی طراحی برمبنای پاسخ فرکانسی بر این است که با رسم نمودار بود[1] تابع تبدیل حلقه باز، تخمینی از پاسخ سیستم حلقه بسته به‌دست آید. اضافه کردن کنترلر به سیستم، نمودار بود حلقه باز را تغییر می‌دهد، پس در ابتدا نمودار بود سیستم حلقه باز را رسم می‌کنیم. برنامه زیر را در یک m-file جدید نوشته و برنامه را اجرا کنید:

```
J=0.01;Cm=0.1;K=0.01;R=1;L=0.5;
num=K;
den=[(J*L)  ((J*R)+(L*Cm))  ((Cm*R)+K^2)];
bode(num,den)
```

[1] Bode diagram



شکل (۲-۱۴) نمودار بود سیستم حلقه باز

۲-۴-۱ اضافه کردن بهره تناسبی

از روی نمودار بود دیده می‌شود در صورتی‌که $\omega < 10\ rad/\sec$ ، حد فاز بزرگتر از 60 می‌شود. اجازه بدهید که یک بهره به سیستم اضافه شود. بنابراین پهنای باند فرکانسی $10\ rad/\sec$، حد فاز حدود $60\deg$ می‌دهد. با استفاده از دستور bode می‌توانید به مقدار دقیق بزرگی[1] نیز برسید.

```
[mag,phase,w] = bode(num,den,10)
mag =
    0.0139
phase =
 -123.6835
```

$$gain = \frac{1}{magnitude} = \frac{1}{0.013} = 72$$

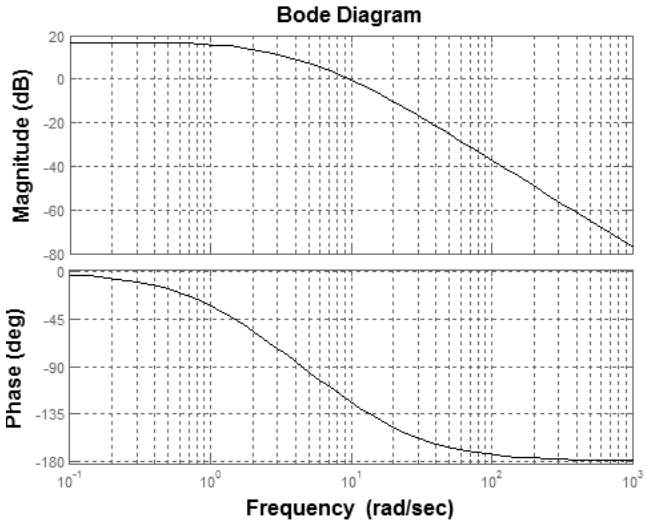
۲-۴-۲ رسم پاسخ حلقه بسته

از نمودار بالا می‌بینید که مقدار حد فاز به اندازه کافی بزرگ است. بهتر است پاسخ سیستم حلقه بسته را بررسی کنید. پس دستورات زیر را به ادامه m-file اضافه کنید.

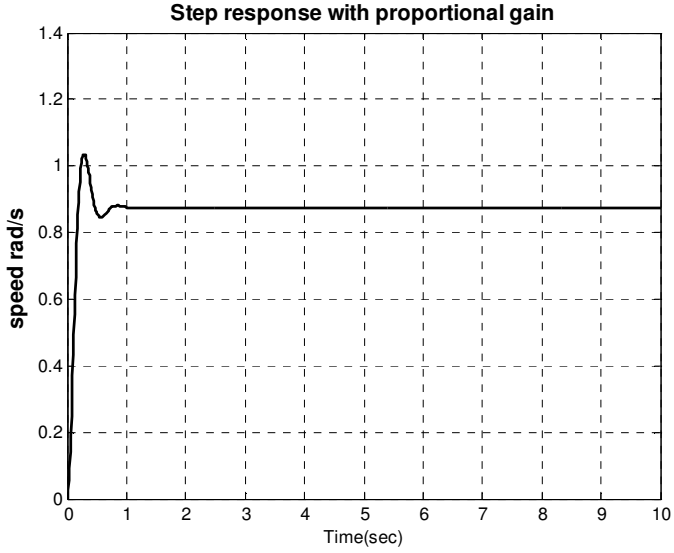
```
num=70*num;
[numc,denc]=cloop(num, den, -1);
t=0:0.01:10;
step(numc,denc,t)
```

توجه : روبروی bode علامت % بگذارید.

[1] magnitude



شکل (۲-۱۵) نمودار بود سیستم حلقه بسته با اضافه شدن بهره تناسبی



شکل (۲-۱۶) پاسخ پله سیستم حلقه بسته

از جواب بر می‌آید که زمان نشست به اندازه کافی سریع است ولی خطای حالت ماندگار و حداکثر جهش زیاد بوده و باید آن‌ها را کاهش دهیم. برای کاهش مقدار جهش باید بهره را کاهش دهیم و کاهش بهره باعث افزایش خطای حالت ماندگار می‌گردد. برای حذف خطای حالت ماندگار باید یک جبران کننده Lag اضافه کنید.

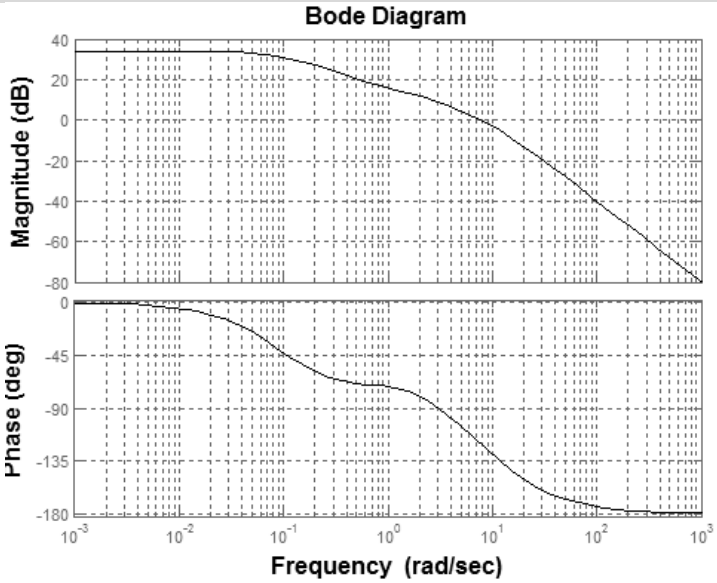
۲-۴-۳ اضافه کردن کنترلر Lag

برای کاهش خطای حالت ماندگار باید از یک کنترلر Lag استفاده کنیم و برای کاهش جهش نیز باید مقدار بهره را کاهش داده و مقدار آن را روی 50 تنظیم کنیم.

$$G_{lag}(s)=\frac{s+1}{s+0.1}$$

دستورات زیر را در یک m-file وارد کنید:

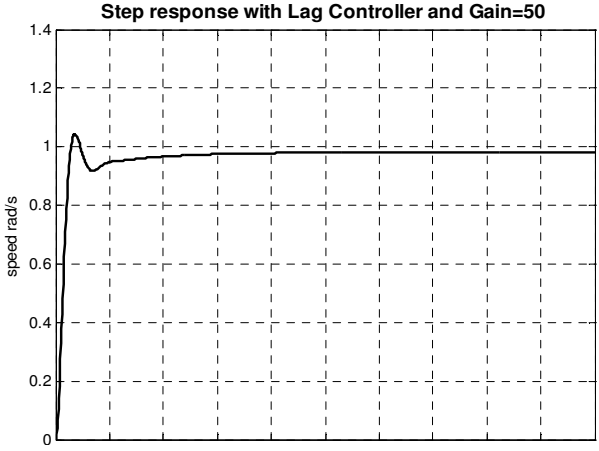
```
num=K;
den=[(J*L)  ((J*R)+(L*Cm))  ((Cm*R)+K^2)];
num=50*K;
z=1;
p=0.1;
numa=[1 z];
dena=[1 p];
numb=conv(num,numa);
denb=conv(den,dena);
bode(numb,denb)
```



شکل (۲-۱۷) نمودار بود برای سیستم با جبران کننده Lag

به نظر می‌رسد که حد فاز مناسب است و پیش بینی می‌شود که خطای حالت ماندگار حدود 1% یا $\frac{1}{40}dB$ باشد. حال پاسخ سیستم حلقه بسته را بررسی کنید، بنابراین دستورات زیر را در یک m-file کپی کنید:

```
[numc,denc]=cloop(numb, denb, -1);
t=0:0.01:10;
step(numc,denc,t)
```



شکل (۲-۱۸) پاسخ پله برای سیستم با کنترلر Lag

از روی پاسخ مشخص است که تمام ملزومات طراحی برآورده شده است.

۲ – ۵ طراحی فضای حالت

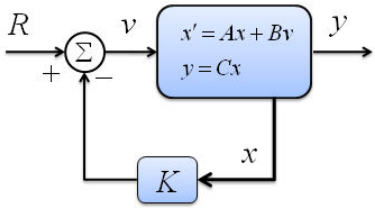
همانطور که قبلاً توضیح داده شده است معادلات فضای حالت برای این سیستم به صورت زیر می‌باشد:

$$\begin{cases} \dot{x} = Ax + Bu \\ y = Cx \end{cases} \Rightarrow \begin{cases} \dfrac{d}{dt}\begin{bmatrix} \dot{\theta} \\ i \end{bmatrix} = \begin{pmatrix} \dfrac{-C_m}{J} & \dfrac{K}{J} \\ \dfrac{-K}{L} & \dfrac{-R}{L} \end{pmatrix}\begin{bmatrix} \dot{\theta} \\ i \end{bmatrix} + \begin{bmatrix} 0 \\ \dfrac{1}{L} \end{bmatrix} V \\ \dot{\theta} = \begin{bmatrix} 1 & 0 \end{bmatrix}\begin{bmatrix} \dot{\theta} \\ i \end{bmatrix} \end{cases}$$

دستورات زیر را در یک m-file وارد کنید:

```
J=0.01;Cm=0.1;K=0.01;R=1;L=0.5;
A=[-Cm/J    K/J;-K/L    -R/L];
B=[0;1/L]; C=[1    0];D=0;
```

از آنجایی که هر دو متغیرحالت را به‌راحتی می‌توان اندازه‌گیری کرد، (جریان با آمپرمتر و سرعت زاویه‌ای با تاکومتر) بنابراین بدون نیاز به طراحی رویتگر می‌توان یک کنترلر با فیدبک کلیه متغیرهای حالت طراحی کرد که شماتیک آن در شکل (۲-۱۹) آمده است.



شکل (۲-۱۹) کنترل به روش فیدبک کلیه متغیرهای حالت

معادله مشخصه این سیستم را می‌توان از رابطه $|sI-(A-BK)|$ به‌دست آورد، که در آن s متغیر حوزه لاپلاس می‌باشد. از آنجا که هر یک از ماتریس‌های A و $B.K_c$، 2×2 می‌باشند بنابراین 2 قطب برای این سیستم می‌توان در نظر گرفت. با استفاده از روش فیدبک کلیه متغیر حالت می‌توانید متغیرها را در هر مکانی که می‌خواهید قرار دهید. پس در ابتدا قطب‌ها را در مکان‌های $-5+5i, -5-5i$ قرار دهید. توجه کنید که این نقاط متناظر با $\omega_n = 5$, $\zeta = 0.98$ می‌باشند. این مقادیر نیز از زمان نشست 1sec و حداکثر جهش 0.1% به‌دست آمده اند.

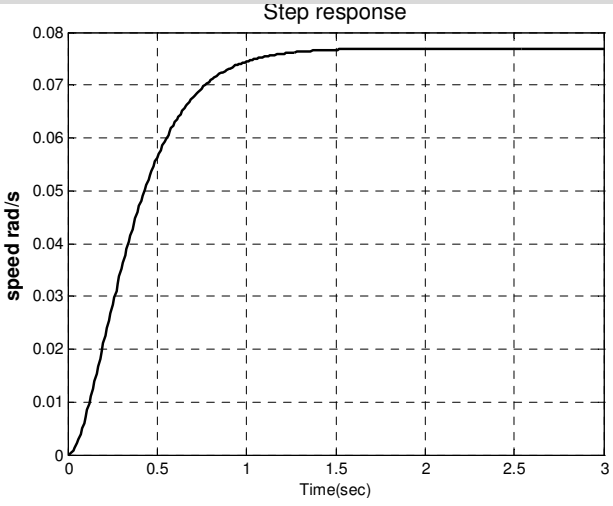
$$M = \exp(\frac{-\pi\zeta}{\sqrt{1-\zeta^2}}) \Rightarrow 0.001 = \exp(\frac{-\pi\zeta}{\sqrt{1-\zeta^2}}) \Rightarrow \zeta = 0.98$$

$$\frac{4.6}{\zeta\omega} = 1 \Rightarrow \omega \simeq 5$$

برای پیدا کردن مقدار k ،دستورات زیر را در ادامه m-file اضافه کنید و خروجی سیستم حلقه بسته را مشاهده کنید.

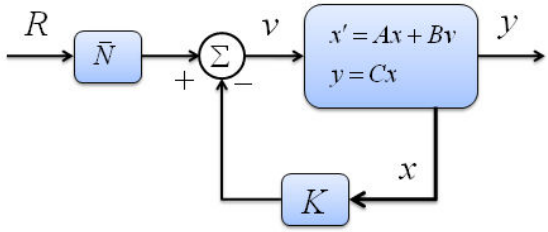
$$\dot{X} = (A-BK)X + B.R \qquad (۸-۲)$$
$$Y = CX$$

```
p1 = -5 + i; p2 = -5 - i;
K = place(A,B,[p1 p2]);
t=0:0.01:3;
step(A-B*K,B,C,D,1,t)
```



شکل (۲-۲۰) پاسخ پله برای سیستم با فیدبک حالت

در این قسمت چون ورودی مرجع نداریم خطای حالت ماندگار بسیار بزرگ است، بنابراین یک مرجع ثابت به ابتدای آن اضافه می‌کنیم.

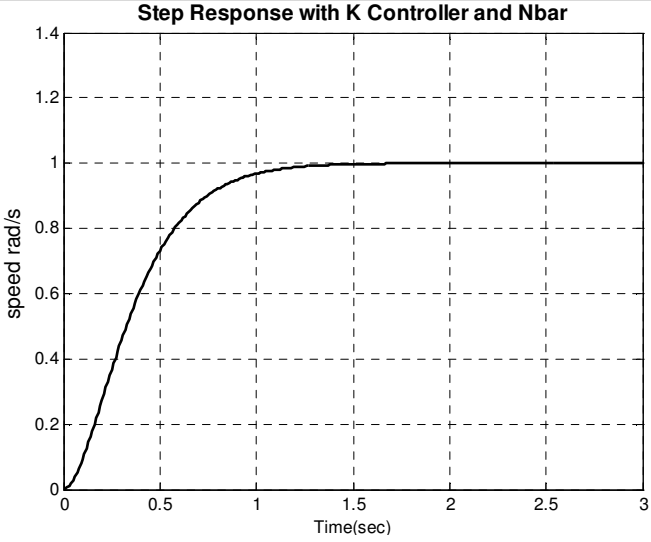


شکل (۲-۲۱) کنترل به روش فیدبک کلیه متغیرهای حالت همراه با ورودی مرجع

$$u = -kx + R\bar{N} \qquad (۹-۲)$$
$$\dot{x} = Ax + Bu \Rightarrow \dot{x} = (A - Bk)x + BR\bar{N}$$

مقدار $\bar{N} = G^{-1}(0) = 13$ به‌دست می‌آید.خروجی را دوباره مشاهده کنید:

```
t=0:0.01:3;
figure(5)
Nbar=13;
sys_N=ss(A-B*K,B*Nbar,C,D);
step(sys_N,1,t);
title('Step Response with K Controller and Nbar');
```



شکل (۲-۲۲) پاسخ سیستم به همراه کنترل فیدبک به مرجع ثابت

۲ – ۶ طراحی کنترلر دیجیتالی

در این قسمت قصد داریم یک کنترلر دیجیتال برای کنترل سرعت موتور DC طراحی کنیم. مدل موتور DC دیجیتالی از تبدیل مدل آنالوگی به‌دست می‌آید. در این قسمت، کنترلر PID دیجیتالی طراحی می‌شود.

اولین قدم در طراحی سیستم کنترل گسسته، تبدیل تابع تبدیل پیوسته به تابع تبدیل گسسته می‌باشد. برای این کار از دستور c2dm نرم افزار MATLAB استفاده کنید. این دستور به 4 آرگومان نیاز دارد: چند جمله‌ای صورت (num) ، چند جمله‌ای مخرج (den)، زمان نمونه برداری، و نوع مدار نگهدارنده. برای این مثال، نوع مدار را نگهدارنده مرتبه صفر[1] درنظر بگیرید ('zoh') و زمان نمونه برداری $T_s = 0.12\,\text{sec}$ را در نظر بگیرید که برابر 0.1 ثابت زمانی سیستم با زمان نشست 2 ثانیه می‌باشد.

```
R=1;L=0.5;Kt=0.01;J=0.01;b=0.1;
num = Kt;
den = [(J*L) (J*R)+(L*b) (R*b)+(Kt^2)];
Ts = 0.12;
[numz,denz] = c2dm(num,den,Ts,'zoh')
```

خروجی دستورات بالا به صورت زیر می‌باشد:

```
numz =
     0    0.0092    0.0057
denz =
    1.0000   -1.0877    0.2369
```

با توجه به ماتریس بالا تابع تبدیل دیجیتالی به صورت زیر نوشته می‌شود:

$$\frac{\dot{\theta}(z)}{V(z)} = \frac{0.0092z + 0.005}{z^2 - 1.0877z + 0.2369}$$

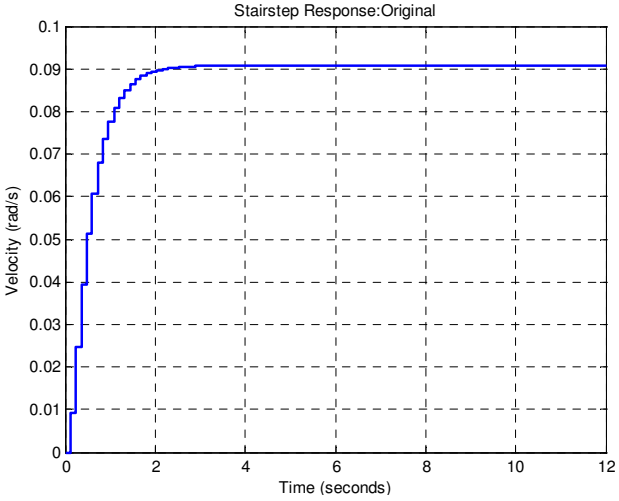
درابتدا ببینید پاسخ سیستم حلقه بسته بدون کنترلر به چه صورت می‌باشد. از روی تابع تبدیل متوجه می‌شوید که یک صفر در سیستم وجود دارد و قبل از حلقه بسته کردن سیستم با دستور Cloop، باید از دست این صفر خلاص شوید. بنابراین دستورات زیر را به ادامه M-file اضافه کنید:

```
numz = [numz(2) numz(3)];
[numz_cl,denz_cl] = cloop(numz,denz);
```

پس از انجام این کار ، ببینید پاسخ سیستم حلقه بسته به چه صورت می‌باشد. با استفاده از دستور dstep می‌توان سیگنال خروجی گسسته را تولید کرد. همچنین با استفاده از دستور stairs می‌توان سیگنال‌های خروجی را به هم پیوند داد. دستورات زیر را به ادامه M-file اضافه کنید و خروجی آن را مشاهده کنید.

```
[x1] = dstep(numz_cl,denz_cl,101);
t=0:0.12:12;
stairs(t,x1)
xlabel('Time (seconds)')
ylabel('Velocity (rad/s)')
title('Stairstep Response:Original')
```

[1] Zero order hold



شکل (۲-۲۳) پاسخ پله سیستم حلقه بسته گسسته

۲-۶-۱ طراحی کنترلر PID

یادآوری می‌کنیم که تابع تبدیل کنترلر PID به صورت زیر می‌باشد:

$$PID:\ k_p + k_D s + \frac{k_I}{s} = \frac{k_D s^2 + k_p s + k_I}{s}$$

راه‌های زیادی برای نگاشت از صفحه S به صفحه Z وجود دارد. یکی از دقیق‌ترین نگاشت‌ها $z = e^{-TS}$ می‌باشد. تابع تبدیل PID را نمی‌توان از این روش به‌دست آورد چون تابع تبدیل زمان گسسته آن تعداد صفرهای بیشتری نسبت به قطب‌ها داشته و این منطقی نیست. بنابراین از تبدیل $s = \frac{2}{T_s} \times \frac{z-1}{z+1}$ استفاده کنید و تابع تبدیل PID گسسته را به‌دست آورید، سپس مقادیر زیر را برای بهره‌ها در نظر بگیرید تا نیازهای طراحی برآورده شود.

$K_p = 100, K_I = 200, K_D = 10$

فرمان‌های زیر را به ادامه m-file اضافه کنید:

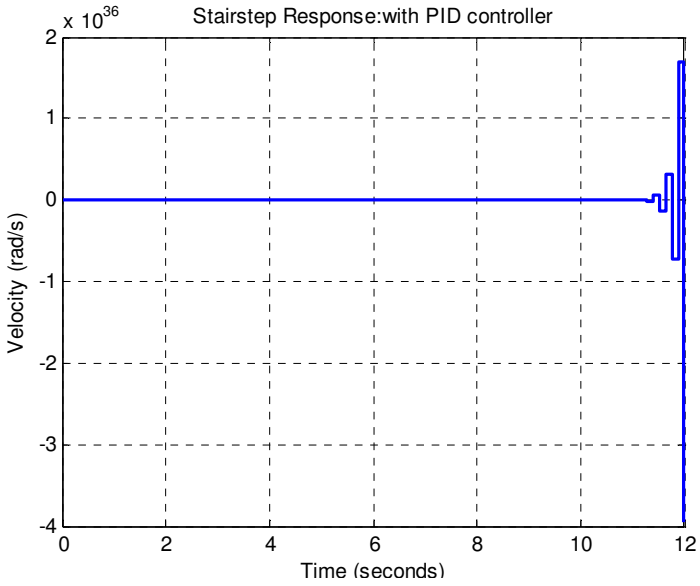
```
% Discrete PID controller with bilinear approximation
Kp = 100;Ki = 200; Kd = 10;
[dencz,numcz]=c2dm([1 0],[Kd Kp Ki],Ts,'tustin');
```

توجه کنید که صورت و مخرج در c2dm برعکس شده‌اند و دلیل آن این است که تابع تبدیل PID سره[1] نیست و MATLAB نیز این اجازه را نمی‌دهد. با برعکس کردن صورت و مخرج در دستور c2dm جواب درستی به ما داده نمی‌شود.

حال برای اینکه پاسخ سیستم با کنترل کننده PID را مشاهده کنید، دستورات زیر را به انتهای M-file خود اضافه نمائید.

[1] proper

```
numaz = conv(numz,numcz);
denaz = conv(denz,dencz);
[numaz_cl,denaz_cl] = cloop(numaz,denaz);
[x2] = dstep(numaz_cl,denaz_cl,101);
t=0:0.12:12;
stairs(t,x2)
xlabel('Time (seconds)')
ylabel('Velocity (rad/s)')
title('Stairstep Response:with PID controller')
```

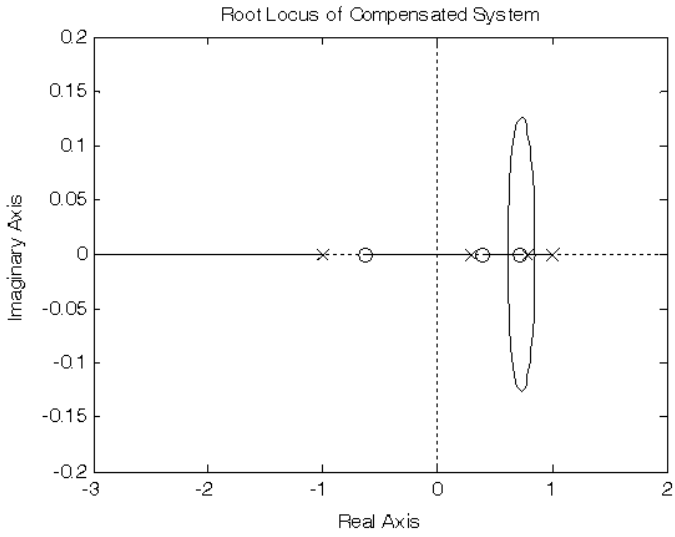


شکل (۲-۲۴) پاسخ پله سیستم با کنترلر PID دیجیتالی

از روی نمودار متوجه می‌شویم که سیستم ناپایدار است بنابراین یک مشکلی وجود دارد که با جبران کننده سیستم در تناقض است. بنابراین باید به مکان هندسی ریشه‌های سیستم جبران شده توجه کنید. بنابراین دستورات زیر را به ادامه M-file اضافه نمائید.

```
rlocus(numaz,denaz)
title('Root Locus of Compensated System')
```

از نمودار مکان هندسی ریشه‌ها متوجه می‌شوید که مخرج کنترلر PID یک قطب در 1- در صفحه z دارد و می‌دانید که اگر یک قطب در خارج دایره به شعاع 1 باشد سیستم ناپایدار خواهد بود.

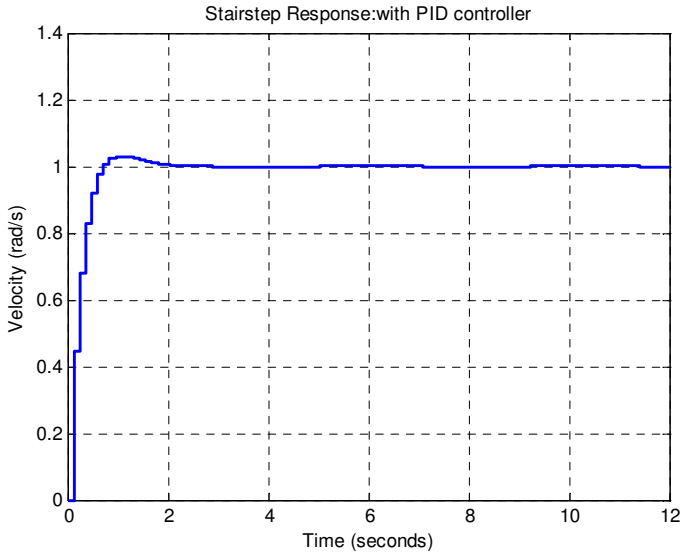


شکل (۲-۲۵) مکان هندسی ریشه‌ها

این سیستم جبران شده به ازای همه مقادیر بهره مثبت ناپایدار خواهد بود زیرا به تعداد زوج قطب و صفر در سمت راست 1- وجود دارد. بنابراین این قطب به سمت چپ حرکت می‌کند و درخارج دایره واحد قرار می‌گیرد. قطب‌های موجود در 1– از جبران کننده ناشی می‌شود بنابراین با تغییر دادن جبران کننده می‌توانید موقعیت قطب‌ها را تغییر بدهید. برای این کار قطب را در 0.62– در نظر بگیرید تا اثر صفر را حذف کند و باعث شود سیستم حداقل به ازای بعضی از بهره‌ها پایدار باشد. می‌توانید بهره‌های مناسب را از مکان هندسی ریشه‌ها به‌دست آورید، به گونه‌ای که ملزومات طراحی را برآورده سازد. دستورات زیر را به ادامه M-file اضافه کنید.

```
dencz = conv([1 -1],[1.6 1])
numaz = conv(numz,numcz);
denaz = conv(denz,dencz);

rlocus(numaz,denaz)
title('Root Locus of Compensated System');
[K,poles] = rlocfind(numaz,denaz)
[numaz_cl,denaz_cl] = cloop(K*numaz,denaz);

[x3] = dstep(numaz_cl,denaz_cl,101);
t=0:0.12:12;
stairs(t,x3)
xlabel('Time (seconds)')
ylabel('Velocity (rad/s)')
title('Stairstep Response:with PID controller')
```

dencz جدید یک قطب در $0.62-$ به‌جای $1-$ دارد که اثر صفر را در سیستم جبران نشده از بین می‌برد. در نتیجه پاسخ سیستم حلقه بسته جبران شده به صورت زیر در خواهد آمد:



شکل (۲-۴۶) پاسخ پله سیستم با کنترلر PID

از روی شکل واضح است که زمان نشست حدود 2 ثانیه و مقدار جهش حدود 2% است و مقدار خطای ماندگار صفر و بهره $k = 0.24$ نیز منطقی به نظر می‌رسد.

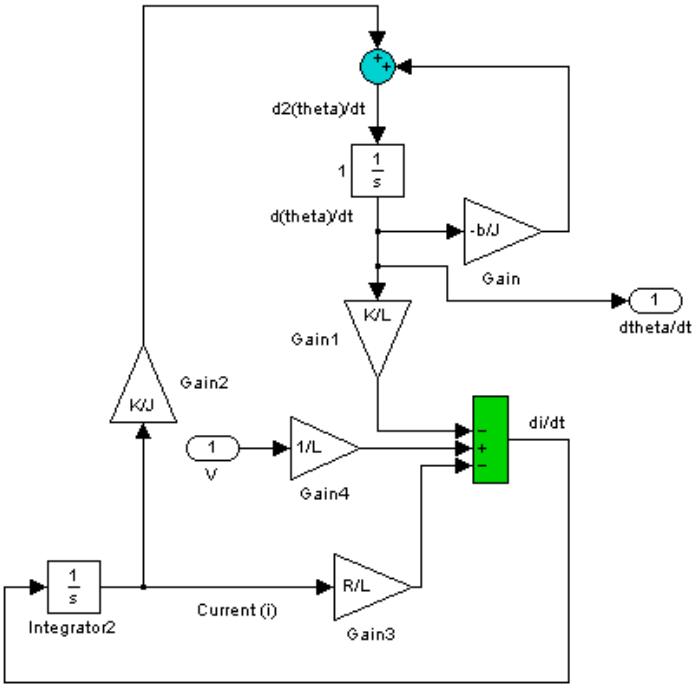
۲ – ۷ طراحی جبران کننده Lag در محیط سیمولینک

قوانین نیوتن و قوانین کیرشهف را برای مدل مورد بررسی بنویسید:

$$(1)\ T = J\ddot{\theta} + C_m\dot{\theta} \Rightarrow Ki = J\ddot{\theta} + C_m\dot{\theta} \Rightarrow \frac{d^2\theta}{dt^2} = \frac{K}{J}i - \frac{C_m}{J}\frac{d\theta}{dt}$$

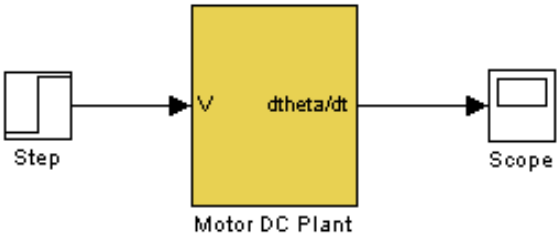
$$(2)\ e_a - e_b = Ri + L\frac{di}{dt} \Rightarrow V - K\dot{\theta} = Ri + L\frac{di}{dt} \Rightarrow \frac{di}{dt} = \frac{V}{L} - \frac{K}{L}\frac{d\theta}{dt} - \frac{R}{L}i$$

حال باید در محیط سیمولینک این مدل دینامیکی را بسازیم. پس مطابق شکل (۲-۲۷) بلوک‌های مورد نظر را از کتابخانه سیمولینک وارد پنجره مدل سازی کرده و با استفاده از سیگنال آنها را به هم ارتباط دهید.

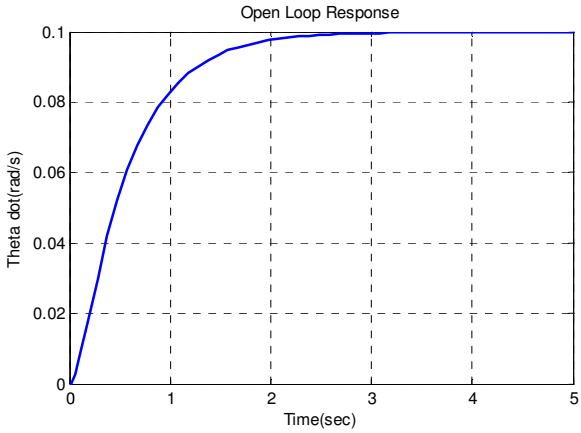


شکل(۲- ۲۷) مدل دینامیکی درمحیط سیمولینک

بعد از انجام مرحله مدل سازی سیستم باید پاسخ پله سیستم را به‌دست آوریم. برای انجام این کار تمام بلوک های کشیده شده در شکل قبل را انتخاب کنید و با کلیک راست کردن بر روی صفحه و انتخاب subsystem ، بلوک‌ها را در داخل یک زیر سیستم قرار دهید. بعد از اینکه مدل سیستم را شبیه سازی کردیم باید پاسخ سیستم حلقه باز را بررسی کنیم. بنابراین در ورودی سیستم از ورودی پله و در خروجی یک Scope قرار دهید.

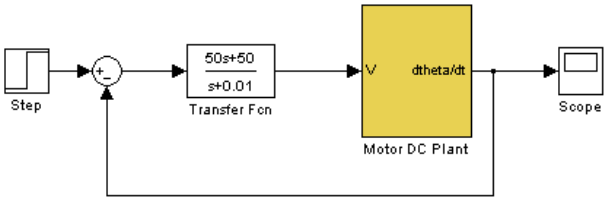


بعد از اجرای برنامه، خروجی زیر حاصل می‌شود:

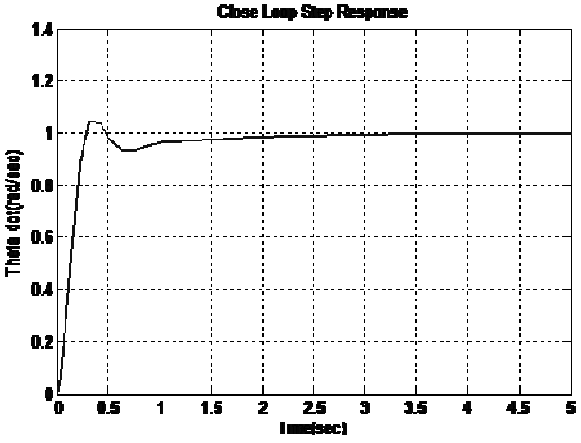


شکل(۲– ۲۸) پاسخ حلقه باز

مقدار خطای حالت ماندگار بسیار زیاد است، پس نیاز داریم که از یک جبران کننده Lag استفاده کنیم. بلوک Transfer Function را از کتابخانه سیمولینک به محیط مدل سازی می‌کشیم و سیستم حلقه بسته زیر را تکمیل می‌کنیم.



بعد از اجرای برنامه خروجی زیر حاصل می‌شود. همانطور که انتظار داشتیم مقدار خطای حالت ماندگار به صفر رسیده است.



شکل(۲– ۲۹) پاسخ حلقه بسته با کنترلر Lag

پروژه سوم

کنترل موقعیت موتور DC

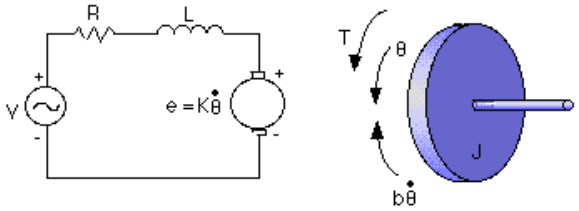
اهداف این پروژه

- مدل سازی
- طراحی کنترلر PID
- رسم مکان هندسی ریشه‌ها (Root Locus)
- پاسخ فرکانسی
- طراحی فضای حالت
- طراحی کنترلر دیجیتالی
- طراحی کنترلر دیجیتال در محیط سیمولینک

۳–۱ مدل‌سازی

۳-۱-۱ مدل‌سازی سیستم براساس اصول فیزیکی و استخراج معادلات سیستم

یکی از محرک‌های رایج در سیستم‌های کنترلی موتور DC است که به طور مستقیم می‌تواند حرکت دورانی و به طور غیرمستقیم حرکت خطی تولید کند. مدار الکتریکی آرمیچر و نمودار آزاد روتور در شکل (۳-۱) نمایش داده شده است.



شکل (۳-۱) موتور DC

در این مساله فرض کنید مقادیر زیر را برای پارامترهای فیزیکی انتخاب کرده‌ایم. این مقادیر از یک آزمایش واقعی استخراج شده اند.

ممان اینرسی روتور: $J = 3.228\times10^{-6}\ \frac{kg.m^2}{s^2}$

ضریب میرایی سیستم مکانیکی: $C_m = 3.5\times10^{-6}\ Nm.s$

ثابت نیروی الکترو موتوری: $K = K_e = K_t = 0.0274 \frac{N.m}{Amp}$

مقاومت الکتریکی : $R = 4\Omega$

اندوکتانس الکتریکی: $L = 2.75 \times 10^{-6}\ H$

ورودی (V) : ولتاژ منبع تغذیه

خروجی θ: موقعیت شفت

فرض بر این است که شفت و روتور صلب می‌باشند.

با توجه به شکل (۳-۱)، گشتاور موتور با جریان ورودی موتور رابطه مسقیم دارد، یعنی $T = k_t i_a$ و نیروی پس-زن الکترموتوری با سرعت موتور رابطه مستقیم دارد ($e_b = k_b \dot{\theta} = EMF$). در سیستم SI دو ضریب K_t ، K_e با هم برابرند ($K_e = K_t = K$). اگر قوانین نیوتن و قوانین کیرشهف را برای مدار شکل (۳-۱) بنویسید، داریم:

$$\begin{cases} e_a - e_b = Ri + L\dfrac{di}{dt} \\ T = J\ddot{\theta} + C_m\dot{\theta} \end{cases} \Rightarrow \begin{cases} V - K\dot{\theta} = Ri + L\dfrac{di}{dt} \\ Ki = J\ddot{\theta} + C_m\dot{\theta} \end{cases} \quad (۱-۳)$$

۳-۱-۲ تابع تبدیل سیستم

برای به‌دست آوردن تابع تبدیل سیستم نیاز داریم از معادله مدل سازی (۳-۱) تبدیل لاپلاس بگیریم. توجه کنید زمانی که در حال پیدا کردن تابع تبدیل سیستم هستیم شرایط اولیه را برابر صفر در نظر می‌گیریم.

$$\begin{cases} V - K.s.\theta(s) = RI(s) + Ls.I(s) \\ KI(s) = Js^2\theta(s) + C_m s.\theta(s) \end{cases} \quad (۲-۳)$$

$$\xrightarrow{Elim inating\ I(s)} V - Ks\theta(s) = \frac{Js^2\theta(s) + C_m s.\theta(s)}{K}(R + Ls) \quad (۳-۳)$$

$$\frac{\omega(s)}{V(s)} = \frac{K}{(Js + C_m)(R + Ls) + K^2} \Rightarrow \frac{\theta(s)}{V(s)} = \frac{K}{s((Js + C_m)(R + Ls) + K^2)} \quad (۴-۳)$$

۳-۱-۳ مدل فضای حالت سیستم

دراین مساله جریان موتور، موقعیت، و سرعت دورانی روتور را به عنوان متغیرهای حالت در نظر بگیرید. بنابراین معادله بالا را می‌توان به صورت زیر نوشت:

$$\begin{cases} \dot{x}(t) = Ax(t) + Bu(t) \\ y(t) = Cx(t) + Du(t) \end{cases} \Rightarrow \begin{cases} \dfrac{d}{dt}\begin{bmatrix} \theta \\ \dot{\theta} \\ i \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & \dfrac{-C_m}{J} & \dfrac{K}{J} \\ 0 & \dfrac{-K}{L} & \dfrac{-R}{L} \end{bmatrix}\begin{bmatrix} \theta \\ \dot{\theta} \\ i \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \dfrac{1}{L} \end{bmatrix} V \\ \theta' = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}\begin{bmatrix} \theta \\ \dot{\theta} \\ i \end{bmatrix} \end{cases} \qquad (۵-۳)$$

۳-۱-۴ قیدهای حاکم بر طراحی

قدم بعدی در مدل سازی این است که بدانیم این سیستم قرار است چه شرایطی را داشته باشد، یعنی قیدهای حاکم بر سیستم باید مشخص شود. برای این سیستم می‌خواهیم موتور با دقت زیاد در یک موقعیت مشخص قرار بگیرد، بنابراین برای ورودی مرجع 1rad باید:

- خطای حالت ماندگار سیستم برابر صفر باشد.
- خطای حالت ماندگار ناشی از اغتشاشات نیز برابر صفر باشد.
- موتور به‌سرعت به موقعیت نهایی برسد، پس زمان رسیدن به‌سرعت ماندگار باید کمتراز 40ms باشد پس $Settling\ time < 40\text{msec}$.
- مقدار جهش کمتر از 16% باشد: $Overshoot < 16\%$

۳-۱-۵ پاسخ سیستم حلقه باز

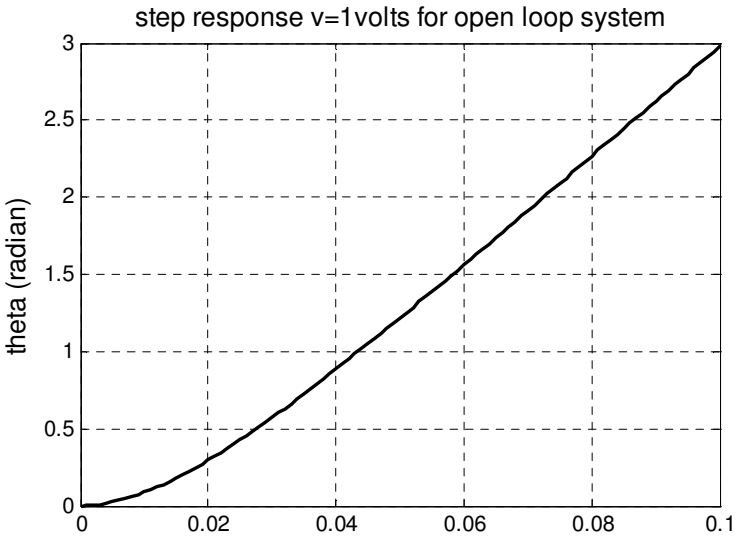
حال بهتر است اجازه دهید پاسخ سیستم حلقه باز را به ورودی پله بررسی کنیم.

$$num = K$$
$$den = s((Js + C_m)(R + Ls) + K^2) \qquad (۶-۳)$$

دستورات زیر را در یک m-file وارد کنید:

```
%~~~~~~~~~ open loop ~~~~~~~~~~~
J=3.2284E-6;
Cm=3.5077E-6;
K=0.0274;
R=4;L=2.75E-6;
num=K;
t=0:0.001:0.2;
den=[(J*L) ((J*R)+(L*Cm)) ((Cm*R)+K^2) 0];
sys=tf(num,den);
step(sys,0:0.001:0.1);
title('step response v=1volts for open loop system')
ylabel('theta (radian)')
```

پاسخ سیستم به صورت زیر خواهد بود:



شکل (۳-۲) پاسخ پله سیستم حلقه باز

از روی شکل مشخص است که سیستم کنترلی معیارهای طراحی را برآورده نمی‌کند. مثلاً به ازای ورودی 1 ولتی انتظار داریم موتور 1 رادیان بچرخد ولی 3 رادیان چرخیده است، یعنی 3 برابر موقعیت مطلوب. پس نیاز به طراحی یک سیستم کنترل حلقه بسته برای این سیستم، احساس می‌شود.

۳-۱-۶ پاسخ حلقه باز با استفاده از معادلات حالت

معادلات حالت در محیط MATLAB به صورت زیر نوشته می‌شوند.

```
A=[0 1 0
   0 -Cm/J K/J
   0 -K/L -R/L];
B=[0 ; 0 ; 1/L];
C=[1 0 0];
D=[0];
step(A,B,C,D)
```

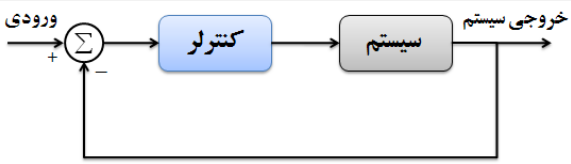
۳-۲ طراحی کنترلر PID

از معادله (۳-۴) تابع تبدیل سیستم به صورت زیر به‌دست آمد:

$$\frac{\theta(s)}{V(s)} = \frac{K}{s((Js + C_m)(R + Ls) + K^2)}$$

بلوک نمودار سیستم کنترلی با فیدبک واحد در شکل (۳-۳) نمایش داده شده است. می‌دانیم تابع تبدیل کنترلر PID به صورت زیر است:

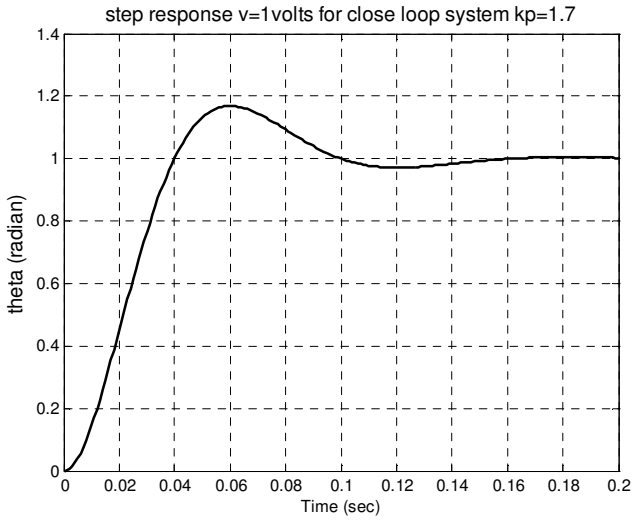
$$PID: k_p + k_D s + \frac{k_I}{s} = \frac{k_D s^2 + k_p s + k_I}{s} \qquad (۷-۳)$$



شکل (۳-۳) بلوک نمودار سیستم کنترلی با فیدبک واحد

در ابتدا یک کنترلر تناسبی قرار می‌دهیم. اولین کاری که باید انجام دهیم این است که تابع تبدیل سیستم حلقه بسته با کنترلر تناسبی(k_p) را به‌دست آوریم . مقدار $k_p = 1.7$ را قرار می‌دهیم.

```
%========== P controller ==========
kp=1.7;
numa=kp*num;
dena=den;
[numc,denc]=cloop(numa,dena);
sys_p=tf(numc,denc);
figure(2)
step(sys_p,0:0.001:0.2);
title('step response v=1volts for close loop system kp=1.7')
```

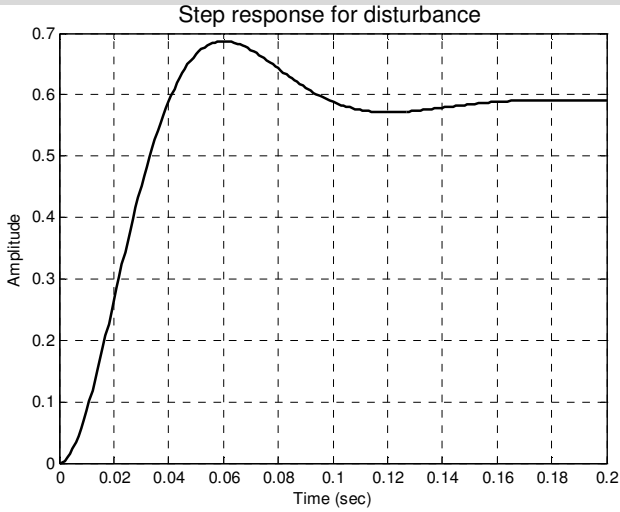


شکل (۳-۴) پاسخ پله سیستم همراه با کنترل تناسبی

حال بهتر است پاسخ سیستم را به اغتشاش پله بررسی کنیم. بنابراین دستورات زیر را به ادامه m-file قبلی اضافه کنید:

```
%======== Step response for disturbance ========
figure(3)
t=0:0.001:0.2;
```

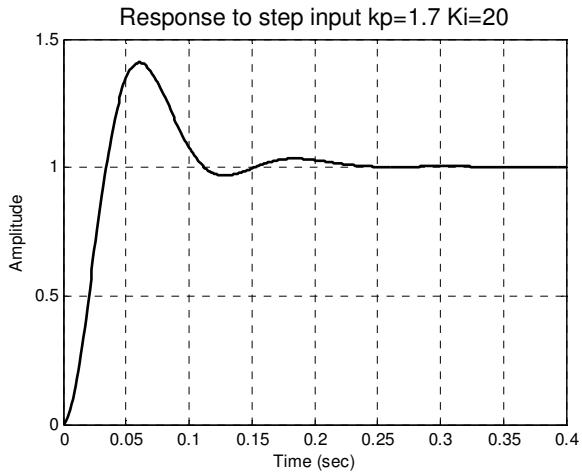
```
numdcl=conv(numc,1);
dendcl=conv(denc,kp);
sys_dis=tf(numdcl,dendcl)
step(sys_dis,t);
title('Step response for disturbance')
```



شکل (۳-۵) پاسخ سیستم به اغتشاش پله

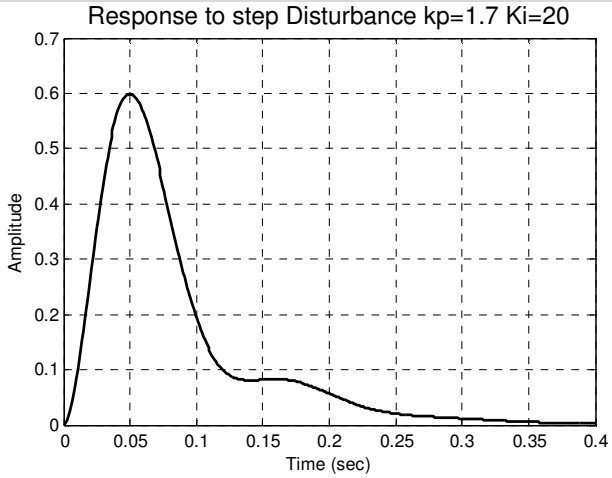
همانطور که از نمودارهای بالا بر می‌آید هرچند خطای حالت ماندگار برای ورودی پله مناسب است، ولی مقدار زمان نشست طولانی می‌باشد. از طرفی با همین کنترلر، برای اغتشاش پله مقدار خطای حالت ماندگار بزرگ است و باید به مقدار صفر برسد. برای کاهش خطای ماندگار از کنترلر انتگرالی و برای کاهش مقدار جهش از کنترلر مشتق‌گیر استفاده می‌کنیم. در ابتدا یک کنترلر PI طراحی می‌کنیم تا مقدار خطای حالت ماندگار ناشی از اغتشاش را به صفر کاهش دهیم.

```
Kp=1.7;
Ki=20;
numcf=[Kp Ki];
dencf=[1 0];
numf=conv(numcf,num);
denf=conv(dencf,den);
[numc,denc]=cloop(numf,denf,-1);
t=0:0.001:0.4;
step(numc,denc,t)
title('Response to step input kp=1.7 Ki=20')
```



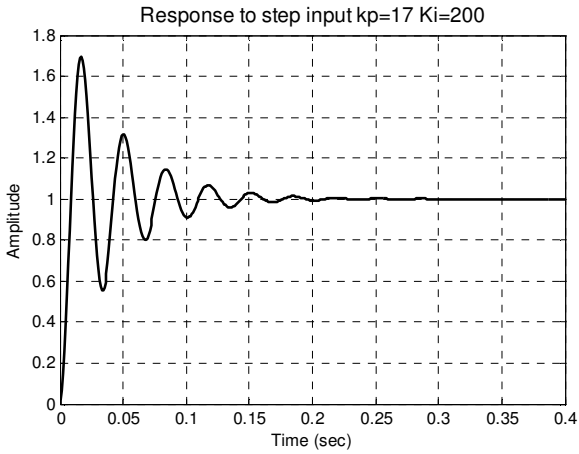
شکل (۳-۶) پاسخ پله سیستم همراه با کنترل تناسبی و انتگرالی

```
figure
numdcl=conv(numc,dencf);
dendcl=conv(denc,numcf);
step(numdcl,dendcl,t);
title('Response to step Disturbance kp=1.7 Ki=20')
```

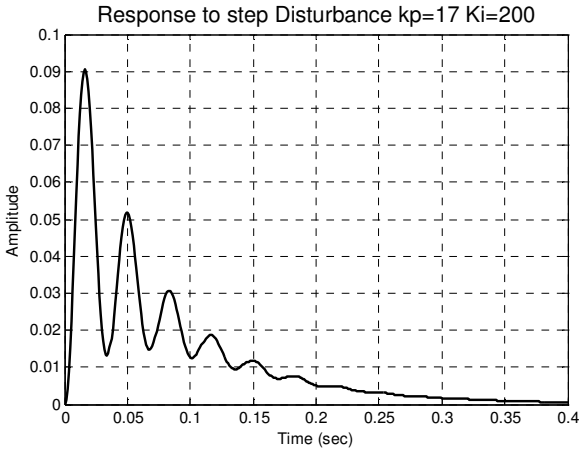


شکل (۳-۷) پاسخ سیستم به اغتشاش پله با کنترلر تناسبی و انتگرالی

تنظیم بهره‌ها: در این مرحله باید بهره‌ها را تنظیم کنیم. از روی نمودار پاسخ به ورودی پله می‌بینیم که زمان نشست تا حدودی طولانی است. بنابراین برای افزایش سرعت پاسخ گویی، مقدار بهره تناسبی و انتگرالی را افزایش می‌دهیم.



شکل (۳-۸) پاسخ پله سیستم همراه با کنترل تناسبی و انتگرالی

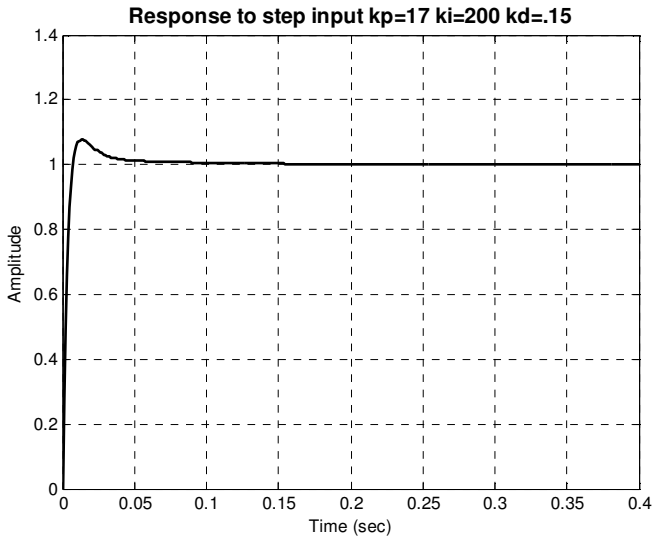


شکل (۳-۹) پاسخ سیستم به اغتشاش پله با کنترلر تناسبی و انتگرالی

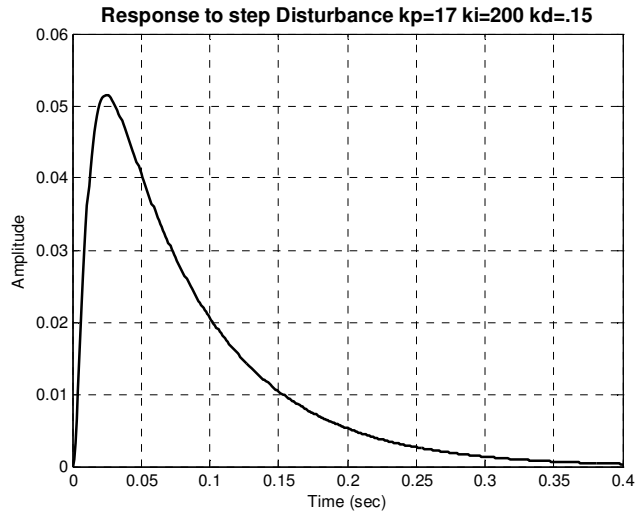
از روی پاسخ‌ها مشخص است که سرعت پاسخ گویی نسبت به قبل افزایش پیدا کرده است ولی به دلیل افزایش مقدار بهره انتگرالی مقدار جهش بیشتر شده است. بنابراین برای کاهش این جهش‌ها از یک کنترلر مشتق‌گیر استفاده می‌کنیم.

```
Kp=17;
Ki=200;
Kd=0.15;
numcf=[Kd Kp Ki];
dencf=[1 0];
numf=conv(numcf,num);
denf=conv(dencf,den);
[numc,denc]=cloop(numf,denf,-1);
t=0:0.001:0.1;
figure(4)
```

```
step(numc,denc,t);
title('Response to step input kp=17 ki=200 kd=0.15’)

numdcl=conv(numc,dencf);
dendcl=conv(denc,numcf);
figure(5)
step(numdcl,dendcl,t);
title(‘Response to step Disturbance kp=17 ki=200 kd=.15’)
```

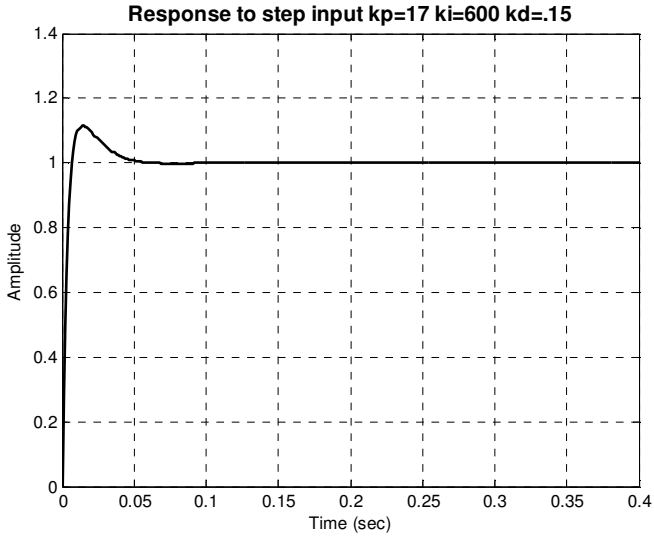


شکل (۳-۱۰) پاسخ پله سیستم همراه با کنترل PID

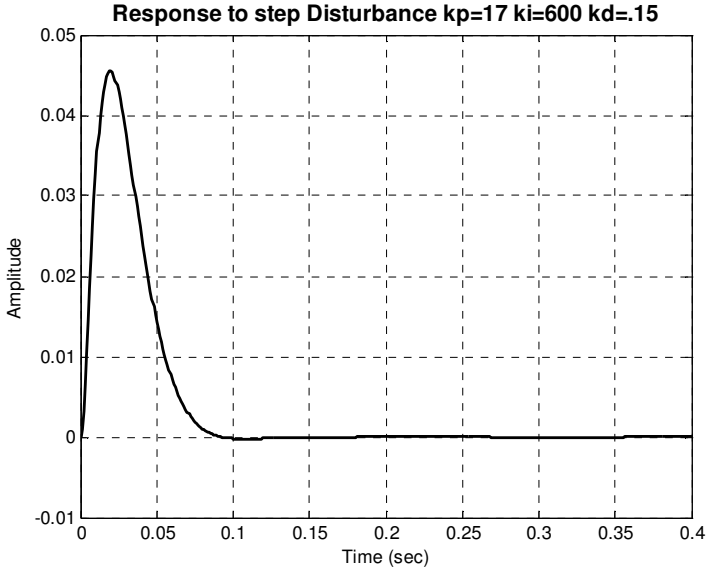


شکل (۳-۱۱) پاسخ سیستم به اغتشاش پله با کنترلر PID

نیاز داریم که پاسخ به اغتشاش پله تا حدودی سریع تر شود.



شکل (۳-۱۲) پاسخ پله سیستم همراه با کنترل PID



شکل (۳-۱۳) پاسخ سیستم به اغتشاش پله با کنترلر PID

۳-۳ رسم مکان هندسی ریشه‌ها (Root Locus)

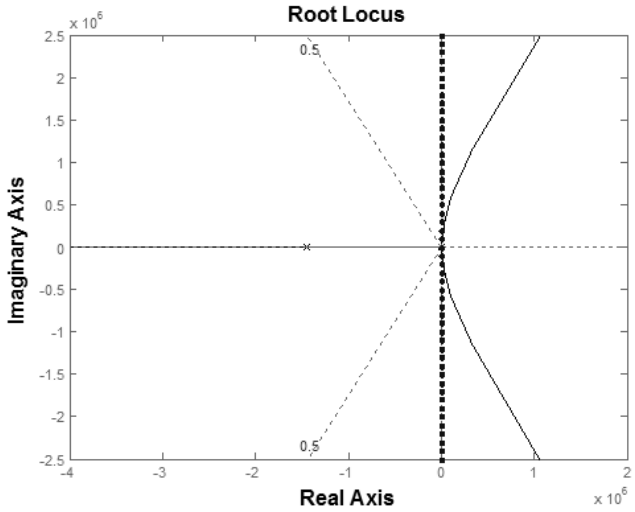
ایده اصلی در طراحی به کمک مکان هندسی ریشه‌ها، پیدا کردن پاسخ سیستم حلقه بسته از روی نمودار مکان هندسی ریشه‌های سیستم حلقه باز می‌باشد، به طوری که با اضافه کردن یک سری صفر و قطب به سیستم اصلی پاسخ سیستم حلقه بسته تصحیح شود. پس فرامین زیر را در یک m-file جدید بنویسید:

```
J=3.2284E-6;
Cm=3.5077E-6;
K=0.0274;
R=4;
L=2.75E-6;
num=K;
den=[(J*L)  ((J*R)+(L*Cm))  ((Cm*R)+K^2)  0];
```

حال برای به‌دست آوردن مکان هندسی ریشه‌ها در حالت حلقه باز دستورات زیر را وارد کنید:

```
rlocus(num,den)
sgrid(.5,0)
sigrid(100)
```

دستورات Sigrid و sgrid هر دو تابع می‌باشند. sgrid جزء دستورات MATLAB است ولی sigrid از جمله دستورات MATLAB نمی باشد. دستور sgrid به دو آرگومان نیاز دارد: نسبت میرایی $\zeta = 0.5$ که متناسب با جهش 16% است و فرکانس طبیعی $\omega_n = 0$ (بدون در نظر گرفتن معیار زمان خیز). دستور Sigrid نیاز به یک آرگومان دارد ($\zeta\omega_n = \frac{4.6}{T_s} = \frac{4.6}{0.04} \simeq 100$). با اجرای دستورات بالا به شکل (۳-۱۴) می-رسید:

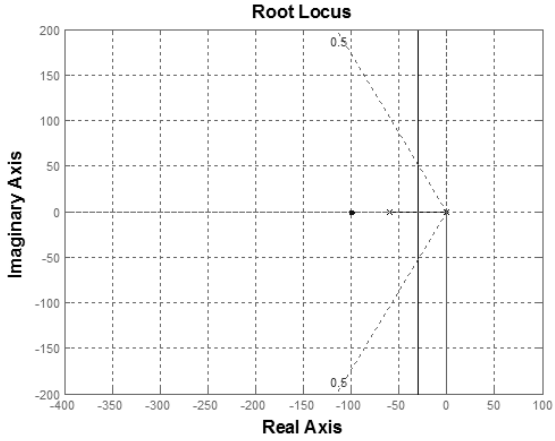


شکل (۳-۱۴) مکان هندسی ریشه‌ها

اگر به مقیاس محورها توجه کنید می‌بینید که یکی از قطب‌ها بسیار دور از محور موهومی قرار گرفته است (دورتر از -10^6). این قطب تاثیر زیادی روی دینامیک سیستم حلقه بسته ندارد مگر اینکه بهره به اندازه کافی بزرگ باشد؛ که در این صورت هم سیستم ناپایدار خواهد شد. ما از این قطب چشم‌پوشی کرده و در ناحیه‌ای خاص متمرکز می‌شویم. دستور زیر را به ادامه m-file اضافه کنید:

```
axis([-400 100 -200 200])
```

همانطور که در شکل(۳-۱۵) مشاهد می‌شود سیستم حلقه بسته به ازای بهره‌های پائین پایدار خواهد بود.



شکل (۳-۱۵) مکان هندسی ریشه‌ها با بهره K

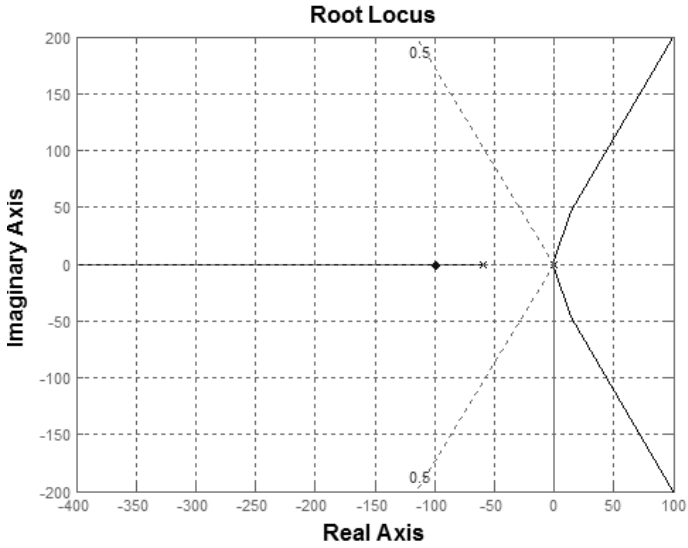
همان طور که از شکل بر می‌آید قطب‌های سیستم‌های حلقه بسته به اندازه کافی سریع نیستند تا بتوانند زمان نشست ما را برآورده کنند (این قطب‌ها هرگز نباید به سمت چپ خط عمودی 100 بروند). همچنین نیاز به یک انتگرال گیر داریم تا خطای حالت ماندگار ناشی از اغتشاشات از بین برود.

۳-۳-۱کنترلر انتگرالی

اکنون اجازه بدهید تا از یک کنترل انتگرالی برای حذف اثر اغتشاشات استفاده کنیم. برنامه m-file را به‌صورت زیر ویرایش کنید:

```
J=3.2284E-6;
Cm=3.5077E-6;
K=0.0274;R=4;L=2.75E-6;
num=K;
den=[(J*L) ((J*R)+(L*Cm)) ((Cm*R)+K^2) 0];
numcf=[1];
dencf=[1 0];
numf=conv(numcf,num);
denf=conv(dencf,den);
rlocus(numf,denf)
sgrid(.5,0)
sigrid(100)
axis([-400 100 -200 200])
```

نمودار حاصل به صورت زیر خواهد بود:



شکل (۳-۱۶) مکان هندسی ریشه‌ها با کنترلر انتگرالی

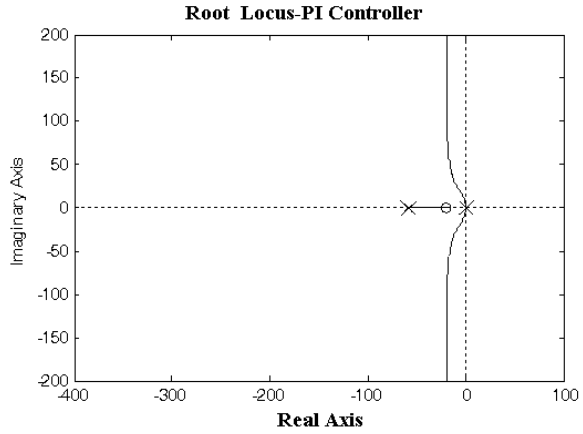
از روی شکل مشخص است که با کنترل انتگرالی نیز سیستم ناپایدار است و باید از کنترلر دیگری استفاده کرد.

۳-۳-۲ کنترلر انتگرالی و تناسبی

حال باید کنترل انتگرالی را به کنترلر PI تبدیل کنیم، برای این تبدیل باید یک صفر به سیستم حلقه باز اضافه کنیم. این صفر را در s=-20 قرار می‌دهیم. این صفر باید بین قطب‌های حلقه باز سیستم قرار بگیرد تا سیستم حلقه بسته پایدار شود. در دستورات M-file بالا خطوط زیر را اصلاح کنید:

```
numcf=[1 20];dencf=[1 0];
```

نمودار حاصل به صورت زیر خواهد بود:



شکل (۳-۱۷) مکان هندسی ریشه‌ها با کنترلر انتگرالی + تناسبی

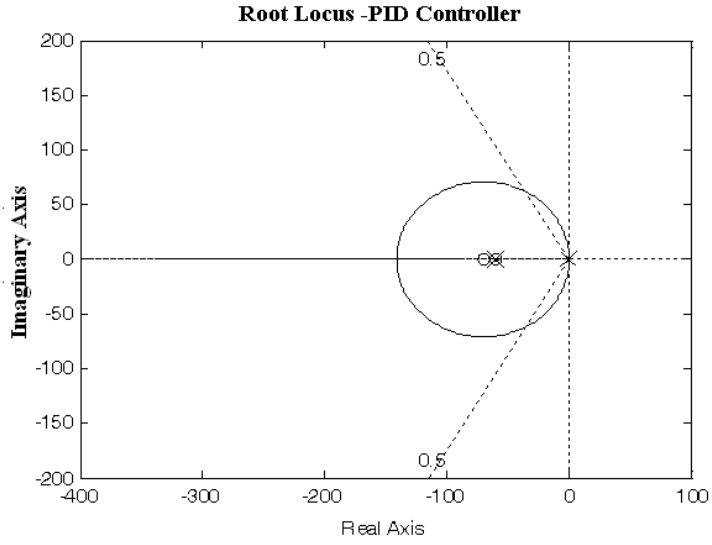
تا اینجا پایداری سیستم و خطای حالت ماندگار صفر به اغتشاش پله را مدیریت کرده‌ایم ولی سیستم به اندازه کافی سریع نیست.

۳-۳-۳ کنترل PID

به منظور افزایش سرعت سیستم حلقه بسته و برای کشیدن بیشتر مکان هندسی ریشه‌ها به سمت چپ، باید دومین صفر حلقه باز را نیز قرار دهیم که نتیجه یک کنترلر PID می‌شود. بعد از چند بار سعی و خطا صفرها را در موقعیت‌های $s=-60, s=-70$ قرار می‌دهیم. در دستورات M-file بالا خطوط زیر را اصلاح کنید:

```
numcf=conv([1 60],[1 70]);
dencf=[1 0];
```

با اجرای دوباره برنامه، خروجی زیر حاصل می‌شود:



شکل(۳-۱۸) مکان هندسی ریشه‌ها با کنترلر PID

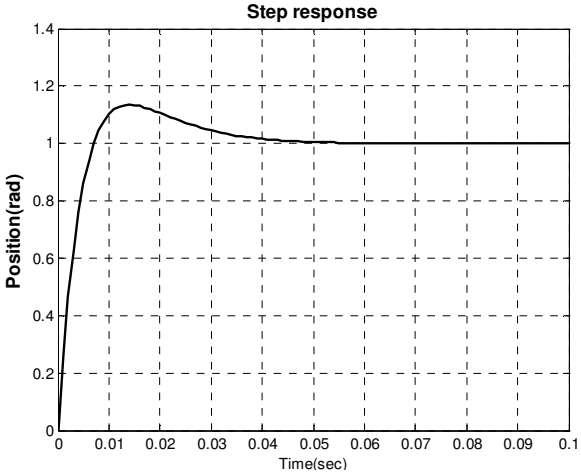
۴-۳-۳ پیدا کردن بهره با استفاده از دستور rlocfind

از آنجایی که ما می‌خواهیم زمان نشست و حداکثر جهش کوچکی داشته باشیم، نیاز به میرایی زیادی داریم، بنابراین باید نقطه‌ای نزدیک محور حقیقی را انتخاب کنیم.همچنین، برای اینکه جواب نسبتاً سریعی داشته باشیم باید تا حد امکان از محور موهومی فاصله بگیریم. برای پیدا کردن بهرهٔ متناظر با این نقاط در مکان هندسی ریشه‌ها، می‌توانیم از دستور rlocfind استفاده کنیم. پس می‌توانیم این بهره را پیدا کرده و پاسخ پله را با استفاده از این بهره رسم کنیم. بنابراین دستورات زیر را به ادامه M-file اضافه کنید:

```
[k,poles] = rlocfind(numf,denf)
[numc,denc]=cloop(k*numf,denf,-1);
t=0:0.001:.1;
step(numc,denc,t)
```

در نمودار، بر روی یک نقطه روی مکان هندسی ریشه‌ها داخل حلقه و سمت چپ و نزدیک محور حقیقی کلیک کنید. این قطب‌ها باعث می‌شوند که جهش کمی داشته باشیم. ولی به این نکته توجه کنید که اضافه کردن صفر باعث افزایش جهش می‌شود.

Select a point in the graphics window
selected_ point = -1.2106e+002 +1.0728e+001i

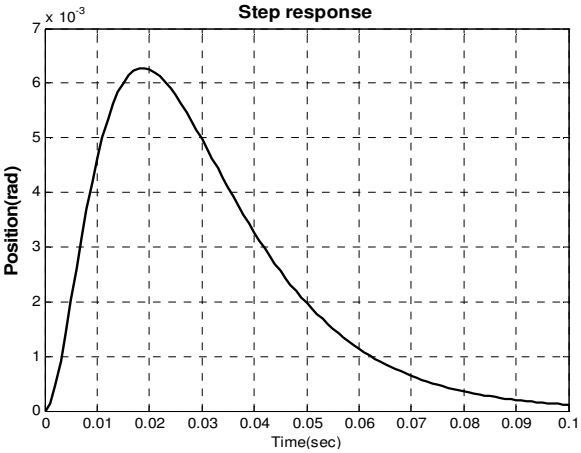


شکل (۳-۱۹) پاسخ پله سیستم حلقه بسته با بهره انتخابی

همانطور که از شکل مشخص است مقدار جهش حدود 15% بوده و مقدار زمان نشست حدود 0.04 ثانیه است، ضمن این که خطای حالت ماندگار صفر است. حال به پاسخ سیستم به اغتشاشات وارده توجه کرده و پاسخ پله را رسم کنید. بنابراین دستورات زیر را به ادامه M-file اضافه کنید:

```
numdcl=conv(numc,dencf);
dendcl=conv(denc,numcf);
step(numdcl,dendcl,t);
```

برنامه را دوباره اجرا نموده، و همان نقاط قبلی را انتخاب کنید. به نمودار زیر خواهید رسید:



شکل (۳-۲۰) پاسخ سیستم حلقه بسته با بهره انتخابی به اغتشاشات پله

همانطور که می‌بینید خطای حالت ماندگار ناشی از اغتشاشات به مقدار صفر می‌رسد و تمام معیارهای طراحی ما برآورده می‌شود.

۳-۴ پاسخ فرکانسی

از قسمت مدل سازی به یاد داریم که تابع تبدیل سیستم به صورت زیر است :

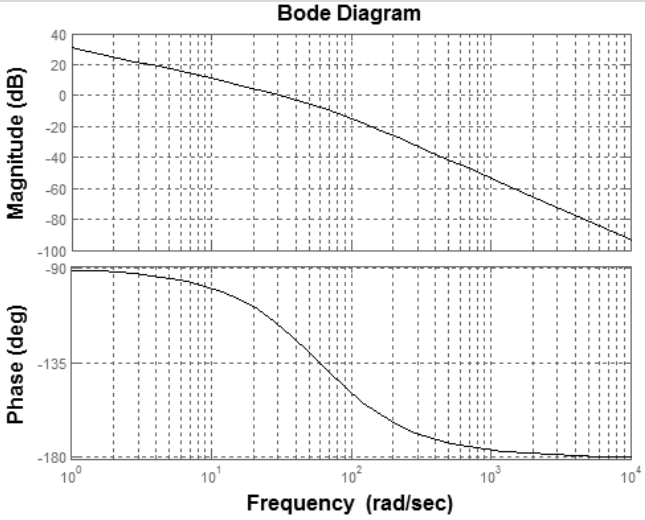
$$\frac{\theta}{V} = \frac{K}{s((Js+C_m)(R+Ls)+K^2)}$$

هدف از طراحی این است که زمان نشستی کمتر از 40 ثانیه، جهشی کمتر از 16%، و خطای حالت ماندگار صفر در برابر ورودی پله و اغتشاش پله داشته باشیم.

۳-۴-۱ رسم نمودار بود

ایده اصلی طراحی برمبنای پاسخ فرکانسی بر این است که با رسم نمودار بود[1] تابع تبدیل حلقه باز، تخمینی از پاسخ سیستم حلقه بسته به‌دست آوریم. اضافه کردن کنترلر به سیستم، نمودار بود حلقه باز را تغییر می‌دهد؛ بنابراین پاسخ سیستم حلقه بسته را نیز تغییر می‌دهد. اجازه بدهید که در ابتدا نمودار بود تابع تبدیل حلقه باز را رسم کنیم. پس تابع تبدیل را در یک m-file وارد کنید:

```
J=3.2284E-6;Cm=3.5077E-6;
K=0.0274;R=4;L=2.75E-6;
num=K;
den=[(J*L)  ((J*R)+(L*Cm))  ((Cm*R)+K^2)  0];
bode(num,den)
```

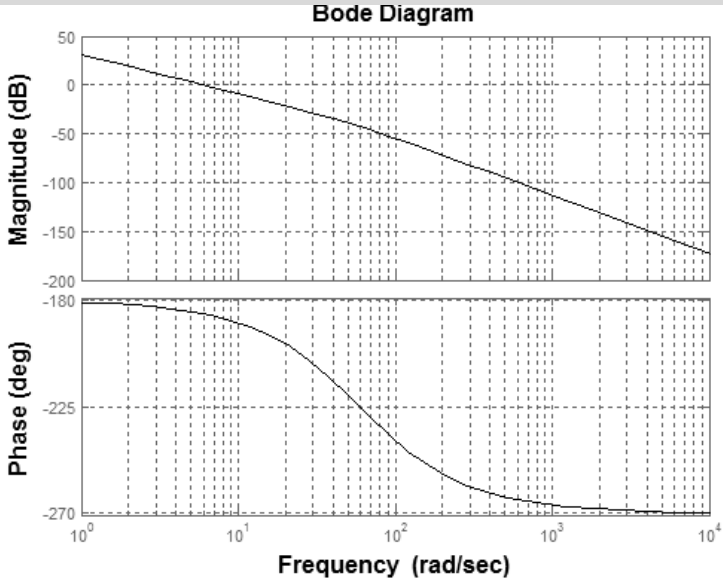


شکل (۳-۲۱) نمودار بود سیستم حلقه باز

[1] Bode diagram

اجازه بدهید که یک انتگرال‌گیر برای صفر کردن خطای حالت ماندگار به ورودی اغتشاش پله‌ای اضافه کنیم. دستورات زیر را به m-file اضافه کنید:

```
numi=1;
deni=[1 0];
numiol=conv(num,numi);
deniol=conv(den,deni);
bode(numiol,deniol)
```



شکل (۳-۲۲) نمودار بود سیستم با اضافه شدن یک انتگرال‌گیر

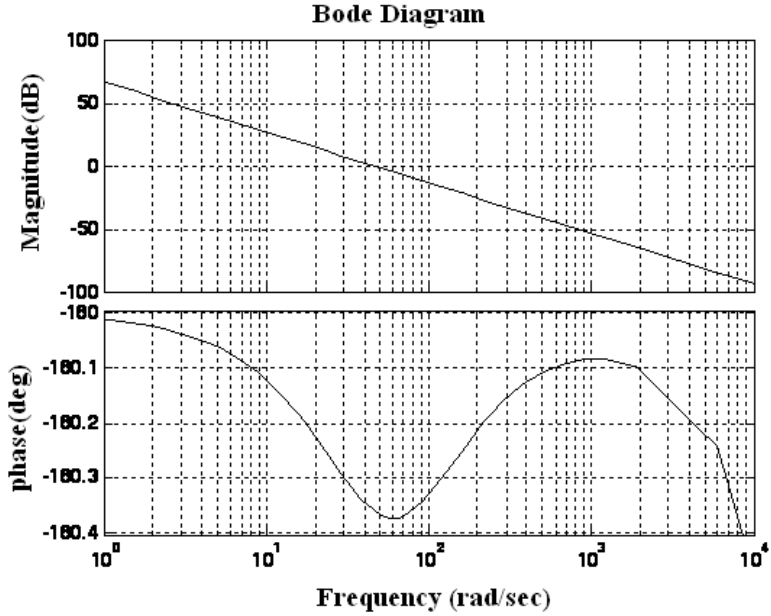
۳-۴-۲ مشخصات حاشیه فاز و بهره و طراحی کنترلر

می‌خواهیم جهشی کمتر از 16% داشته باشیم، پس اجازه بدهید که ثابت میرایی متناسب با آن را به‌دست آوریم. از طرفی حد فاز متناظر با آن 100 برابر ضریب میرایی است. ازسوی دیگراز روی زمان نشست می‌توانیم پهنای باند فرکانسی را نیز محاسبه کنیم:

```
zeta=-log(.16)/sqrt(pi^2+(log(.16))^2);
PM=100*zeta;
wbw=(4/(0.04*zeta))*sqrt((1-2*zeta^2)+sqrt(4*zeta^4-
4*zeta^2+2));
w=logspace(0,4,101);
```

می‌خواهیم حداقل 50 درجه حد فاز داشته باشیم ، بنابراین برای فرکانس‌های بیشتر از 250rad/s باید بهره بین -7.5dB ~ -6 باشد. از دیاگرم بود می‌بینیم که باید 80 درجه فاز و 60db به بهره در فرکانس 250rad/s اضافه کنیم. در منحنی فاز نمودار بود می‌بینیم که یک قطب نزدیک 60rad/s است. از یک کنترلر PI استفاده می‌کنیم تا یک صفر در 60rad/sec قرار دهیم تا آنجا که منحنی فاز را از همواری خارج کند.

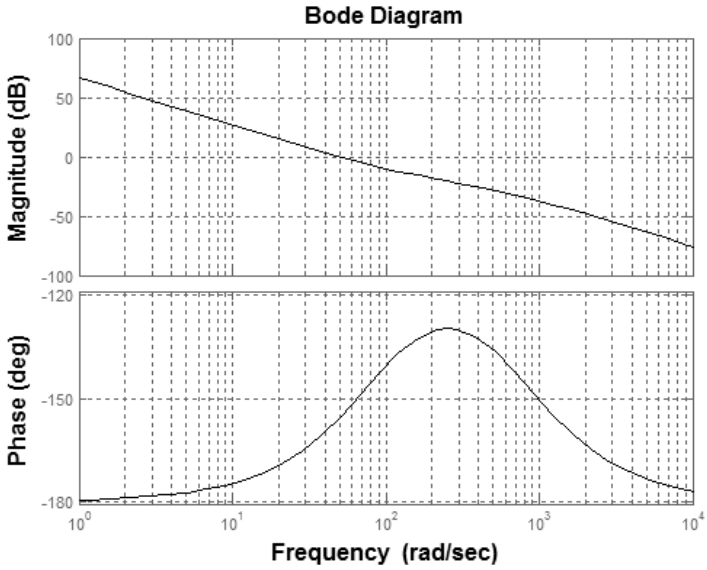
```
numpi=[1 60];
denpi=[1 0];
numpiol=conv(numpi,num);
denpiol=conv(denpi,den);
bode(numpiol,denpiol,w)
```

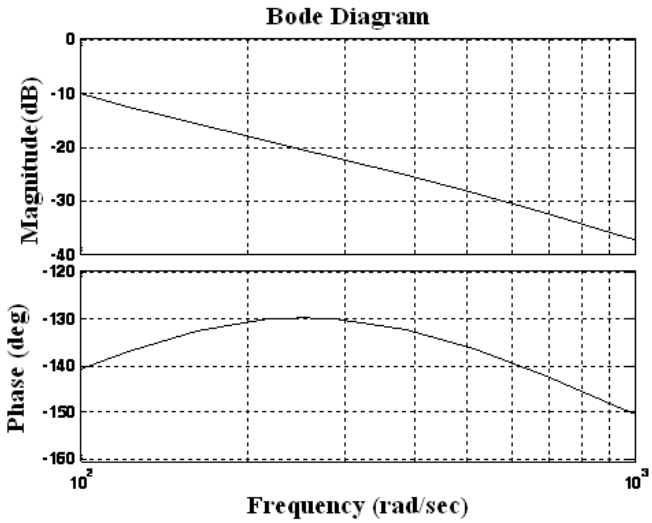


شکل (۳-۲۳) نمودار بود سیستم با اضافه شدن یک کنترلر PI

در نمودار بود می‌بینیم که به بیشتر از 50 درجه فاز در فرکانس 250rad/s نیاز داریم. پس از یک کنترلر پیش فاز Lead استفاده می‌کنیم تا دقیقاً 50 درجه فاز اضافه شود.

```
a=(1 - sin(PM*pi/180))/(1 + sin(PM*pi/180));
T=1/(wbw*sqrt(a));
numpil = conv([1 60],[T 1]);
denpil = conv([1 0],[a*T 1]);
numpilol = conv(numpil,num);
denpilol = conv(denpil,den);
bode(numpilol,denpilol)
%w = logspace(2,3,101);
%bode(numpilol,denpilol,w)
```

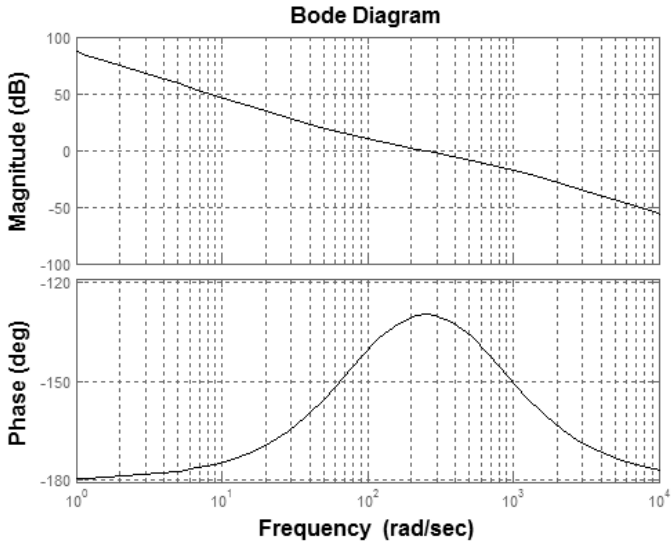




شکل (۳-۲۴) نمودار بود سیستم با اضافه شدن یک کنترلر Lead

از نمودار بود می‌بینیم که حد فاز در 250rad/s مطلوب است، اما مقدار بهره خیلی کوچک بوده و نزدیک 20dB است. باید W_g در فرکانس 250rad/s اتفاق بیفتد، بنابراین برای اضافه کردن دامنه، بهره 10 را در آن ضرب می‌کنیم.

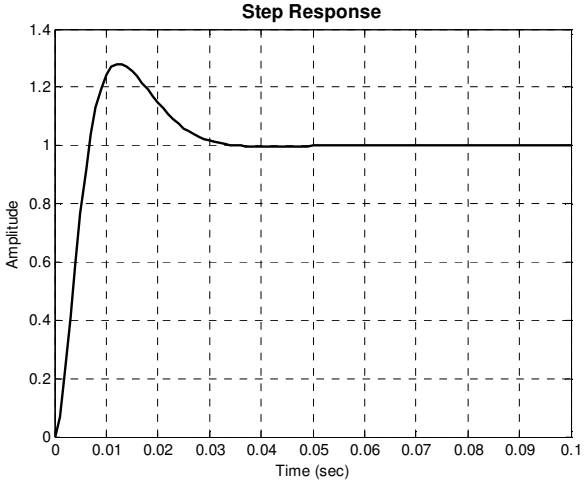
```
kpid = 10;
bode(kpid*numpilol,denpilol)
%w = logspace(2,3,101);
%bode(numpilol,denpilol,w)
```



شکل (۳-۲۵) نمودار بود سیستم با اضافه شدن یک کنترلر Lead

حال پاسخ پله را بررسی می‌کنیم:

```
[numpilcl,denpilcl] = cloop(kpid*numpilol,denpilol,-1);
t = 0:0.001:0.1;
step(numpilcl,denpilcl)
```

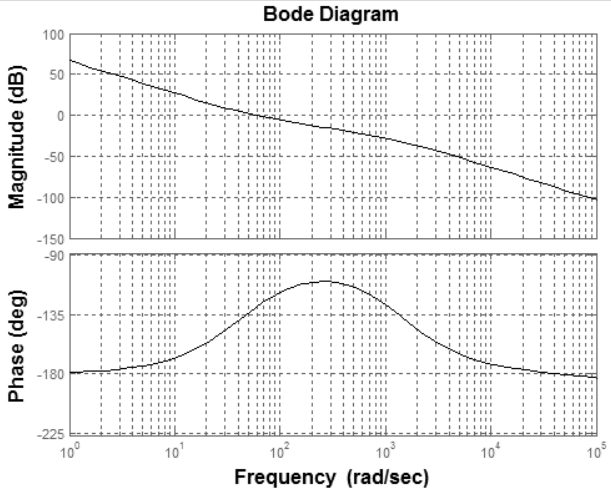


شکل (۳-۲۶) پاسخ پله سیستم حلقه بسته

مقدار جهش خیلی زیاد است ولی مقدار زمان نشست بهتر از حد انتظار می‌باشد. پس باید حد فاز را تا مقدار 70 درجه افزایش دهیم.

```
PM=70; a=(1 - sin(PM*pi/180))/(1 + sin(PM*pi/180));
T=1/(wbw*sqrt(a));
```

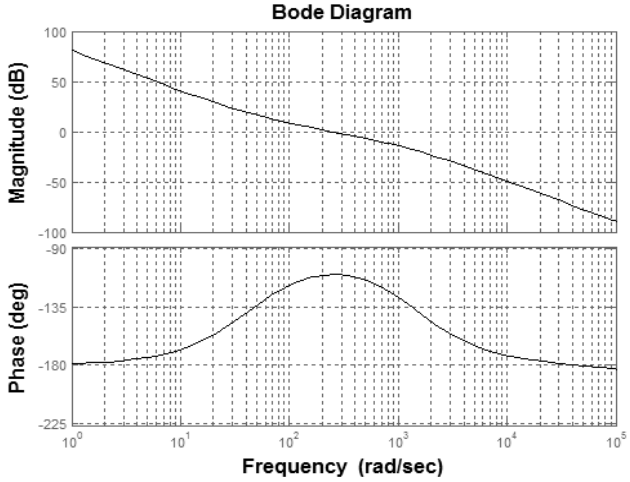
```
numpil=conv([1 60],[T 1]);
denpil=conv([1 0],[a*T 1]);
numpilol=conv(numpil,num);
denpilol=conv(denpil,den);
bode(numpilol,denpilol)
```



شکل (۳-۲۷) نمودار بود سیستم با اضافه شدن یک کنترلر Lead با حد فاز ۷۰ درجه

از روی شکل مشخص است که حد فاز در حدود 250rad/s مناسب است ولی مقدار بهره 14db خیلی کوچک است. بنابراین مقدار بهره را افزایش داده و آن را در 5 ضرب می‌کنیم.

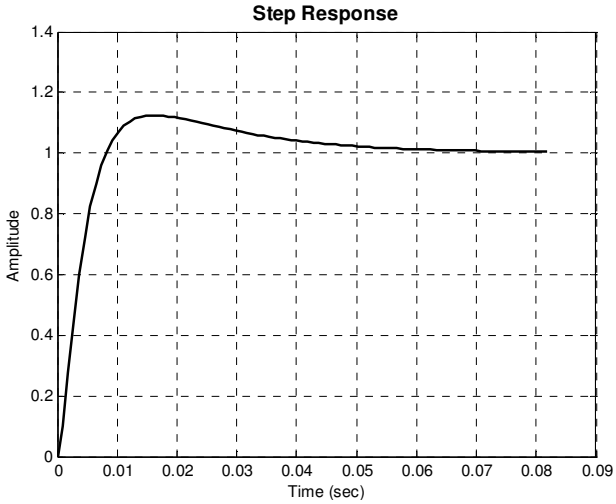
```
kpid = 5;
bode(kpid*numpilol,denpilol)
w = logspace(2,3,101);
bode(kpid*numpilol,denpilol,w)
```



شکل (۳-۲۸) نمودار بود سیستم با اضافه شدن یک کنترلر Lead

حال پاسخ پله را یک بار دیگر بررسی می‌کنیم:

```
[numpilcl,denpilcl] = cloop(kpid*numpilol,denpilol,-1);
t = 0:0.001:0.1;
step(numpilcl,denpilcl)
```



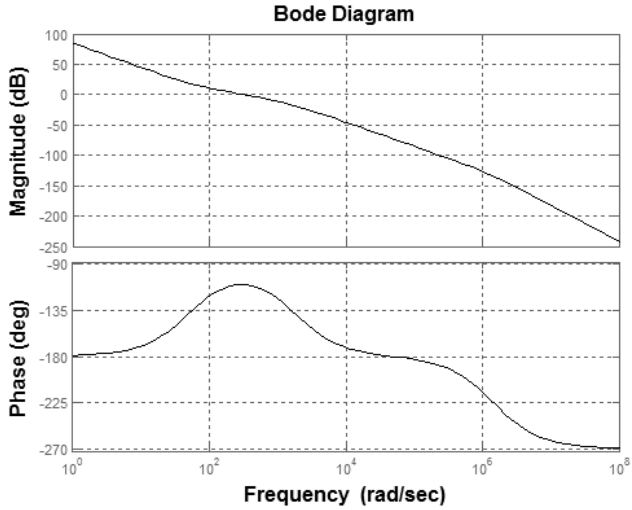
شکل (۳-۲۹) پاسخ پله سیستم حلقه بسته

از روی پاسخ پله مشاهده می‌شود که جهش مناسب است اما زمان نشست طولانی است. پس تا حدودی پهنای باند فرکانسی باید بیشتر شود، از اینرو مقدار بهره را برابر 8 قرار می‌دهیم تا مقدار دامنه بیشتر شود.

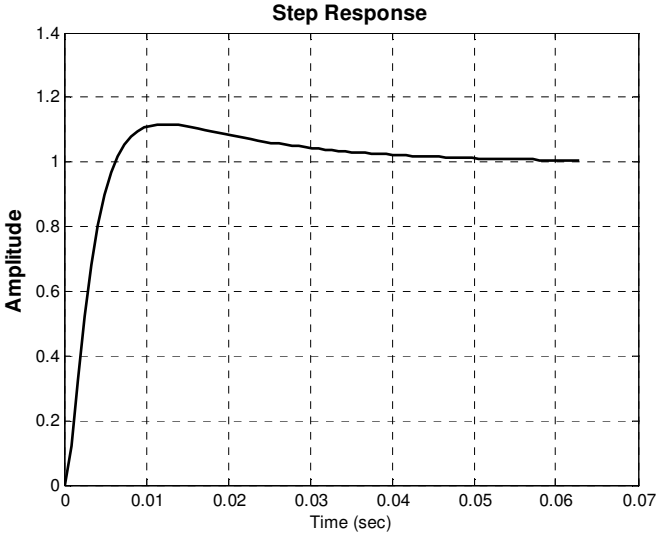
```
wbw=300;
a=(1-sin(PM*pi/180))/(1+sin(PM*pi/180));
T=1/(wbw*sqrt(a));
numpil=conv([1 60],[T 1]);
denpil=conv([1 0],[a*T 1]);
numpilol=conv(numpil,num);
denpilol=conv(denpil,den);
bode(numpilol,denpilol)
%w = logspace(2,3,101);
%bode(numpilol,denpilol,w)
```

پاسخ پله را یک بار دیگر چک می‌کنیم.

```
kpid=8;
bode(kpid*numpilol,denpilol);
%w = logspace(2,3,101);
%bode(kpid*numpilol,denpilol,w)
[numpilcl,denpilcl]=cloop(kpid*numpilol,denpilol,-1);
t=0:0.001:0.1;
step(numpilcl,denpilcl)
```



شکل (۳-۳۰) نمودار بود سیستم با اضافه شدن یک کنترلر Lead با پهنای باند فرکانس ۳۰۰



شکل (۳-۳۱) پاسخ پله سیستم حلقه بسته

همان‌طورکه می‌بینیم روند طراحی به صورتی بود که نیاز به چند بار سعی و خطا داشت. سرانجام تمام معیارهای طراحی برآورده شد.

۳-۵ طراحی فضای حالت

معادلات دینامیکی به فرم فضای حالت به صورت زیر نوشته می‌شود:

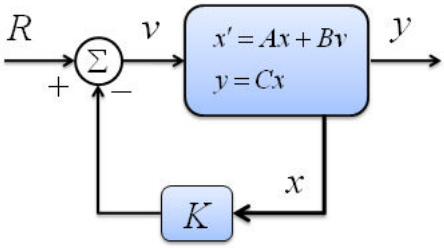
$$\begin{cases}\dot{x}(t)=Ax(t)+Bu(t)\\ y(t)=Cx(t)+Du(t)\end{cases}\Rightarrow\begin{cases}\dfrac{d}{dt}\begin{bmatrix}\theta\\ \dot{\theta}\\ i\end{bmatrix}=\begin{bmatrix}0 & 1 & 0\\ 0 & \frac{-C_m}{J} & \frac{K}{J}\\ 0 & \frac{-K}{L} & \frac{-R}{L}\end{bmatrix}\begin{bmatrix}\theta\\ \dot{\theta}\\ i\end{bmatrix}+\begin{bmatrix}0\\ 0\\ \frac{1}{L}\end{bmatrix}V\\ \theta'=\begin{bmatrix}1 & 0 & 0\end{bmatrix}\begin{bmatrix}\theta\\ \dot{\theta}\\ i\end{bmatrix}\end{cases}$$

دستورات زیر را در یک m-file کپی کنید:

```
J=3.2284E-6;
Cm=3.5077E-6;
K=0.0274;R=4;L=2.75E-6;
A=[0 1 0
   0 -Cm/J K/J
   0 -K/L -R/L];
B=[0 ; 0 ; 1/L];
C=[1 0 0];
D=[0];
```

۳-۴-۳ طراحی کنترلر با فیدبک کلیه متغیرهای حالت

از آنجایی که تمام متغیرهای حالت را به راحتی می‌توان اندازه گیری کرد (جریان با آمپر متر، سرعت زاویه‌ای را با تاکومتر و موقعیت را با پتانسیومتر) بنابراین بدون نیاز به طراحی رویتگر می‌توان یک کنترلر با فیدبک تمام حالت‌ها را طراحی کنیم که شماتیک آن در شکل(۳-۳۲) آمده است.



شکل (۳-۳۲) کنترل به روش فیدبک کلیه متغیرهای حالت

$$\dot{X}=(A-BK)X+B.R$$
$$Y=CX$$

معادله مشخصه این سیستم را می‌توان از رابطه $|sI-(A-BK)|$ به‌دست آورد که s متغیر حوزه لاپلاس می‌باشد. از آنجایی که هر یک از ماتریس‌های A و $B.K$، 3×3 می‌باشند بنابراین سه قطب برای این سیستم می‌توان در نظر گرفت. با استفاده از روش فیدبک کلیه متغیرهای حالت می‌توانیم متغیرها را در هر مکانی که می‌خواهیم قرار دهیم. در ابتدا قطب‌ها را در مکان‌های $-200, -100-100i, -100+100i$ قرار می‌دهیم.

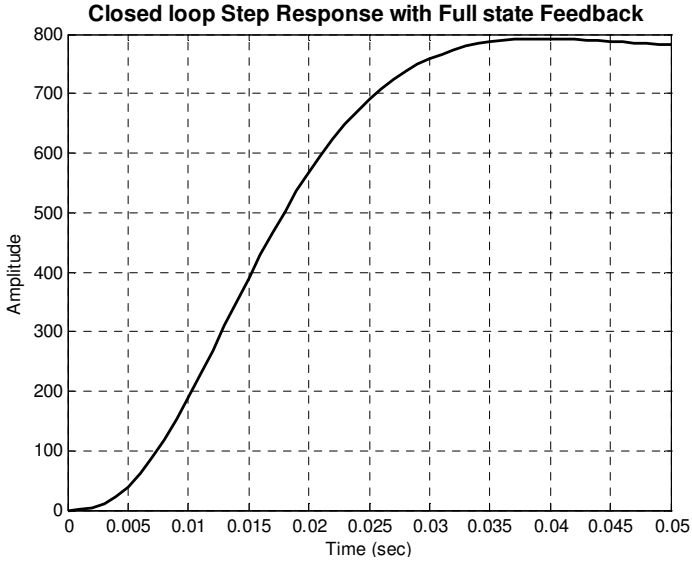
توجه کنید که این نقاط متناظر با $\zeta = 0.5, \omega_n = 100$ می‌باشند که این مقادیر نیز از زمان نشست $0.04s$ و خیز 16% به‌دست آمده اند.

$$M = \exp(\frac{-\pi\zeta}{\sqrt{1-\zeta^2}}) \Rightarrow 0.16 = \exp(\frac{-\pi\zeta}{\sqrt{1-\zeta^2}}) \Rightarrow \zeta = 0.5$$

$$\frac{4.6}{\zeta\omega} = 0.04 \Rightarrow \omega \simeq 100$$

برای پیدا کردن مقدار k دستورات زیر را در ادامه m-file اضافه کنید و خروجی آن‌را مشاهده کنید.

```
p1=-100+100i;
p2=-100-100i;
p3=-200;
Kc=place(A,B,[p1,p2,p3]);
t=0:0.001:.05;
step(A-B*Kc,B,C,D,1,t)
```



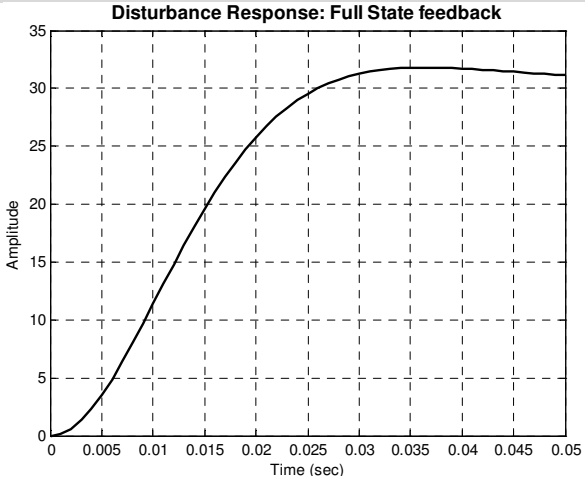
شکل (۳-۳۳) پاسخ پله سیستم با فیدبک کلیه متغیرهای حالت

۳-۴-۴پاسخ به اغتشاشات

به منظور رسیدن به پاسخ سیستم در برابر اغتشاشات باید ورودی مناسبی را برای سیستم فراهم کنیم. به طور فیزیکی اغتشاشات به صورت یک گشتاور در ورودی، روی سیستم تاثیر می‌گذارند. این گشتاور به صورت یک ترم اضافی به معادلات حالت اضافه می‌شود. با تصحیح ماتریس B سیستم حلقه بسته می‌توانیم پاسخ سیستم را در برابر اغتشاشات به‌دست آوریم.

```
Bn=[0; 1/J;0];
sys_D=ss(A-B*Kc,Bn,C,D);
figure(7)
step(sys_D,1,t);
```

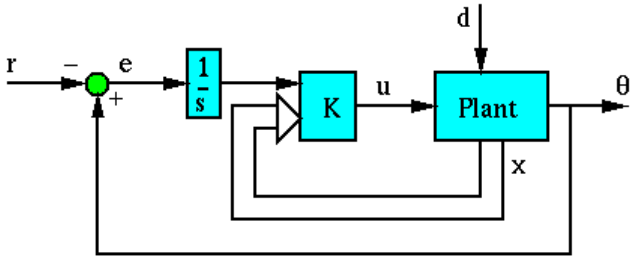
```
title('Disturbance Response: Full State feedback')
```



شکل (۳-۳۴) پاسخ سیستم حلقه بسته به اغتشاشات پله

۳-۴-۵ اضافه کردن عملگر انتگرالی

می‌دانیم که با اضافه کردن یک انتگرال‌گیر به صورت سری با سیستم تحت کنترل می‌توانیم خطای حالت ماندگار را کاهش دهیم پس با گذاشتن یک انتگرال‌گیر خطای حالت ماندگار ناشی از اغتشاشات را نیز حذف می‌کنیم. شکل زیر نحوه افزودن انتگرال‌گیر را نمایش می‌دهد. برای نوشتن معادلات جدید باید متغیر حالت انتگرال‌گیر را نیز در نظر گرفته و به معادلات اضافه کنیم.



شکل (۳-۳۵) شماتیک کنترل به روش فیدبک کلیه متغیرهای حالت با افزودن انتگرال‌گیر

$$\begin{cases} \dfrac{d}{dt}\begin{bmatrix} \int\theta \\ \theta \\ \dot{\theta} \\ i \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \dfrac{-C_m}{J} & \dfrac{K}{J} \\ 0 & 0 & \dfrac{-K}{L} & \dfrac{-R}{L} \end{bmatrix}\begin{bmatrix} \int\theta \\ \theta \\ \dot{\theta} \\ i \end{bmatrix} + \begin{bmatrix} -1 \\ 0 \\ 0 \\ 0 \end{bmatrix} r \\ y = \begin{bmatrix} 0 & 1 & 0 & 0 \end{bmatrix}\begin{bmatrix} \int\theta \\ \theta \\ \dot{\theta} \\ i \end{bmatrix} \end{cases}$$

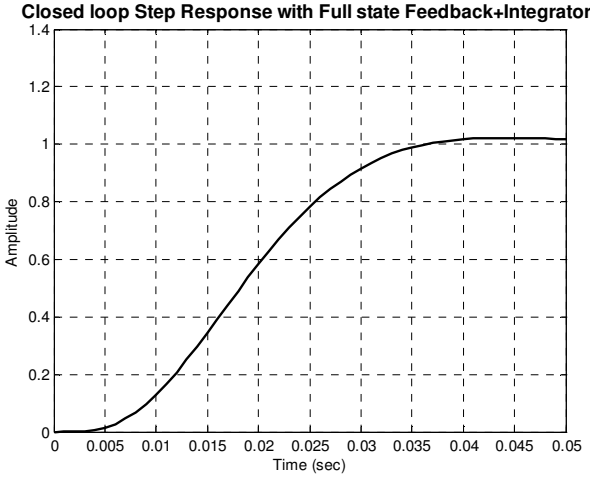
این معادله دینامیک سیستم قبل از بسته شدن حلقه کنترلی را نشان می‌دهد. در معادله بالا ماتریس‌های معادله حالت جدید را به ترتیب با A_a, B_a, C_a, D_a و متغیرهای حالت جدید را با X_a نمایش می‌دهیم. توجه کنید که ورودی r روی هیچکدام از متغیرها و خروجی سیستم، به‌جز متغیر انتگرال گیر تاثیر نمی‌گذارد. هیچ راهی جز استفاده از K برای بستن حلقه کنترلی و ارتباط ورودی r و u وجود ندارد.

برای پیدا کردن معادله سیستم حلقه بسته باید بفهمیم u چگونه روی سیستم تاثیر می‌گذارد. ماتریس B_{au} زمانی که به u به عنوان ورودی سیستم نگاه می‌کنیم جایگزین B_a می‌شود. این ماتریس همان ماتریس B است که یک سطر صفر به سطرهای آن اضافه شده است. ورودی $u=-KX_a$ به عنوان ورودی سیستم در حلقه بسته کنترلی و r به عنوان ورودی حلقه کنترلی است. معادله سیستم حلقه بسته هم به B_{au} و هم به B_a وابسته است. پس معادله سیستم حلقه بسته به صورت زیر خواهد بود:

$$\dot{X}_a = (A_a - B_{au}K)X_a + B_a.r \qquad (۸-۳)$$
$$Y = C_a X_a$$

یکی از قطب‌های سیستم حلقه بسته را در 300- قرار می‌دهیم. این قطب از بقیه قطب‌ها سریعتر می‌باشد. دستورات زیر را به ادامه m-file اضافه کنید:

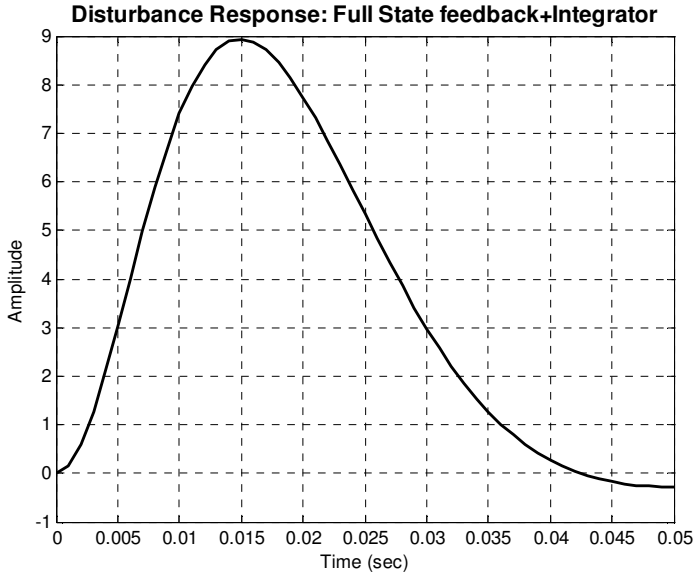
```
Aa=[0 1 0 0;0 0 1 0 ;0 0 -Cm/J K/J;0 0 -K/L -R/L];
Ba=[ -1 ; 0 ; 0 ; 0];
Bau=[0 ; 0 ; 0 ; 1/L ];
Ca=[0 1 0 0];
Da=[0];
p4=-300; Kc=place(Aa,Bau,[p1,p2,p3,p4]);
t=0:0.001:.05;
step(Aa-Bau*Kc,Ba,Ca,Da,1,t)
```



شکل (۳-۳۶) پاسخ سیستم به ورودی پله با روش کنترل به روش فیدبک کلیه متغیرهای حالت با اضافه کردن انتگرال گیر

حال پاسخ سیستم را در برابر اغتشاشات مشاهده کنید.

```
step(Aa-Bau*Kc,[0 ; 0 ; 1/J ; 0] ,Ca,Da,1,t)
```



شکل (۳-۳۷) پاسخ سیستم به اغتشاش پله با کنترل به روش فیدبک کلیه متغیرهای حالت با اضافه کردن انتگرال گیر

۳-۶ طراحی کنترلر دیجیتالی

در این قسمت، هدف طراحی کنترلر کننده دیجیتالی موقعیت موتور DC بر اساس مکان هندسی ریشه‌ها بوده و می‌خواهیم خطای حالت ماندگار سیستم به ورودی پله و اغتشاشات برابر صفر باشد. از طرفی موتور به سرعت به موقعیت نهایی برسد، یعنی زمان رسیدن به موقعیت ماندگار باید کمتر از 40ms بوده و مقدار جهش باید کمتر از 16% باشد.

اولین قدم در طراحی سیستم گسسته این است که تابع تبدیل سیستم پیوسته را به سیستم گسسته تبدیل کنیم. برای انجام این تبدیل از دستور c2dm استفاده می‌کنیم. در این طراحی زمان نمونه‌برداری را برابر 0.001 ثانیه در نظر می‌گیریم. پس یک m-file جدید باز کنید و دستورات زیر را وارد کنید:

```
R=4; K=0.0274;L=2.75E-6;J=3.2284E-6;Cm=3.5077E-6;
num = K;
den = [(J*L) (J*R)+(L*Cm) (R*Cm)+(K^2) 0];
T = 0.001;
[numd,dend] = c2dm(num,den,T,'zoh')
```

خروجی این دستورات به صورت زیر می‌باشد:

```
numd =
    0   0.0010   0.0010   0.0000
```

```
dend =
   1.0000   -1.9425    0.9425    0.0000
```

با توجه به نتایج بالا صورت و مخرج تابع تبدیل دارای یک ریشه اضافی در $z = 0$ می‌باشند. پس باید از دست این صفر خلاص شویم . برای حذف این صفر و قطب دستورات زیر را به ادامه m-file اضافه کنید:

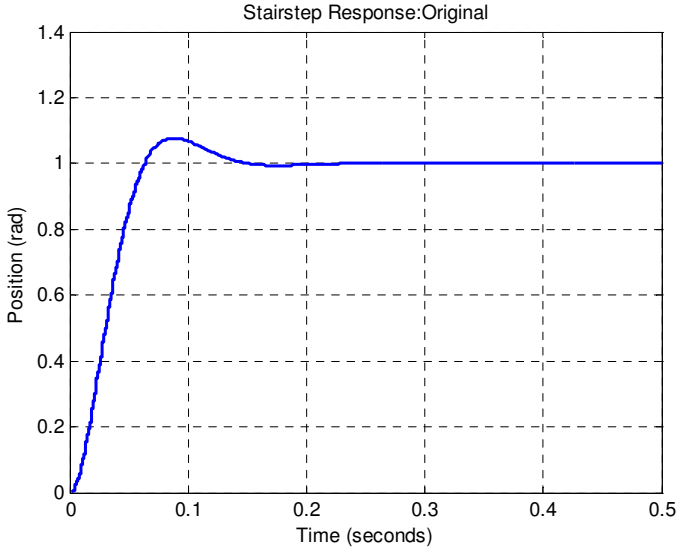
```
numd = numd(2:3)
dend = dend(1:3);
```

بنابراین تابع تبدیل گسسته با خروجی موقعیت و ورودی ولتاژ به صورت زیر به‌دست می‌آید:

$$\frac{\theta(z)}{V(z)} = \frac{0.001Z + 0.001}{Z^2 - 1.9425Z + 0.942}$$

می‌خواهیم ببینیم پاسخ سیستم حلقه بسته بدون کنترلر به چه صورت خواهد بود. پس اول باید تابع تبدیل سیستم حلقه بسته را با دستور Cloop به‌دست آوریم. سپس با استفاده از دستور dstep و stairs پاسخ سیستم حلقه بسته به ورودی پله را به‌دست می‌آوریم. پس دستورات زیر را به ادامه m-file اضافه کنید:

```
[numd_cl,dend_cl] = cloop(numd,dend);
[x1] = dstep(numd_cl,dend_cl,501);
t=0:0.001:0.5;
stairs(t,x1)
xlabel('Time (seconds)')
ylabel('Position (rad)')
title('Stairstep Response:Original')
```

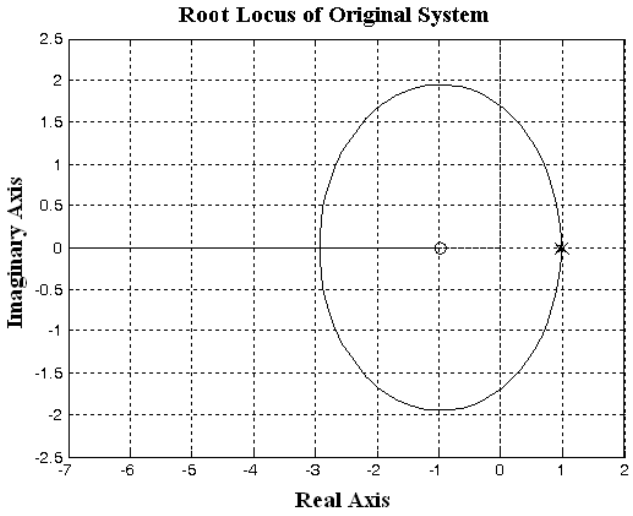


شکل (۳-۳۸) پاسخ پله سیستم حلقه بسته بدون کنترلر

۳-۶-۱ طراحی مکان هندسی ریشه‌ها

هدف اصلی در طراحی براساس مکان هندسی ریشه‌ها، تعیین پاسخ حلقه بسته از روی نمودار حلقه باز است. پس مکان هندسی ریشه‌ها را با اضافه کردن دستورات زیر بررسی می‌کنیم.

```
rlocus(numd,dend)
title('Root Locus of Original System')
zgrid(0,0)
```



شکل (۳-۳۹) مکان هندسی ریشه‌ها بدون کنترلر

برای رسیدن به خطای حالت ماندگار صفر نیاز داریم که یک کنترلر انتگرالی به سیستم اضافه کنیم. یاد آوری می‌کنیم که کنترلر انتگرالی در سیستم‌های پیوسته برابر $1/s$ است ولی با نگاشت از حوزه s به حوزه z، کنترلر انتگرالی در حوزه z به صورت $s=\frac{1}{z-1}$ خواهد بود. در مکان هندسی ریشه‌ها باید یک قطب در 1 اضافه شود. بعد از اضافه کردن قطب در 1 می‌بینیم که مکان هندسی دارای سه قطب در نزدیکی 1 است. بنابراین مکان هندسی به سمت راست کشیده شده و پاسخ سیستم حلقه بسته ناپایدارتر می‌شود. پس باید یک صفر در نزدیکی 1 در داخل دایره واحد اضافه کنیم تا اثر قطب را خنثی کند. پس صفر را در z=0.95 در نظر می‌گیریم. پس دستورات زیر را به ادامه M-file اضافه کنید:

```
numi = [1 -0.95];
deni = [1 -1];
numad = conv(numd,numi);
denad = conv(dend,deni);
```

حال از روی جهش و زمان نشست دلخواه مقدار فرکانس طبیعی و نسبت میرایی را به‌دست می‌آوریم.

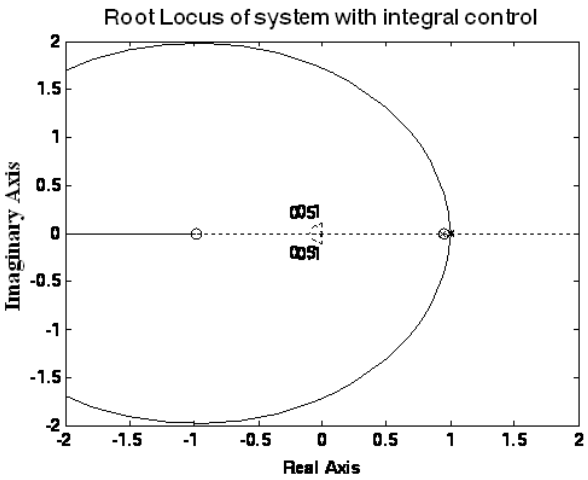
$$\zeta \geq \sqrt{\frac{(\ln M_p/\pi)^2}{1+(\ln M_p/\pi)^2}}$$
$$\zeta\omega_n \geq \frac{4.6}{T_s} \qquad (۹-۳)$$

مقدار نسبت میرایی برابر 0.5 و فرکانس طبیعی برابر 100 rad/sec به‌دست می‌آید. ولی برای استفاده از دستور zgrid و به‌دست آوردن ناحیه مطلوب، نیاز است که فرکانس طبیعی را تبدیل واحد کنیم. پس

$$w_n = 100 \times T = 0.1 \frac{rad}{sample}$$

```
rlocus(numad,denad);
zgrid(0.5,0.1)
title('Root Locus of system with integral control')
```

و نمودار زیر حاصل می‌شود:

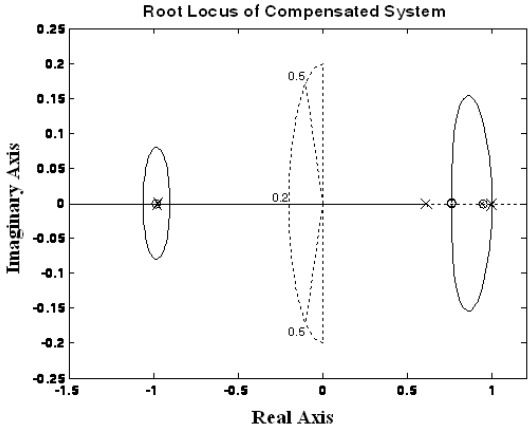


شکل (۳-۴۰) مکان هندسی ریشه‌ها با کنترلر انتگرالی

از روی نمودار مشخص است که برای تمامی بهره‌ها سیستم ناپایدار است، چون مکان هندسی خارج دایره واحد قرار گرفته است. بنابراین باید مکان هندسی را بیشتر بکشیم. پس ابتدا باید صفری را که تقریباً در 0.98- قرار دارد حذف کنیم. از آنجایی که این صفر باعث اضافه شدن جهش به ورودی پله می‌شود بنابراین باید بیشتر از یک قطب و دو صفر نزدیک قطب‌های مطلوب اضافه کنیم. بعد از چند بار سعی و خطا چند قطب باید در 0.61 و دو صفر در 0.76 اضافه شود. دستورات زیر را به ادامه M-file اضافه کنید:

```
numc = conv([1 -0.76],[1 -0.76]);
denc = conv([1 0.98],[1 -0.61]);
numoc = conv(numad,numc);
denoc = conv(denad,denc);
rlocus(numoc,denoc);
zgrid(0.5,0.1)
title('Root Locus of Compensated System')
```

denoc یک قطب در 0.61 به جای 0.98- دارد.



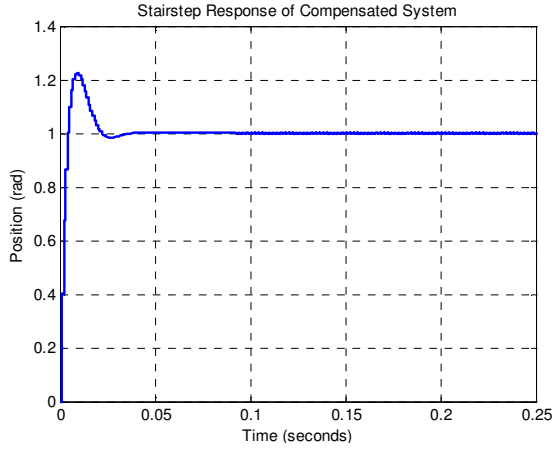
شکل (۳-۴۱) مکان هندسی ریشه‌های سیستم جبران شده

از روی نمودار بالا مشخص است که مکان هندسی ریشه‌ها به ناحیه مطلوب کشیده شده است. اجازه بدهید بهره K را با استفاده از دستور rlocfind پیدا کنیم و پاسخ پله را به‌دست آوریم. پس فرامین زیر را به ادامه m-file اضافه کنید:

```
K = rlocfind(numoc,denoc)
[numd_cl,dend_cl] = cloop(K*numoc,denoc);
[x2] = dstep(numd_cl,dend_cl,251);
t=0:0.001:0.25;
stairs(t,x2)
xlabel('Time (seconds)')
ylabel('Position (rad)')
title('Stairstep Response of Compensated System')
```

با اجرای فرامین بالا از شما یک نقطه روی مکان هندسی ریشه‌ها پرسیده می‌شود. بهره مورد نظر برابر 385 است و پاسخ پله سیستم جبران شده به صورت زیر خواهد بود:

selected_point = 0.4994 - 0.0008i

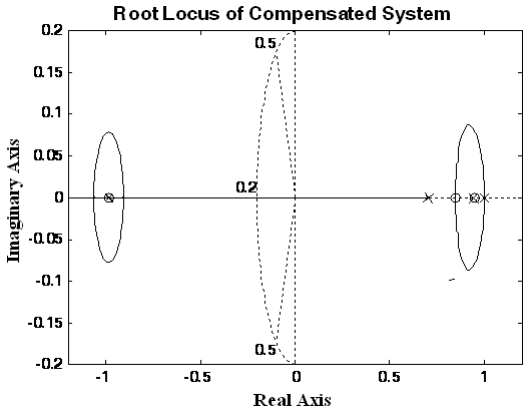


شکل (۳-۴۲) پاسخ پله سیستم جبران شده

از روی پاسخ حلقه بسته مشخص است که زمان نشست حدود 0.05 ثانیه است که رضایت بخش می‌باشد. ولی مقدار جهش حدود 22% است که مقدار آن به دلیل وجود صفرها زیاد است. واضح است که اگر ما مقدار بهره را بزرگتر انتخاب کنیم، به هدفمان می‌رسیم ولی با این کار مساله تا حدودی غیر واقعی شده و نیاز به یک عملگر بسیار بزرگ داریم. بنابراین مجبوریم که هم قطب و هم هر دو صفر را اندکی بیشتر به سمت راست حرکت دهیم تا مکان هندسی را کمی بیشتر بکشیم. قطب جدید در 0.7 خواهد بود و دو صفر باید در 0.85 باشد. به M-file بر گردید و numc , denc را تغییر دهید:

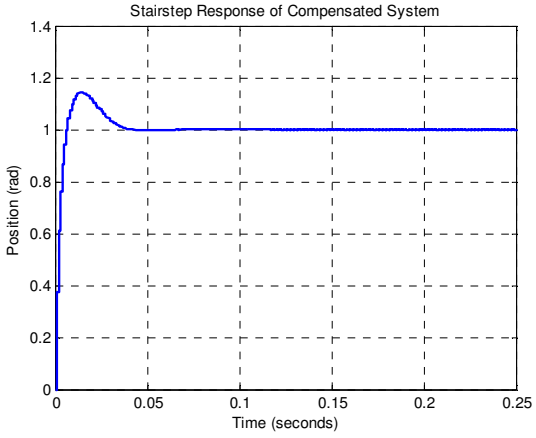
```
numc = conv([1 -0.85],[1 -0.85]);
denc = conv([1 0.98],[1 -0.7]);
```

بنابراین شما باید چنین نموداری را ببینید:



شکل (۳-۴۳) مکان هندسی ریشه‌ها ی سیستم جبران شده و اصلاح شده

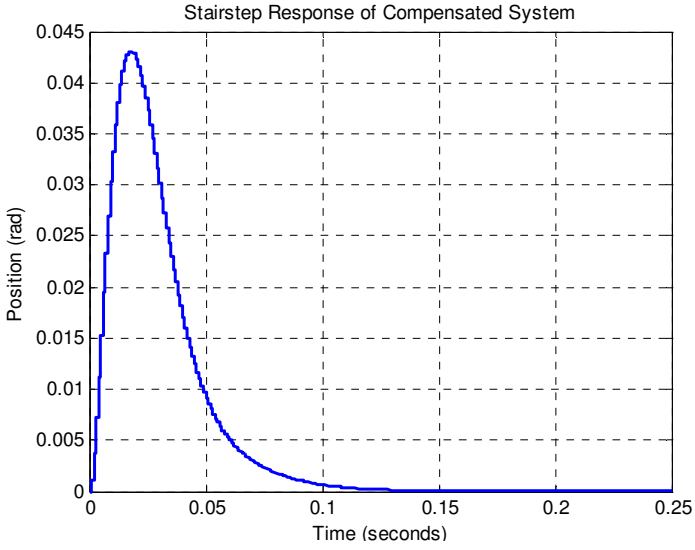
حال باید روی یک نقطه روی نمودار کلیک نمائید تا بهره جدیدی را انتخاب کنید. بهره انتخاب شده باید حدود 450 باشد. پس پاسخ سیستم حلقه بسته جبران شده به صورت زیر خواهد بود:



شکل (۳-۴۴) پاسخ پله سیستم جبران شده و اصلاح شده

همان طور که از پاسخ پله بر می‌آید زمان نشست، در حدود 0.04 ثانیه و مقدار جهش در حدود 10% است. پس اکنون به پاسخ سیستم حلقه بسته به اغتشاشات خارجی با حذف بهره ی انتخاب شده، تابع تبدیل انتگرالی، و تابع تبدیل کنترلی از تابع تبدیل حلقه بسته نگاه کنید. بنابراین کدهای زیر را به ادامه M-file اضافه کنید:

```
numcld = conv(numd_cl,conv(denc,deni));
dencld = conv(dend_cl,conv(K*numc,numi));
[x4] = dstep(numcld,dencld,251);
t=0:0.001:.25;
stairs(t,x4)
xlabel('Time (seconds)')
ylabel('Position (rad)')
title('Stairstep Response of Compensated System')
```



شکل (۳-۴۵) پاسخ سیستم جبران شده به اغتشاشات خارجی

همانطور که مشخص است پاسخ به اغتشاشات بسیار کوچک بوده (4.3% اغتشاشات وارده) و به تدریج در زمان 0.2 ثانیه به مقدار صفر می‌رسد.

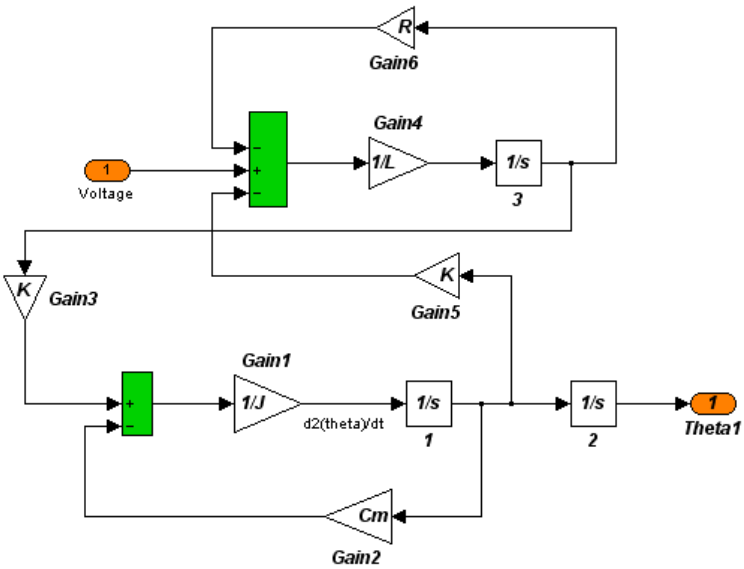
۳-۷ طراحی کنترلر دیجیتال در محیط سیمولینک

اگر قوانین نیوتن و قوانین کیرشهف را برای مدل مورد مطالعه بنویسیم به روابط زیر خواهیم رسید:

$$(1)\ T = J\ddot{\theta} + C_m\dot{\theta} \Rightarrow Ki = J\ddot{\theta} + C_m\dot{\theta} \Rightarrow \frac{d^2\theta}{dt^2} = \frac{K}{J}i - \frac{C_m}{J}\frac{d\theta}{dt}$$

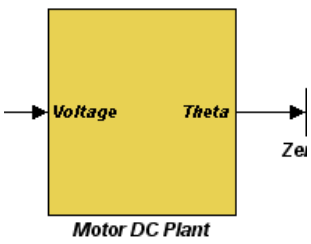
$$(2)\ e_a - e_b = Ri + L\frac{di}{dt} \Rightarrow V - K\dot{\theta} = Ri + L\frac{di}{dt} \Rightarrow \frac{di}{dt} = \frac{V}{L} - \frac{K}{L}\frac{d\theta}{dt} - \frac{R}{L}i$$

در این قسمت باید مدل ریاضی به‌دست آمده را در محیط سیمولینک به یکسری بلوک نمودار تبدیل کنیم. در ابتدا مدل دینامیکی سیستم را در این محیط شبیه سازی کرده، و پاسخ سیستم حلقه باز را بررسی می‌کنیم. پس مطابق شکل (۴-۴۵) مدل دینامیکی سیستم را بسازید.

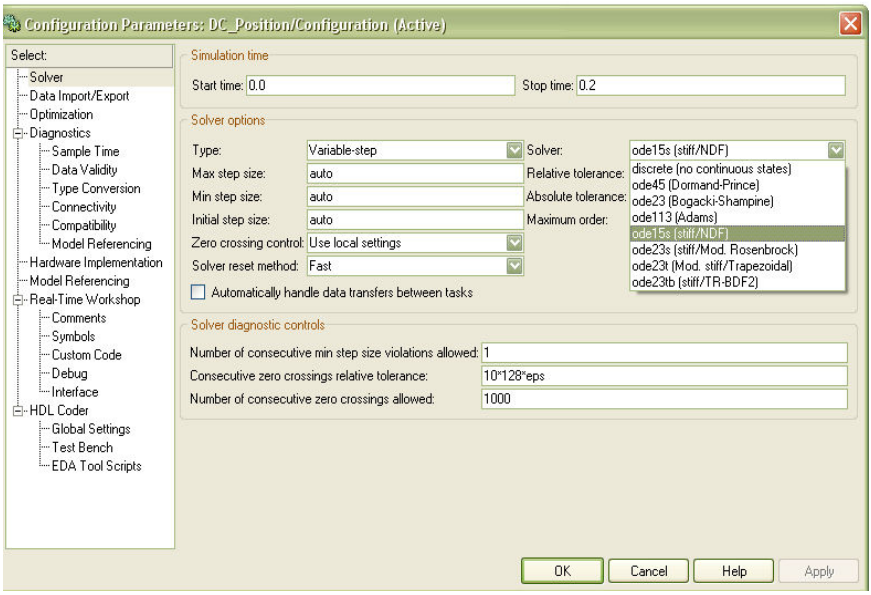


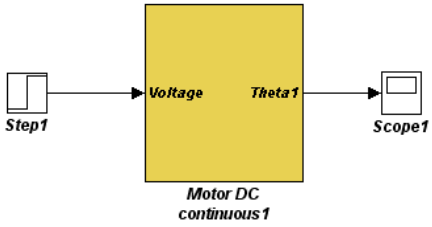
شکل (۳-۴۶) مدل دینامیکی سیستم

سپس تمامی المان‌های کشیده‌شده را انتخاب کرده و روی صفحه کلیک راست کنید. سپس Create Subsystem را انتخاب کنید تا تمامی المان‌ها در داخل یک بلوک قرار بگیرند. از این به بعد این بلوک نشان دهنده سیستم است.



بعد از شبیه‌سازی مدل سیستم، باید پاسخ سیستم حلقه باز را بررسی کنیم. بنابراین در ورودی سیستم از ورودی پله و در خروجی یک Scope قرار داده، پارامترها را به صورت زیر تنظیم می‌کنیم. توجه کنید که بهتر است نوع حل را "ode15s (stiff/NDF)" انتخاب کنیم. زیرا، زمان نمونه برداری کوچک است و این حل کارآمدتر می‌باشد.

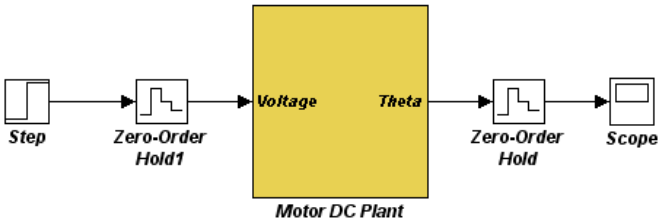


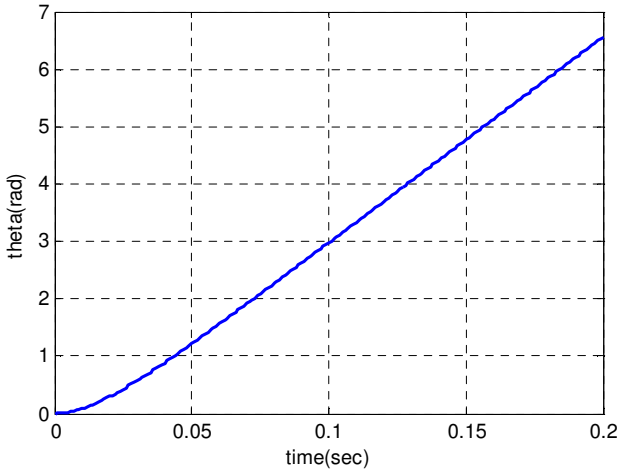


خروجی زیر حاصل می‌شود. سپس این فایل را با نام Dc_Position.mdl ذخیره کنید و در محیط نرم‌افزار دستورات زیر را تایپ نمائید.

```
J=3.2284E-6; Cm=3.5077E-6;
K=0.0274;R=4; L=2.75E-6;
```

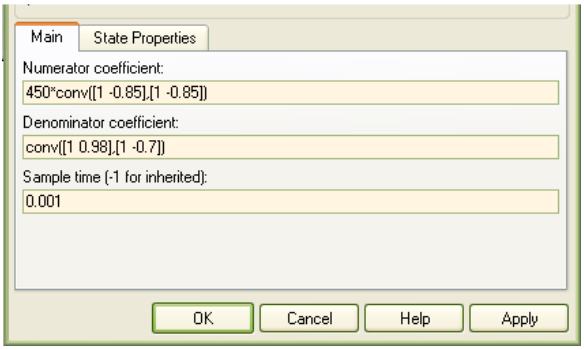
برای طراحی کنترلر دیجیتالی برای این سیستم پیوسته باید بلوکهای Zero order-Hold را در ورودی و خروجی سیستم اضافه کنیم، به‌طوری‌که از یک A/D در خروجی و یک D/A در ورودی سیستم استفاده کنیم. روی هر یک از این المان‌های ZOH کلیک می‌کنیم و مقدار زمان نمونه برداری را برابر 0.001 قرار می‌دهیم. روی ورودی پله نیز کلیک کرده و مقدار زمان پله را برابر 0 و زمان نمونه برداری آن را نیز برابر 0.001 قرار می‌دهیم.



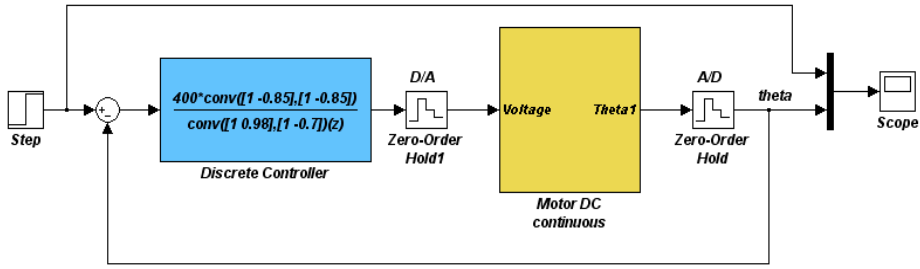


شکل (۳-۴۷) پاسخ پله سیستم حلقه باز

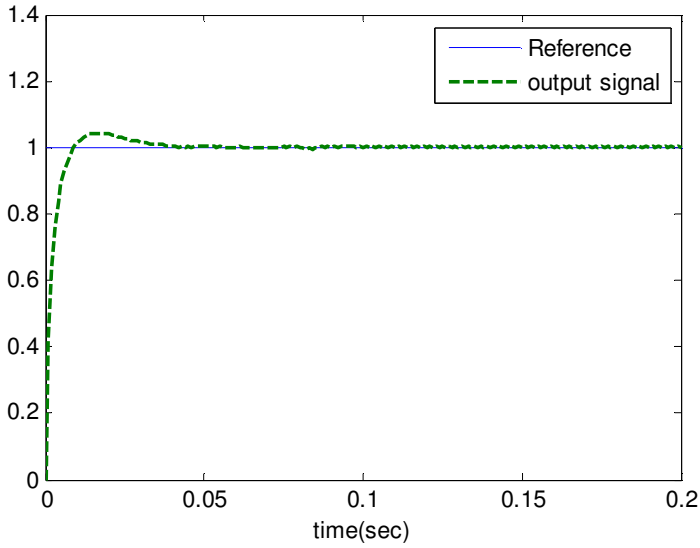
در مرحله بعد برای سیستم یک کنترلر دیجیتالی طراحی می‌کنیم. در بخش طراحی کنترلر دیجیتالی تابع تبدیل کنترلر را به صورت $\frac{450\times(z-0.85)(z-0.85)}{(z+0.98)(z-0.7)}$ به‌دست آوردیم. بنابراین از کتابخانه سیمولینک از قسمت Discrete بلوک Discrete Transfer Fcn را انتخاب کرده، آن‌را به محیط شبیه‌سازی می‌کشیم و روی آن دو بار کلیک کرده و مقادیر زیر را وارد می‌کنیم.



سپس مطابق شکل مدار سیستم کنترل حلقه بسته را طراحی می‌کنیم.



خروجی سیستم تحت کنترل به صورت زیر به‌دست می‌آید:



شکل (۳-۴۸) پاسخ پله سیستم حلقه بسته با کنترلر دیجیتالی

پروژه چهارم

کنترل سیستم تعلیق خودرو

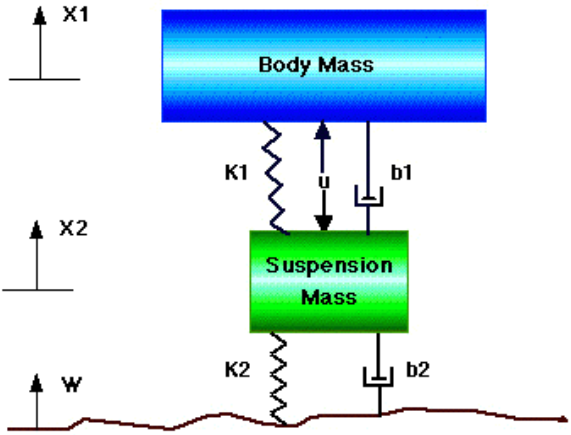
اهداف این پروژه

- **مدل سازی**
- **طراحی کنترلر PID**
- **رسم مکان هندسی ریشه ها (Root Locus)**
- **پاسخ فرکانسی**
- **طراحی فضای حالت**
- **طراحی کنترلر دیجیتالی**
- **طراحی کنترلر با فیدبک تمامی متغیر های حالت در محیط سیمولینک**

۴ –۱ مدل‌سازی

۴-۱-۱ مدل سازی سیستم براساس اصول فیزیکی و استخراج معادلات سیستم

مسأله طراحی سیستم تعلیق اتوماتیک برای یک خودرو یک مساله جالب کنترلی است. شکل (۴-۱) سیستم تعلیق فعال یک خودرو را نمایش می دهد. در این مدل که به مدل یک چهارم خودرو مشهور است تنها حرکت عمودی خودرو با جرم M با متغیر نسبی x_1 مشخص شده است. اکسل خودرو نیز دارای جرم m و جابه‌جایی عمودی x_2 می باشد. تایر خودرو با ضریب سختی K_2 و ضریب استهلاک b_2 وکمک فنر خودرو با ضریب سختی K_1 و ضریب استهلاک ویسکوز b_1 مدل شده اند. علاوه بر وجود فنر و کمک فنر، همانند سیستم های تعلیق خودرو یک عملگر هیدرولیکی یا نیوماتیکی نیروی u را به بدنه خودرو اعمال می کند تا از نوسانات خودرو جلوگیری کند. ناهمواری را با ورودی اغتشاشی W نمایش داده ایم.



شکل (۴-۱) مدل $\frac{1}{4}$ سیستم تعلیق خودرو

در این سیستم مقادیر پارامتر های فیزیکی را به صورت زیر در نظر می‌گیریم:

جرم بدنه خودرو: $M_1 = 2500\,kg$

جرم اکسل و سیستم تعلیق : $M_2 = 320\,kg$

ضریب سختی فنر سیستم تعلیق : $K_1 = 80000N/m$

ضریب سختی تایر خودرو: $K_2 = 500000N/m$

ضریب استهلاک ویسکوز سیستم تعلیق: $b_1 = 350N.s/m$

ضریب استهلاک تایر خودرو: $b_2 = 15020N.s/m$

نیروی کنترلی u نیرویی است که از طرف کنترلر وارد می شود و قصد داریم آن را طراحی کنیم.

نیروی فنر به صورت خطی است یعنی $F = K\Delta x$ ، و نیروی کمک فنر نیز به صورت استهلاک ویسکوز با رابطه $F = b.\Delta\dot{x}$ عمل می کند. معادلات حاکم بر سیستم، معادلات نیوتن می‌باشد که با توجه به شکل می‌توان معادلات دینامیکی را به صورت زیر نوشت:

$$\begin{cases} M_1\ddot{x}_1 = -b_1(\dot{x}_1 - \dot{x}_2) - k_1(x_1 - x_2) + u \\ M_2\ddot{x}_2 = b_1(\dot{x}_1 - \dot{x}_2) + k_1(x_1 - x_2) + b_2(\dot{w} - \dot{x}_2) + k_2(w - x_2) - u \end{cases} \quad (۱-۴)$$

۴-۱-۲ تابع تبدیل سیستم

فرض کنید که تمامی شرایط اولیه برابر صفر می باشد. با گرفتن لاپلاس از طرفین معادلات (۴-۱) تابع تبدیل سیستم به‌دست می آید. این سیستم دو ورودی دارد یکی U و دیگری W و خروجی سیستم برابر $x_1 - x_2$ می‌باشد. بنابراین با توجه به اینکه دو ورودی داریم درنتیجه، دو تابع تبدیل $G_1(s), G_2(s)$ خواهیم داشت که به صورت زیر به‌دست می‌آیند:

$$(M_1s^2+b_1s+k_1)X_1(s)-(b_1s+k_1)X_2(s)=U(s)$$
$$-(b_1s+k_1)X_1(s)+(M_2s^2+(b_1+b_2)s+(k_1+k_2))X_2(s)=(b_2s+k_2)W(s)-U(s)$$

(۲-۴)

$$\underbrace{\begin{bmatrix}(M_1s^2+b_1s+k_1) & -(b_1s+k_1)\\ -(b_1s+k_1) & (M_2s^2+(b_1+b_2)s+(k_1+k_2))\end{bmatrix}}_{A}\begin{bmatrix}X_1(s)\\X_2(s)\end{bmatrix}=\begin{bmatrix}U(s)\\(b_2s+k_2)W(s)-U(s)\end{bmatrix}$$

(۳-۴)

$$A=\begin{bmatrix}(M_1s^2+b_1s+k_1) & -(b_1s+k_1)\\ -(b_1s+k_1) & (M_2s^2+(b_1+b_2)s+(k_1+k_2))\end{bmatrix}$$
$$\Delta=\det\begin{bmatrix}(M_1s^2+b_1s+k_1) & -(b_1s+k_1)\\ -(b_1s+k_1) & (M_2s^2+(b_1+b_2)s+(k_1+k_2))\end{bmatrix}$$
$$\Delta=(M_1s^2+b_1s+k_1).(M_2s^2+(b_1+b_2)s+(k_1+k_2))-(b_1s+k_1).(b_1s+k_1)$$

(۴-۴)

حال باید معکوس ماتریس A را به‌دست آوریم:

$$\begin{bmatrix}X_1(s)\\X_2(s)\end{bmatrix}=\frac{1}{\Delta}\begin{bmatrix}(M_2s^2+(b_1+b_2)s+(k_1+k_2)) & (b_1s+k_1)\\ (b_1s+k_1 & (M_1s^2+b_1s+k_1)\end{bmatrix}\begin{bmatrix}U(s)\\(b_2s+k_2)W(s)-U(s)\end{bmatrix}$$

(۵-۴)

$$\begin{bmatrix}X_1(s)\\X_2(s)\end{bmatrix}=\frac{1}{\Delta}\begin{bmatrix}(M_2s^2+b_2s+k_2) & (b_1b_2s^2+(b_1k_2+b_2k_1)s+k_1k_2)\\ -M_1s^2 & \left(M_1b_2s^3+(M_1k_2+b_1b_2)s^2+(b_1k_2+b_2k_1)s+k_1k_2\right)\end{bmatrix}\begin{bmatrix}U(s)\\W(s)\end{bmatrix}$$

(۶-۴)

اگر تنها نیروی ورودی U(s) باشد و قرار دهیم W(s)=0 ، در نتیجه تابع تبدیل $G_1(s)$ به صورت زیر به‌دست می آید:

$$G_1(s)=\frac{X_1(s)-X_2(s)}{U(s)}=\frac{(M_1+M_2)s^2+b_2s+k_2}{\Delta}$$

(۷-۴)

اگر تنها نیروی ورودی W(s) باشد و قرار دهیمU(s)=0 ، در نتیجه تابع تبدیل $G_2(s)$ به صورت زیر به‌دست می آید:

$$G_2(s)=\frac{X_1(s)-X_2(s)}{W(s)}=\frac{-M_1b_2s^3-M_1k_2s^2}{\Delta}$$
$$\Delta=(M_1s^2+b_1s+k_1).(M_2s^2+(b_1+b_2)s+(k_1+k_2))-(b_1s+k_1).(b_1s+k_1)$$

(۸-۴)

۴-۱-۳ مدل فضای حالت سیستم

فرم فضای حالت سیستم به صورت زیر نوشته می شود:

$$\begin{bmatrix} \dot{x}_1 \\ \ddot{x}_1 \\ \dot{y}_1 \\ \ddot{y}_1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ \frac{-b_1 b_2}{M_1 M_2} & 0 & \left(\frac{b_1}{M_1}\left(\frac{b_1}{M_1}+\frac{b_1}{M_2}+\frac{b_2}{M_2}\right)-\frac{k_1}{M_1}\right) & \frac{-b_1}{M_1} \\ \frac{b_2}{M_2} & 0 & -\left(\frac{b_1}{M_1}+\frac{b_1}{M_2}+\frac{b_2}{M_2}\right) & 1 \\ \frac{k_2}{M_2} & 0 & -\left(\frac{k_1}{M_1}+\frac{k_1}{M_2}+\frac{k_2}{M_2}\right) & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ \dot{x}_1 \\ y_1 \\ \dot{y}_1 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ \frac{1}{M_1} & \frac{b_1 b_2}{M_1 M_2} \\ 0 & \frac{-b_2}{M_2} \\ (\frac{1}{M_1}+\frac{1}{M_2}) & \frac{-k_2}{M_2} \end{bmatrix} \begin{bmatrix} U \\ W \end{bmatrix} \qquad (۹-۴)$$

$$y = [\,0\;\;0\;\;1\;\;0\,] \begin{bmatrix} x_1 \\ \dot{x}_1 \\ y_1 \\ \dot{y}_1 \end{bmatrix} + [0\;\;0] \begin{bmatrix} U \\ W \end{bmatrix}$$

۴-۱-۴ قید های حاکم بر طراحی

یک سیستم تعلیق خوب باید هنگام عبور خودرو از دست اندازها شرایط بسیار راحتی را برای راننده فراهم کند. هنگام تست، خودرو در عبور از خیابان و دست اندازها نباید نوسانات بزرگی داشته باشد و سریعاً نوسانات را پخش و مهار کند. از آنجایی که اندازه گیری $X_1 - W$ (فاصله بدنه خودرو تا سطح جاده) بسیار مشکل است و از طرفی نیز می‌توانیم از تغییر شکل تایر ($X_2 - W$) چشم‌پوشی کنیم، بنابراین فاصله نسبی $y_1 = X_1 - X_2$ را به جای $X_1 - W$ به عنوان خروجی در نظر می گیریم. پس به خاطر داشته باشید که y_1 یک تخمین است.

- اغتشاشات جاده (W) با ورودی پله شبیه سازی می شود.
- ما می خواهیم با به‌کارگیری کنترل فیدبک خروجی سیستم $X_1 - X_2$ جهشی کمتر از 5% داشته باشد.
- زمان نشست کمتر از 5 ثانیه باشد.

یعنی زمانی که خودرو از یک دست انداز 10cm عبور می‌کند، دامنه نوسانات باید در محدوده $\pm 5\,mm$ بوده و بعد از 5 ثانیه به حرکت آرام خود ادامه دهد.

۴-۱-۵ پاسخ سیستم حلقه باز

حال بهتر است اجازه دهید پاسخ سیستم حلقه باز را به ورودی پله بررسی کنیم. برای این کار باید صورت و مخرج توابع تبدیل را به صورت برداری در محیط MATLAB وارد کنیم.

$$G_1(s) = \frac{nump}{denp} \quad : Actuated\ Force\ input$$

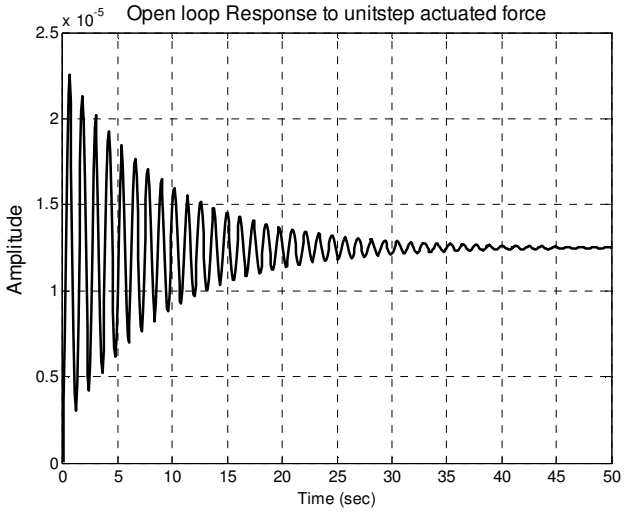
$$G_2(s) = \frac{num1}{den1} \quad : disturbance\ input$$

دستورات زیر را در یک m-file جدید بنویسید:

```
%~~~~~~~~Open loop Response~~~~~~~~~
m1=2500;
m2=320;
k1=80000;
k2=500000;
b1 = 350;
b2 = 15020;
t=0:0.01:5;
nump=[(m1+m2) b2 k2];
denp=[(m1*m2) (m1*(b1+b2))+(m2*b1)
(m1*(k1+k2))+(m2*k1)+(b1*b2) (b1*k2)+(b2*k1) k1*k2];
'G1(s)'

printsys(nump,denp)
num1=[-(m1*b2) -(m1*k2) 0 0];
den1=[(m1*m2) (m1*(b1+b2))+(m2*b1)
(m1*(k1+k2))+(m2*k1)+(b1*b2) (b1*k2)+(b2*k1) k1*k2];
'G2(s)'

printsys(0.1*num1,den1)
figure(1)
sys1=tf(nump,denp);
sys2=tf(num1,den1);
step(sys1)
figure(2)
step(0.1*sys2)
axis([0 10 -0.1 0.1])
```

خروجی سیستم بدون کنترل فیدبک به صورت زیر خواهد بود:

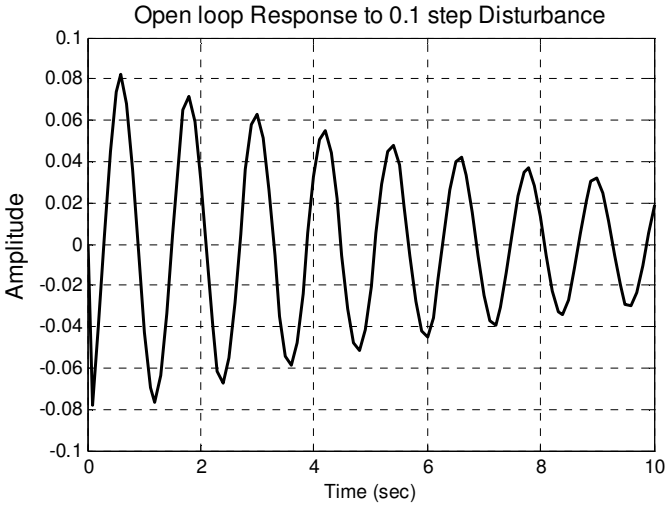
```
                          2820 s^2 + 15020 s + 500000
G1(s)=  ------------------------------------------------------------------------
        800000 s^4 + 38537000 s^3 + 1480857000 s^2 + 1376600000 s + 40000000000
```



شکل (۴-۲) پاسخ پله سیستم حلقه باز

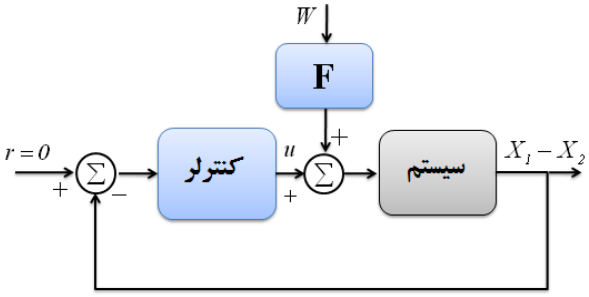
$$G2(s) = \frac{-3755000\ s^3 - 125000000\ s^2}{800000\ s^4 + 38537000\ s^3 + 1480857000\ s^2 + 1376600000\ s + 40000000000}$$

از روی گراف مشخص است که سیستم در اثر نیروی پله واحد از طرف عملگر به صورت یک سیستم زیر میرای بحرانی می باشد. یعنی افرادی که درون خودرو نشسته اند به مدت طولانی نزدیک 40 ثانیه نوسانات کوچکی را حس می‌کنند که این مقدار غیرقابل قبول است. از طرفی خطای حالت ماندگار سیستم حدود 0.013mm می‌باشد که باید آن را حذف کرد. برای حل این مساله باید از کنترل فیدبک استفاده کرد.



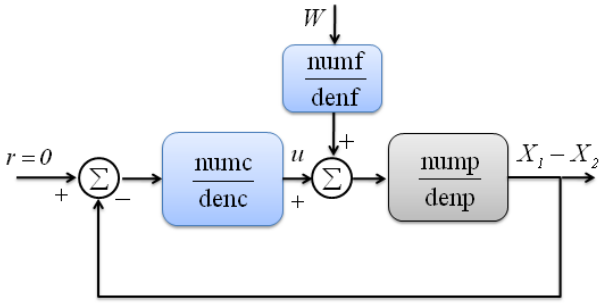
شکل (۴-۳) پاسخ سیستم به اغتشاش پله 0.1m

از روی این گراف می بینیم مدت زمان نوسانات طولانی و حدود 50 ثانیه است. از طرفی در اثر اعمال اغتشاش از طرف جاده به اندازه 10cm ، مقدار دامنه نوسانات حدود 8cm است که شرایط مطلوب ما را برآورده نمی-کند. برای حل این مشکل نیز باید از یک سیستم کنترل حلقه بسته با فیدبک واحد استفاده کنیم که نمودار بلوک نمودار آن در زیر نمایش داده شده است.



شکل (۴-۴) سیستم کنترل حلقه بسته با فیدبک واحد

برای سیستم، تابع تبدیل به صورت زیر تعریف می شود و به‌دنبال آن می‌توان بلوک نمودار سیستم تعلیق را به صورت زیر رسم کرد:



شکل (۴-۵) سیستم کنترل حلقه بسته با فیدبک واحد

$$plant = \frac{nump}{denp}$$

$$F \times plant = \frac{num1}{den1} = G_2(s)$$

$$\Rightarrow F = \frac{num1}{den1 \times plant} = \frac{num1 \times denp}{den1 \times nump} = \frac{num1}{nump} = \frac{numf}{denf}$$

(۱۰-۴)

۴-۲طراحی کنترلر PID

همانطور که قبلاً گفته شد می‌توانیم معادلات دینامیکی را به صورت تابع تبدیل بنویسیم:

$$G_1(s) = \frac{X_1(s) - X_2(s)}{U(s)} = \frac{(M_1 + M_2)s^2 + b_2 s + k_2}{\Delta}$$

$$G_2(s)=\frac{X_1(s)-X_2(s)}{W(s)}=\frac{-M_1b_2s^3-M_1k_2s^2}{\Delta}$$

$$\Delta=(M_1s^2+b_1s+k_1).(M_2s^2+(b_1+b_2)s+(k_1+k_2))-(b_1s+k_1).(b_1s+k_1)$$

بنابراین در ادامه m-file دستورات زیر را وارد می کنیم:

```
numf=num1;
denf=nump;
```

از طرفی تابع تبدیل کنترلر PID به صورت زیر می‌باشد:

$$PID: k_p+k_Ds+\frac{k_I}{s}=\frac{k_Ds^2+k_ps+k_I}{s}$$

که در آن، k_p بهره تناسبی، k_I بهره انتگرالی، و k_D بهره مشتق‌گیر می باشد. فرض می‌کنیم که برای طراحی کنترلر به هر 3 بهره نیاز داریم. حال باید مقدار دهی این بهره ها را شروع کنیم. بنابراین در ادامه m-file دستوارت زیر را وارد کنید:

```
KD=200000;
KP=800000;
KI=600000;
numc=[KD,KP,KI];
denc=[1 0];
```

حال باید رفتار خروجی سیستم را به ورودی پله ناشی از اغتشاش وارده از طرف جاده بررسی کنیم. با توجه به شکل (۴-۵) تابع تبدیل سیستم حلقه بسته به صورت زیر به‌دست می‌آید:

$$\begin{aligned}
&[\frac{numf}{denf}W-\frac{numc}{denc}(X_1-X_2)]\frac{nump}{denp}=X_1-X_2\\
&[\frac{numf}{denf}W-\frac{numc}{denc}(X_1-X_2)]=(X_1-X_2)\frac{denp}{nump}\\
&\frac{numf}{denf}W=(\frac{denp}{nump}+\frac{numc}{denc})(X_1-X_2)\\
&\Rightarrow\frac{(X_1-X_2)}{W}=\frac{nump.numf.denc}{denf\,(denp.denc+nump.numc)}=\frac{numf.denc}{(denp.denc+nump.numc)}
\end{aligned}\qquad(۴-۱۱)$$

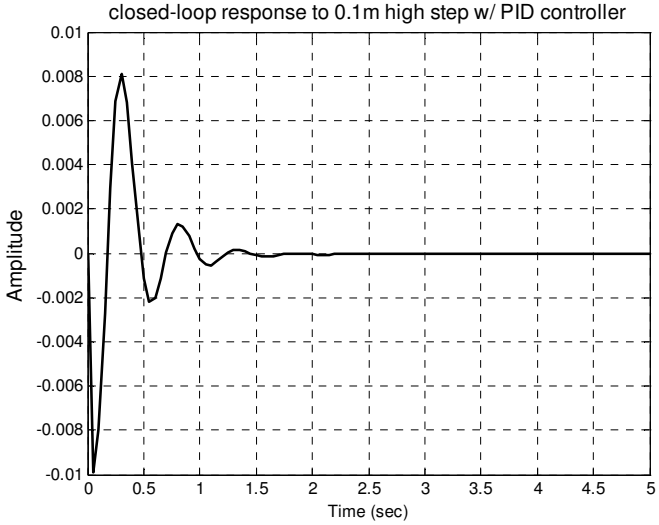
برای نوشتن این تابع تبدیل در نرم افزار MATLAB نیاز داریم که در ابتدا function زیر را بنویسیم:

```
Function [poly]=polyadd(poly1,poly2)
%polyadd(poly1,poly2) adds two polynominals possibly of
uneven length
if length(poly1)<length(poly2)
  short=poly1;
  long=poly2;
else
  short=poly2;
```

```
  long=poly1;
end
mz=length(long)-length(short);
if mz>0
  poly=[zeros(1,mz),short]+long;
else
  poly=long+short;
end
```

از این تابع برای جمع کردن دو چند جمله ای با طول غیریکسان استفاده می‌کنیم. در ادامه دستورات زیر را به ادامه m-file اضافه می کنیم:

```
numa=conv(numf,denc);
dena=polyadd(conv(denp,denc),conv(nump,numc));
sys=tf(numa,dena);
t=0:0.05:5;
figure(3)
step(0.1*sys,t)
title('closed-loop response to 0.1m high step w/ PID
controller')
```

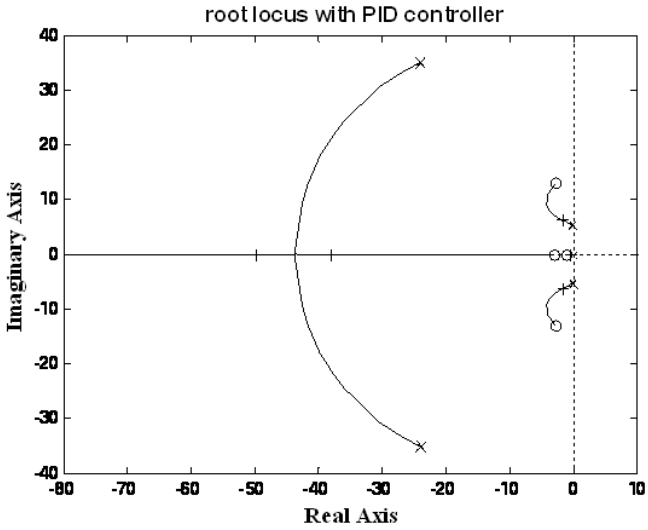


شکل (۴-۶) پاسخ پله سیستم همراه با کنترلر PID

از روی نمودارمشخص است که مقدار جهش حدود 8% است که حدود 3% بیشتر از مقدار مطلوب می باشد. ولی زمان نشست مطلوب و زیر5 ثانیه می‌باشد. برای انتخاب بهره های مناسب برای کنترلر PID باید یک قطب و دو صفر برای آن پیدا کنیم. قطب این کنترلر در صفر قرار گرفته است، بنابراین یکی از صفرها را باید نزدیک قطب در مبدا به طور مثال در 1 و صفر دیگر را در 3 قرار دهیم. دستورات زیر را به ادامه m-file اضافه کنید.

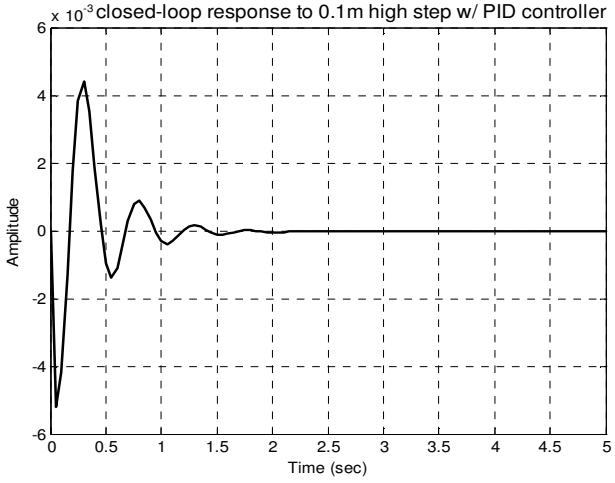
```
z1=1;z2=3;p1=0;
numc=conv([1 z1],[1 z2])
```

```
denc=[1 p1]
num2=conv(nump,numc);
den2=conv(denp,denc);
rlocus(num2,den2)
title('root locus with PID controller')
[K,p]=rlocfind(num2,den2)
```



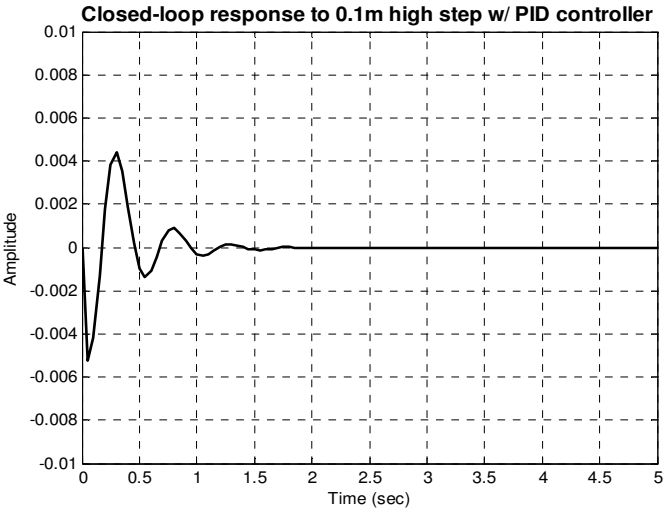
شکل (۴-۷) مکان هندسی ریشه ها با کنترلر PID

تا اینجا تابع تبدیل سیستم حلقه بسته را به دست آورده ایم. مساله اصلی تغییر بهره‌های تناسبی، انتگرالی، و مشتق‌گیر است. از روی شکل (۴-۶) مشخص است که به میرایی بیشتری نیاز داریم. بنابراین مقادیر بهره‌ها را دو برابر می کنیم تا ببینیم چه اتفاقی می افتد. پس به m-file برگشته مقادیر بهره‌ها را تغییر دهید.



شکل (۴-۸) پاسخ پله سیستم همراه با کنترلر PID

```
axis([0 5 -.01 .01])
```



شکل (۴-۹) پاسخ پله سیستم همراه با کنترلر PID

همانطور که از شکل ها بر می‌آید مقدار جهش و زمان نشست به مقدار مطلوب ما رسیده است.

۴ -۳ رسم مکان هندسی ریشه ها (Root Locus)

هدف ما در طراحی کنترل فیدبک این است که خروجی سیستم یعنی $X_1 - X_2$ جهشی کمتر از 5% داشته و درکمتر از 5 ثانیه نوسانات آن میرا شود. یعنی زمانی که خودرو از یک دست انداز 10cm عبور می‌کند، دامنه نوسانات باید در محدوده $\pm 5\,mm$ بوده و بعد از 5 ثانیه به حرکت آرام خود ادامه دهد. ایده اصلی در طراحی به کمک مکان هندسی ریشه ها پیدا کردن پاسخ سیستم حلقه بسته از روی نمودار مکان هندسی ریشه‌های سیستم حلقه باز می‌باشد، به طوری که با اضافه کردن یک سری صفر و قطب به سیستم اصلی، پاسخ سیستم حلقه بسته تصحیح شود. برای مدل سازی سیستم از دستورات زیر استفاده کردیم:

```
m1=2500;m2=320;k1 = 80000;k2 = 500000;b1 = 350; b2 = 15020;
nump=[(m1+m2) b2 k2];
denp=[(m1*m2) (m1*(b1+b2))+(m2*b1)
(m1*(k1+k2))+(m2*k1)+(b1*b2) (b1*k2)+(b2*k1) k1*k2];

num1=[-(m1*b2) -(m1*k2) 0 0];
den1=[(m1*m2) (m1*(b1+b2))+(m2*b1)
(m1*(k1+k2))+(m2*k1)+(b1*b2) (b1*k2)+(b2*k1) k1*k2];
numf=num1;
denf=nump;
```

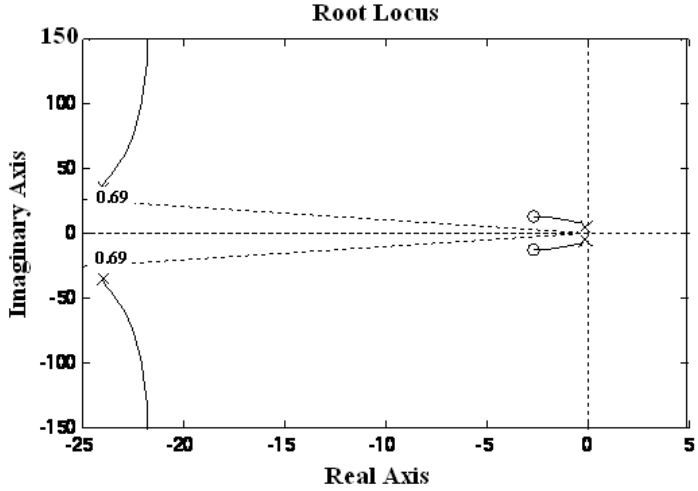
حال با داشتن این اطلاعات می‌توانیم مکان هندسی ریشه ها را طراحی کنیم ولی ابتدا باید قطب های سیستم را به‌دست آوریم.

R=roots(denp)

```
R =
 -23.9758 +35.1869i
 -23.9758 -35.1869i
  -0.1098 + 5.2504i
  -0.1098 - 5.2504i
```

قطب های غالب $-0.1098 \pm 5.2504i$ هستند که نزدیک محور موهومی می‌باشند. مقدار میرایی نیز بسیار کوچک است. برای رسم مکان هندسی ریشه ها به صورت زیر عمل می کنیم:

```
rlocus(nump,denp)
zeta=-log(0.05)/sqrt(pi^2+(log(0.05)^2))
sgrid(zeta,0)
```



شکل (۴-۱۰) مکان هندسی ریشه ها همراه با خطوط نسبت میرایی ثابت

دستور sgrid به دو آرگومان نیاز دارد. نسبت میرایی $\zeta = 0.7$ که متناسب با جهش 5% است. از روی نمودار می‌بینید که دو جفت از قطب و صفرها نزدیک محور موهومی قرار دارند. این جفت قطب و صفرها تقریباً روی محور موهومی قرار دارند و باید تا آنجا که امکان دارد قطب ها را به سمت چپ بکشانیم و از ناپایداری اجتناب کنیم. مجبوریم دو تا صفر نزدیک این قطب های سیستم جبران نشده قرار دهیم تا اثر قطب ها را خنثی کنیم. دو قطب هم بسیار دورتر از محور موهومی قرار می دهیم تا پاسخ سریع تری داشته باشیم.

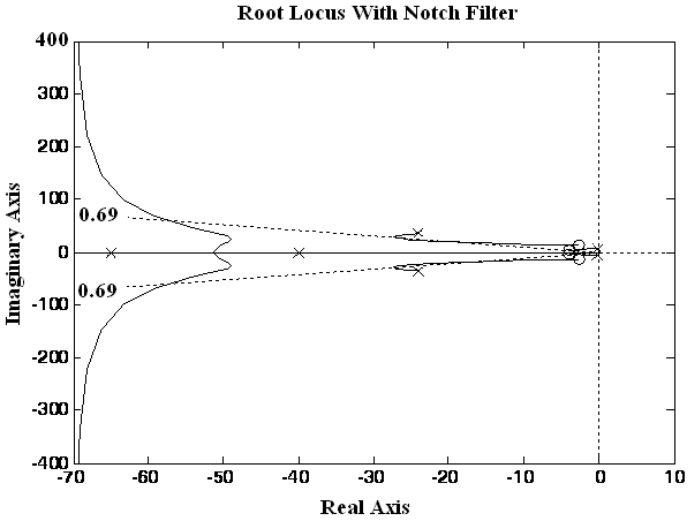
۴-۳-۱ اضافه کردن Notch Filter

برای رسیدن به هدف، به نظر می‌رسد که باید از Notch Filter (دو تا کنترلر پیش فاز) استفاده کنیم. پس باید قطب ها را در 40 و65 و صفرها را در $4 \pm 3.5i$ قرار دهیم. پس دستورات زیر را به ادامه M-file اضافه کنید:

```
z1=4+3.5i; z2=4-3.5i;
p1=40; p2=65;
```

```
numc=conv([1 z1],[1 z2]);
denc=conv([1 p1],[1 p2]);
```

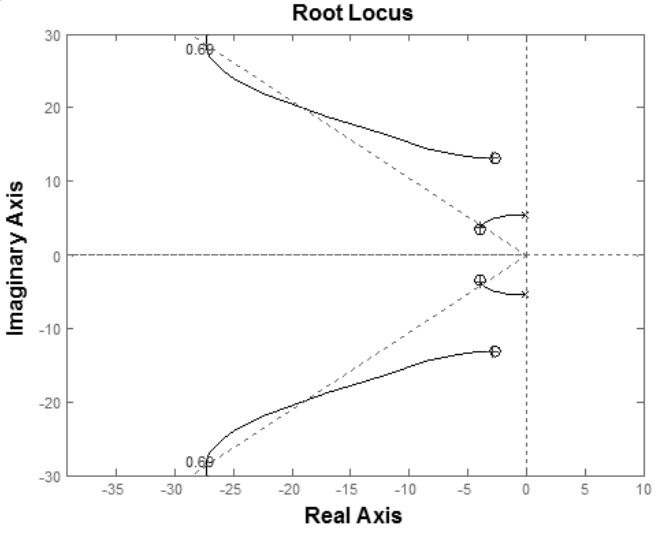
توجه: علامت % را در جلوی دستور rlocus(nump,denp) اضافه کنید و سپس برنامه را اجرا کنید.



شکل (۴-۱۱) مکان هندسی ریشه ها همراه با Notch Filter

برای دیدن جزئیات بیشتر باید محدوده نمودار را تغییر دهیم.

```
rlocus(conv(nump,numc),conv(denp,denc))
axis([-40 10 -30 30])
z=-log(0.05)/sqrt(pi^2+(log(0.05)^2))
sgrid(z,0)
```



شکل (۴-۱۲) مکان هندسی ریشه ها همراه با Notch Filter

حالا باید بهره ای را انتخاب کنیم که ملزومات طراحی ما را برآورده کند. دوباره یادآوری می‌کنیم که می‌خواهیم زمان نشست و جهش کوچکی داشته باشیم. برای رسیدن به یک جهش کوچک و پاسخ سریع نیازاست نقطه ای نزدیک محور حقیقی و دور از محور موهومی در نظر بگیریم. برای انجام این کار از دستور rclocfind استفاده می کنیم. پس دستورات زیر را به ادامه m-file اضافه کنید:

```
[k,poles]=rlocfind(conv(nump,numc),conv(denp,denc))
```

حال نقطه ای را که نمایش داده شده است را انتخاب کنید تا مقدار بهره پیدا شود.

```
selected_point = -2.8448 +13.1366i
```

حال این بهره را به سیستم اضافه کنید:

```
numc=k*numc;
```

همانطور که در بخش PID توضیح داده شد، تابع تبدیل حلقه بسته به صورت زیر به‌دست می آید:

$$\frac{(X_1 - X_2)}{W} = \frac{nump.numf.denc}{denf\,(denp.denc + nump.numc)} = \frac{numf.denc}{(denp.denc + nump.numc)}$$

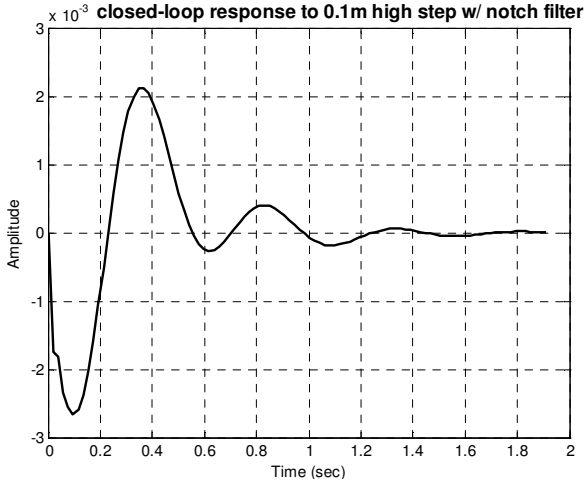
برای به‌دست آوردن تابع تبدیل حلقه بسته از W و X1-X2 دستورات زیر را به ادامه M-file اضافه کنید:

```
numa=conv(conv(numf,nump),denc);
dena=conv(denf,polyadd(conv(denp,denc),conv(nump,numc)));
```

حال باید پاسخ پله سیستم حلقه بسته را به ورودی پله با استفاده از این جبران کننده رسم کنیم. به خاطر داشته باشید که ورودی **0.1m** به عنوان اغتشاش تلقی می شود.

```
step(0.1*numa,dena)
title('closed-loop response to 0.1m high step w/ notch
filter')
```

با اجرای برنامه نمودار زیر حاصل می شود.



شکل (۴-۱۳) پاسخ پله سیستم حلقه بسته با بهره انتخابی

همانطور که از روی شکل مشخص است بعد از اینکه خودرو از یک دست انداز 0.1m عبور می کند ماکزیمم انحراف بدنه خودرو از جاده یا چرخ ها حدود 3 mm است و بعد از 2 ثانیه نیز نوسانات متوقف می شود. این جواب کاملاً رضایت بخش است. توجه کنید این جواب، تنها جواب مساله نیست و با استفاده از مکان هندسی ریشه ها می‌توانیم جبران‌کننده های دیگری را طراحی کنیم. فقط کافی است به مکان هندسی ریشه های حلقه باز برگردیم و برای رسیدن به پاسخ بهتر، قطب ها و صفرها را جابه‌جا کنیم.

۴-۴ پاسخ فرکانسی

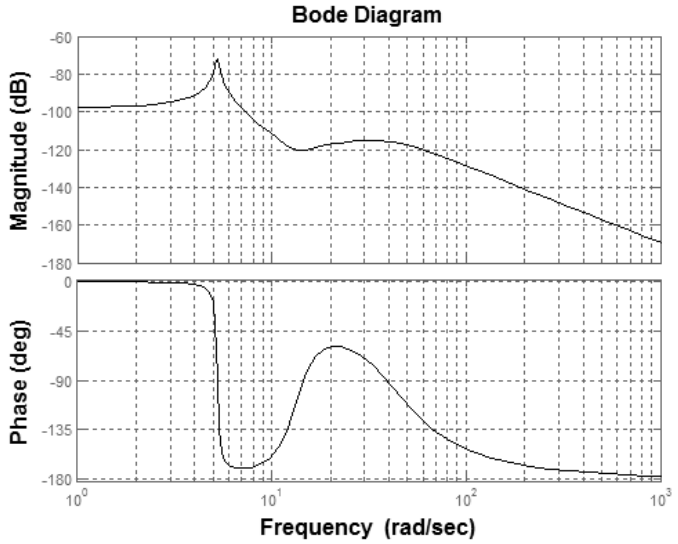
در قسمت مدل سازی، تابع تبدیل سیستم به‌دست آمد، حال مدل سیستم را در محیط MATLAB نمایش می دهیم. دستورات زیر را در یک m-file وارد کنید:

```
m1=2500;
m2=320;
k1 = 80000;
k2 = 500000;
b1 = 350;
b2 = 15020;
nump=[(m1+m2) b2 k2]
denp=[(m1*m2) (m1*(b1+b2))+(m2*b1)
(m1*(k1+k2))+(m2*k1)+(b1*b2) (b1*k2)+(b2*k1) k1*k2]

num1=[-(m1*b2) -(m1*k2) 0 0]
den1=[(m1*m2) (m1*(b1+b2))+(m2*b1)
(m1*(k1+k2))+(m2*k1)+(b1*b2) (b1*k2)+(b2*k1) k1*k2]

numf=num1;
denf=nump;
```

ایده اصلی طراحی بر مبنای پاسخ فرکانسی بر این است که با رسم نمودار بود[1] تابع تبدیل حلقه باز، تخمینی از پاسخ سیستم حلقه بسته به‌دست آید. اضافه کردن کنترلر به سیستم، نمودار بود حلقه باز را تغییر می دهد. پس در ابتدا نمودار بود سیستم حلقه باز را رسم می‌کنیم:

```
bode(nump,denp)
```

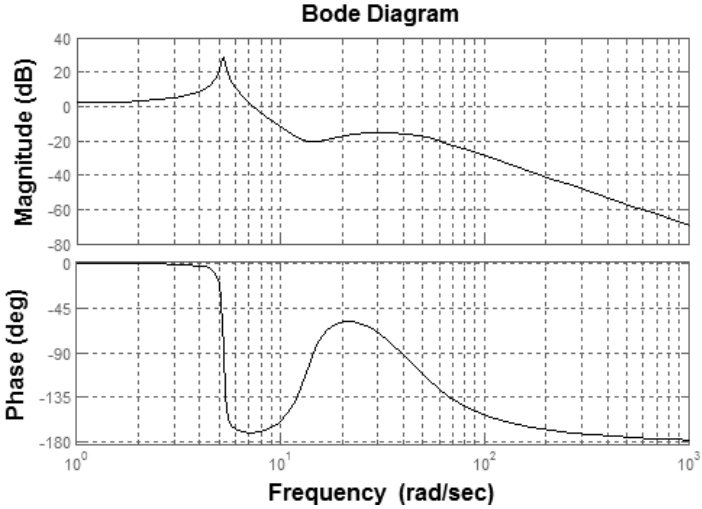
[1] Bode diagram



شکل (۴-۱۴) نمودار بود سیستم حلقه باز

برای راحتی در نمایش سیستم ها با فرکانس‌های طبیعی مختلف قبل از رسم نمودار بود، آنها را مقیاس‌بندی و نرمالیزه می کنیم تا اینکه مجانب فرکانس پایین مقدار 0dB داشته باشد. عمل نرمالیزه کردن با تنظیم بهره k انجام می‌گیرد. تاثیر بهره k در این است که با افزایش و یا کاهش k منحنی دامنه نمودار بود به سمت بالا یا پایین شیفت پیدا می‌کند. ولی این بهره هیچ تاثیری روی فاز سیستم ندارد. از روی نمودار مشخص است که باید بهره k=100dB یا 100,000 باشد تا منحنی دامنه را به 0dB در 0.1rad/sec برساند.

```
nump=100000*nump
```



شکل (۴-۱۵) نمودار بود سیستم حلقه باز و جابه‌جایی نمودار دامنه به سمت بالا

۴-۴-۱ اضافه کردن دو کنترلر Lead

از نمودار فاز واضح است که منحنی در 5rad/sec مقعر است. بنابراین سعی می کنیم در ابتدا یک فاز مثبت به این ناحیه اضافه کنیم. از آنجایی که حد فاز زیاد به جهش کم منجر می شود، پس قصد داریم در حول و حوش فرکانس 5rad/sec حداقل 140 درجه فاز مثبت داشته باشیم. ولی از آنجاکه کنترلر Lead بیش از 90 درجه فاز مثبت اضافه نمی‌کند، بنابراین از دو کنترلر Lead استفاده می‌کنیم.

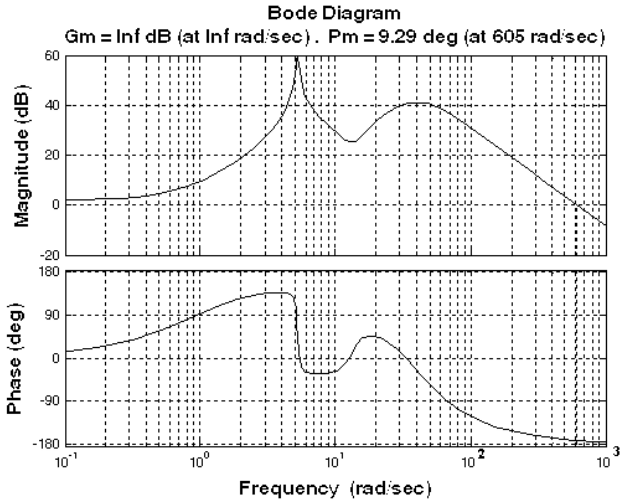
$$a=\frac{1-\sin 70}{1+\sin 70}=0.031091$$

پس با هر کنترلر Lead یک فاز 70درجه ای به سیستم اضافه می‌کنیم. از طرفی می‌خواهیم که این فاز در فرکانس 5rad/s اضافه شود. پس باید ثابت T را پیدا کنیم.

$$T=\frac{1}{w\sqrt{a}}=\frac{1}{5\sqrt{0.03109}}=1.134$$
$$aT=0.0352$$

حال دو کنترل Lead را به سیستم اضافه کنید و نمودار بود حاصل را مشاهده نمائید.

```
numc=conv([1.134 1], [1.134 1]);
denc=conv([0.0352 1], [0.0352 1]);
margin(conv(nump,numc),conv(denp,denc))
```

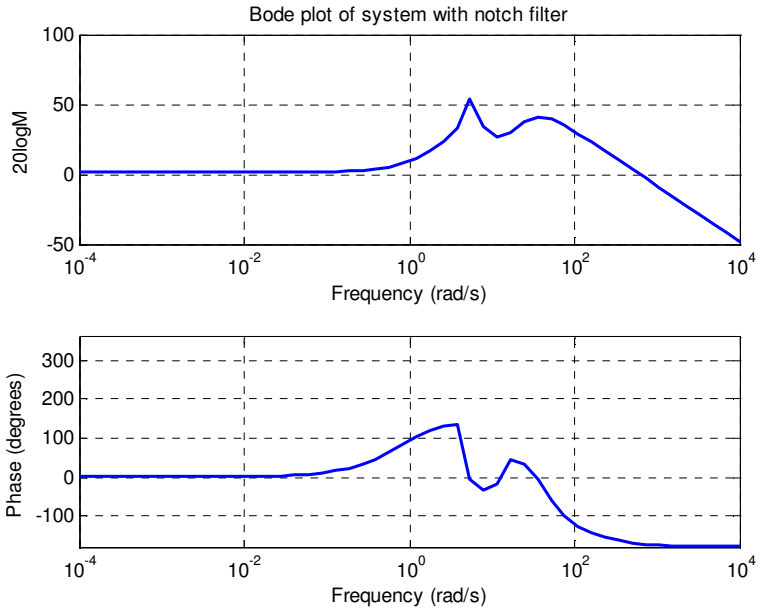


شکل (۴-۱۶) نمودار بود سیستم همراه با دو جبران ساز Lead

از آنجاکه نمودار بود محدوده فازی معینی دارد، پس تا حدودی بازه را پهن تر می‌کنیم.

```
w=logspace(-4,4);
[mag,phase,w] = bode(conv(nump,numc),conv(denp,denc),w);
subplot(2,1,1);
semilogx(w,20*log10(mag));
grid
title('Bode plot of system with notch filter')
xlabel('Frequency (rad/s)')
```

```
ylabel('20logM')
subplot(2,1,2);
semilogx(w,phase);
axis([1e-4, 1e4, -180, 360])
grid
xlabel('Frequency (rad/s)')
ylabel('Phase (degrees)')
```



شکل (۴-۱۷) نمودار بود سیستم همراه با دو جبران ساز Lead

از روی نمودار بود می‌بینیم که قسمت مقعر نمودار فازی بالای 180- درجه است و حد فاز به اندازه کافی بزرگ است تا معیارهای طراحی ما را برآورده کند. حال اجازه دهید ببینیم که خروجی $X_1 - X_2$ به ورودی اعمالی از طرف جاده W چگونه پاسخ می دهد؟ تابع تبدیل سیستم حلقه بسته به صورت زیر می باشد:

$$\frac{(X_1 - X_2)}{W} = \frac{nump.numf.denc}{denf\,(denp.denc + nump.numc)} = \frac{numf.denc}{(denp.denc + nump.numc)}$$

پس دستورات زیر را وارد کنید:

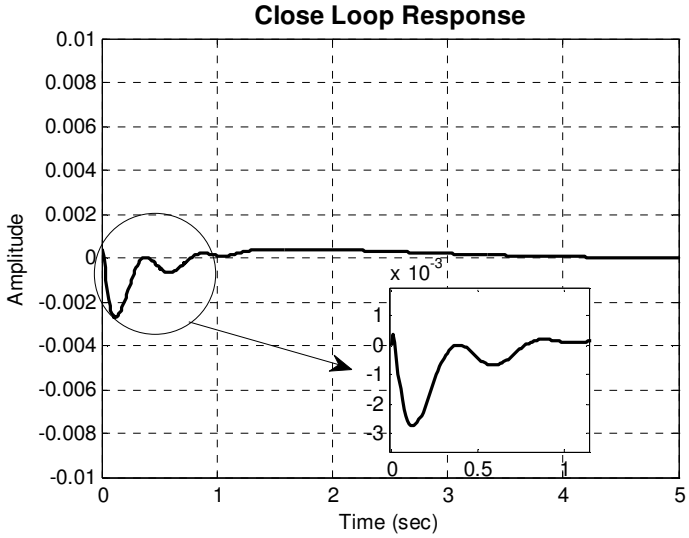
```
numa=conv(numf,denc);
dena=polyadd(conv(denp,denc),conv(nump,numc));
```

۴-۴-۲ رسم پاسخ حلقه بسته

حال باید ببینیم پاسخ سیستم حلقه بسته به چه صورت می باشد. توجه کنید که ما حداکثر ورودی 0.1m را به صورت پله به سیستم اعمال می‌کنیم .

```
t=0:0.01:5;
```

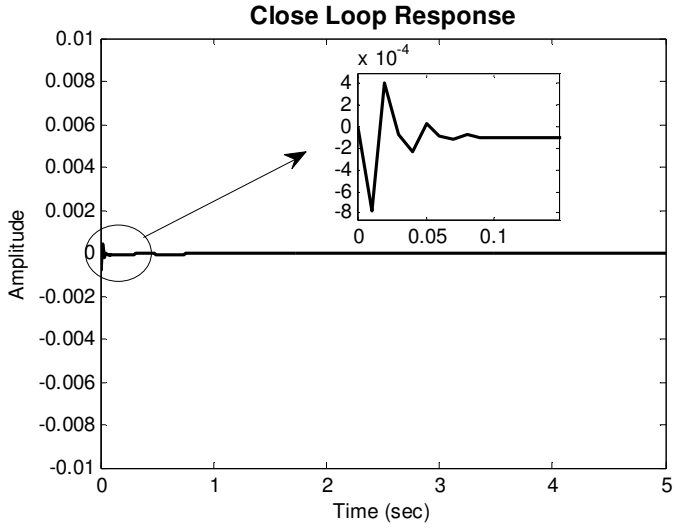
```
step(0.1*numa,dena,t)
axis([0 5 -.01 .01])
```



شکل (۴-۱۸) پاسخ پله سیستم حلقه بسته

دامنه پاسخ بسیار کوچکتر از چیزی است که انتظارش را داشتیم و زمان نشست کمتر از 5 ثانیه است . از نمودار بود می‌بینیم که افزایش بهره باعث افزایش فرکانس W_g می شود و پاسخ سریعتر می‌شود. پس بهره را افزایش می‌دهیم تا پاسخ بهتری بگیریم. درm-file تغییر زیر را اعمال کنید و برنامه را دوباره اجرا کنید.

```
numc=4*conv([3.1483 1],[3.1483 1])
```



شکل (۴-۱۹) پاسخ پله سیستم حلقه بسته

۴ –۵ طراحی فضای حالت

همانطور که قبلاً گفته شد، هدف این است که وقتی از طرف جاده یک اغتشاش به صورت پله به سیستم وارد می شود، خروجی $X_1 - X_2$ زمان نشستی کمتر از 5 ثانیه داشته باشد و مقدار جهش آن کمتر از 5% باشد. یعنی زمانی که خودرو 10cm بالا و پایین می رود، بدنه فقط به اندازه $\pm 5mm$ به سمت بالا و پایین حرکت کند و در مدت 5 ثانیه متوقف شود.

معادلات دینامیکی به فرم فضای حالت به صورت زیر نوشته می شود:

$$\begin{bmatrix} \dot{x}_1 \\ \ddot{x}_1 \\ \dot{y}_1 \\ \ddot{y}_1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ \frac{b_1 b_2}{M_1 M_2} & 0 & \left(\frac{-b_1}{M_1}\left(\frac{b_1}{M_1}+\frac{b_1}{M_2}+\frac{b_2}{M_2}\right)-\frac{k_1}{M_1}\right) & \frac{-b_1}{M_1} \\ \frac{-b_2}{M_2} & 0 & -\left(\frac{b_1}{M_1}+\frac{b_1}{M_2}+\frac{b_2}{M_2}\right) & 1 \\ \frac{-k_2}{M_2} & 0 & \left(\frac{k_1}{M_1}+\frac{k_1}{M_2}+\frac{k_2}{M_2}\right) & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ \dot{x}_1 \\ y_1 \\ \dot{y}_1 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ \frac{1}{M_1} & \frac{b_1 b_2}{M_1 M_2} \\ 0 & \frac{-b_2}{M_2} \\ (\frac{1}{M_1}+\frac{1}{M_2}) & \frac{-k_2}{M_2} \end{bmatrix} \begin{bmatrix} U \\ W \end{bmatrix}$$

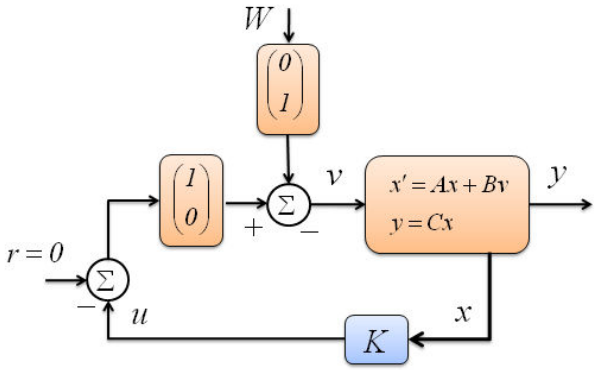
$$y = \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ \dot{x}_1 \\ y_1 \\ \dot{y}_1 \end{bmatrix} + \begin{bmatrix} 0 & 0 \end{bmatrix} \begin{bmatrix} U \\ W \end{bmatrix}$$

پس ماتریس های A,B,C,D را در محیط MATLAB وارد کنید:

```
m1=2500;m2=320;k1 = 80000;k2 = 500000;b1 = 350;b2 = 15020;
A=[0, 1 ,0 ,0;-(b1*b2)/(m1*m2),
0,((b1/m1)*((b1/m1)+(b1/m2)+(b2/m2)))-(k1/m1),-(b1/m1); -
b2/m2, 0  -((b1/m1)+(b1/m2)+(b2/m2)) 1;-k2/m2 ,0
,-((k1/m1)+(k1/m2)+(k2/m2)) 0];

B=[0                    0
   1/m1                 (b1*b2)/(m1*m2)
   0                   -(b2/m2)
   (1/m1)+(1/m2)       -(k2/m2)];

C=[0   0   1   0];
D=[0    0];
```

۴-۵-۱ طراحی کنترلر با فیدبک کلیه متغیرهای حالت

دراین قسمت قصد داریم یک کنترلر، با فیدبک کلیه متغیرهای حالت طراحی کنیم. فرض می‌کنیم که تمام متغیرها قابل اندازه‌گیری هستند (هرچند این فرض صحیح نیست). شماتیک بلوک نمودار در زیر نمایش داده شده است:



شکل (۴-۲۰) کنترل به روش فیدبک کلیه متغیرهای حالت

معادله مشخصه سیستم حلقه بسته از دترمینان $(sI-(A-B[1\ 0]^T K)$ به‌دست می‌آید. توجه کنید که رابطه بالا برابر $(sI-(A-BK))$ نیست چون کنترلر K فقط نیروی ورودی U را کنترل می‌کند و اثری روی اغتشاش W جاده ندارد.

ماتریس B یک ماتریس 2×4 است و فقط ستون اول ماتریس B را برای کنترل نیروی U می خواهیم. در این مثال برای حذف خطای حالت ماندگار نیاز داریم تا یک حالت دیگر را نیز اضافه کنیم، پس:

$$\int X_1 - X_2 = \int y_1 \qquad (۱۲-۴)$$

زیرا در واقعیت نیز خودرو به نقطه تعادل صفر می رسد، پس متغیر های حالت جدید به صورت زیر خواهند بود:

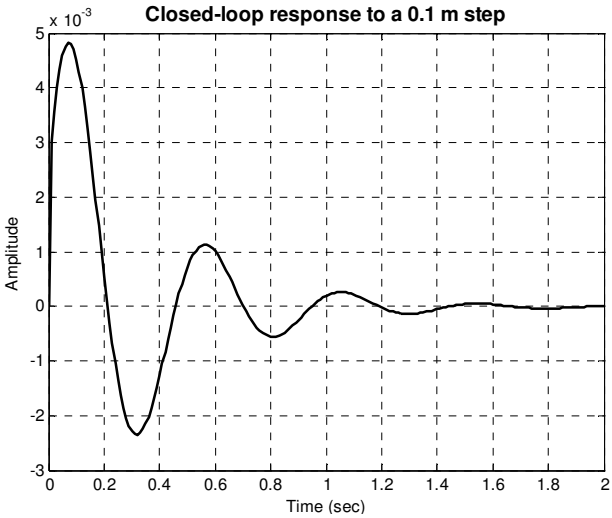
$$X = [X_1\ \ X_2\ \ y_1\ \ y_2\ \ \int y_1] \qquad (۱۳-۴)$$

و معادله فضای حالت جدید به صورت زیر خواهد بود:

```
Aa = [[A,[0 0 0 0]'];[C, 0]];
Ba = [B;[0 0]];
Ca = [C,0];
Da = D;
```

برای کنترلر K مقادیر زیر را در یک m-file وارد کنید و سپس برای به‌دست آوردن پاسخ سیستم حلقه بسته دستورات بعدی را اضافه کنید:

```
K = [0 2.3e6 5e8 0 8e6]
t=0:0.01:2;
step(Aa-Ba(:,1)*K,-0.1*Ba,Ca,Da,2,t)
title('Closed-loop response to a 0.1 m step')
```



شکل (۴-۲۱) پاسخ پله سیستم با فیدبک کلیه متغیر های حالت

از روی پاسخ کاملاً واضح است که سیستم تعلیق خودرو در کمتر از 2 ثانیه از حرکت متوقف شده، مقدار جهش نیز کمتر از 5% می‌باشد و خطای حالت ماندگار نیز برابر صفر می گردد. پس تمام ملزومات طراحی ما برآورده می‌گردد.

۴ –۶ طراحی کنترلر دیجیتالی

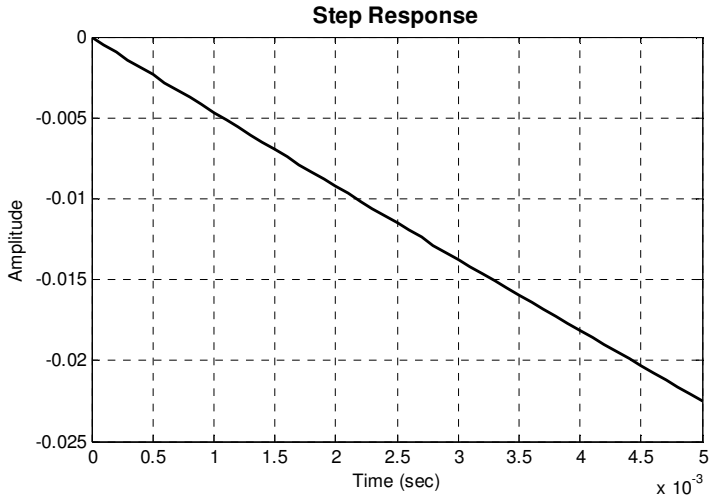
دراین قسمت می‌خواهیم یک کنترلر فضای حالت دیجیتالی برای سیستم تعلیق خودرو طراحی کنیم تا معیارهای طراحی را برآورده کند. پس در ابتدا باید مدل گسسته فضای حالت سیستم را به‌دست آوریم و سپس با استفاده از روش جایابی قطب ها کنترلر را طراحی کنیم.

۴-۶-۱ انتخاب زمان نمونه برداری

اولین قدم در طراحی کنترل زمان گسسته این است که سیستم پیوسته را به سیستم گسسته تبدیل کنیم. دراین مثال انتخاب زمان نمونه برداری بسیار با اهمیت است. زیرا ورودی پله که از طرف جاده به خودرو اعمال می شود بسیار سریع است. بنابراین فنر K2 و دمپر b2 فشرده می‌شوند. ولی از آنجایی که جرم سیستم نسبتاً کم بوده و فنر نیز تا حدودی سفت است بنابراین سیستم تعلیق سریع بالا می رود و مقدار X2 به طور ناگهانی افزایش پیدا می کند. از آنجایی که کنترلر فقط می تواند تاثیر اغتشاشات را بعد از یک پریود کامل ببیند، بنابراین زمان نمونه برداری باید به اندازه کافی کوچک باشد به‌طوری‌که خروجی X1-X2 بیشتر از 5% مقدار دلخواه در یک دوره زمانی نشود. فاصله های زمانی را به صورت برداری در بازه 0 تا 0.005 تنظیم کنید. دستور زیر را در یک m-file جدید کپی کنید:

```
m1=2500;m2=320;
k1 = 80000;k2 = 500000;
b1 = 350;b2 = 15020;
```

```
A=[0, 1 ,0 ,0;-(b1*b2)/(m1*m2)
,0,((b1/m1)*((b1/m1)+(b1/m2)+(b2/m2)))-(k1/m1),-(b1/m1);-
b2/m2;0  -((b1/m1)+(b1/m2)+(b2/m2))                    1;-k2/m2
,0  ,-((k1/m1)+(k1/m2)+(k2/m2)) 0];

B=[0                  0
   1/m1               (b1*b2)/(m1*m2)
   0                 -(b2/m2)
   (1/m1)+(1/m2)     -(k2/m2)];
C=[0   0   1   0];
D=[0    0];

step(A,.1*B,C,D,2,0:0.0001:.005);
```



شکل (۴-۲۲) پاسخ پله سیستم برای نمایش سرعت پاسخگویی

نمودار (۴-۲۲) نشان می‌دهد که فنر k1 به سرعت فشرده می شود و با ورودی پله اعمالی 0.1m، در کمتر از 0.001 ثانیه، بیشتر از 5mm جابه‌جا می شود. بنابراین مقدار زمان نمونه برداری را روی 0.0005 ثانیه تنظیم کنید. بعد از انتخاب زمان نمونه برداری باید با استفاده از دستور c2dm سیستم پیوسته را به سیستم گسسته تبدیل کنیم. دستورات زیر را به ادامه M-file اضافه کنید:

```
T=.0005;
[Ad Bd Cd Dd]=c2dm(A,B,C,D,T,'zoh')
```

جواب زیر برای مدل فضای حالت گسسته به‌دست می‌آید:

```
Ad =
    1.0000    0.0005    0.0000    0.0000
   -0.0035    1.0000   -0.0124   -0.0001
    0.0234    0.0000    0.9760    0.0005
```

```
    0.7705    0.0002   -0.9112    0.9998
Bd =
    0.0000    0.0000
    0.0000    0.0035
    0.0000   -0.0234
    0.0000   -0.7705
Cd =
     0     0     1     0
Dd =
     0     0
```

۴-۶-۲ اضافه کردن انتگرال‌گیر

برای رسیدن به خطای حالت ماندگار صفر نیاز داریم که یک انتگرال‌گیر به سیستم اضافه کنیم. این انتگرال‌گیر را به صورت سری به سیستم اضافه می‌کنیم. انتگرال‌گیر را باید به فرم فضای حالت بنویسیم و سپس از دستور series استفاده کنیم. نمایش فضای حالت انتگرالی در زمان گسسته با یک تقریب ذوزنقه ای انتگرالی در یک پریود نمونه برداری به صورت زیر می باشد:

$$x(k+1) = x(k) + Tu(k)$$
$$y(k) = x(k) + \frac{T}{2}u(k) \qquad (۱۴-۴)$$

دستورات زیر را به ادامه M-file اضافه کنید:

```
Ai=1;
Bi=T;
Ci=1;
Di=T/2;
[Ada,Bda,Cda,Dda]=series(Ad,Bd,Cd,Dd,Ai,Bi,Ci,Di)
Cda=[Cd 0]
```

ماتریس حاصل 5×5 می باشد. ساختار کنترلر دیجیتالی مشابه ساختار کنترلر فضای حالت زمان پیوسته است. برای قرار دادن قطب های سیستم حلقه بسته در محل دلخواه و به‌دست آوردن بهره ماتریس K باید از دستور place استفاده کنیم. حال باید تصمیم بگیریم که قطب های سیستم حلقه بسته را در کجا قرار دهیم. ما می توانیم این پنج قطب را در جایی قرار دهیم که صفر های سیستم را خنثی کند. پس ابتدا باید صفر های سیستم را پیدا نماییم. برای این کار فضای حالت دیجیتالی را به تابع تبدیل تبدل کرده و سپس ریشه های صورت را پیدا می کنیم. برای این کار از دستور ss2tf استفاده می‌کنیم. دستورات زیر را به ادامه m-file اضافه کنید:

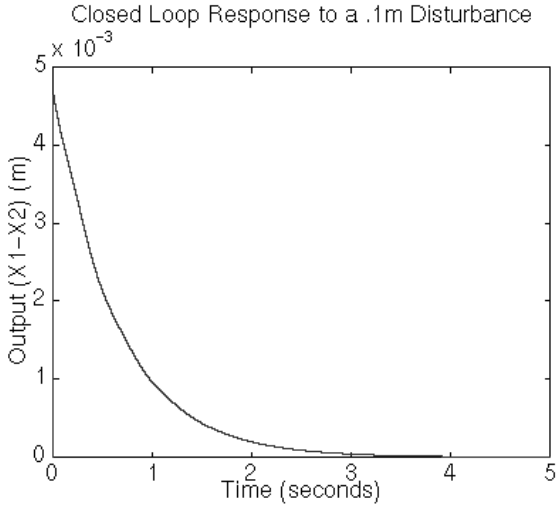
```
[num,den]=ss2tf(Ad,Bd,Cd,Dd,1);
zeros=roots(num)
```

ما این سه صفر رابه عنوان سه قطب مطلوب سیستم حلقه بسته خود انتخاب می‌کنیم. بنابراین دو قطب باقی می‌ماند یکی از آنها را در 0.9992 و دیگری را در $z = 0.2$ قرار می دهیم. پس دستورات زیر را به ادامه M-file اضافه کنید:

```
p1=zeros(1);
p2=zeros(2);
p3=zeros(3);
p4=.9992;
p5=.5;
K=place(Ada,Bda*[1;0],[p1 p2 p3 p4 p5])
```

حال می توانید با استفاده از دستور dstep پاسخ سیستم حلقه بسته را به‌دست آورید. از آنجایی که بردار حالت در ماتریس K ضرب شده و فقط یک سیگنال u را بر می گرداند، باید یک ردیف صفر به ماتریس k اضافه کنیم. برای این کار ماتریس K را در $[1 \quad 0]^T$ ضرب می کنیم.

```
Yout=dstep(Ada-Bda*[1 0]'*K,-.1*Bda,Cda,-.1*Dda,2,10001);
t=0:.0005:5;
stairs(t,yout);
```



شکل (۴-۲۳) پاسخ سیستم حلقه بسته به اغتشاشات وارده از طرف جاده

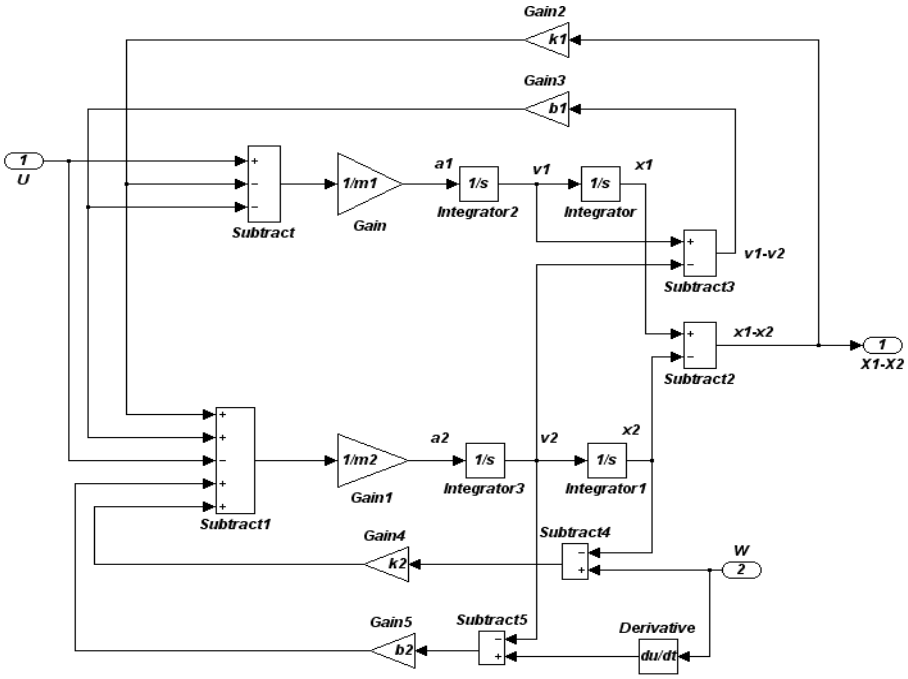
همان طور که از شکل مشخص است مقدار جهش کمتر از 5mm و زمان نشست حدود 5 ثانیه است.

۴ -۷ طراحی کنترلر با فیدبک تمامی متغیر های حالت در محیط سیمولینک

همانطور که قبلاً گفته شد معادلات حاکم بر سیستم، معادلات نیوتن است که می توان این معادلات دینامیکی را به صورت زیر نوشت:

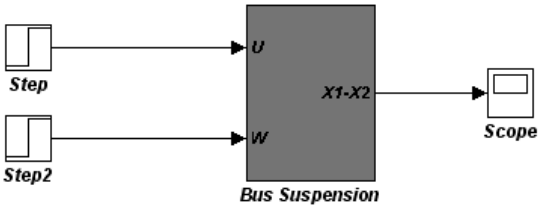
$$\begin{cases} M_1\ddot{x}_1 = -b_1(\dot{x}_1 - \dot{x}_2) - k_1(x_1 - x_2) + u \\ M_2\ddot{x}_2 = b_1(\dot{x}_1 - \dot{x}_2) + k_1(x_1 - x_2) + b_2(\dot{w} - \dot{x}_2) + k_2(w - x_2) - u \end{cases}$$

حال باید در محیط سیمولینک این مدل دینامیکی را بسازیم. پس مطابق شکل (۴-۲۴) بلوک‌های مورد نظر را از کتابخانه سیمولینک وارد پنجره مدل‌سازی کرده و با استفاده از سیگنال آنها را به هم ارتباط دهید.



شکل (۴-۲۴) مدل دینامیکی سیستم

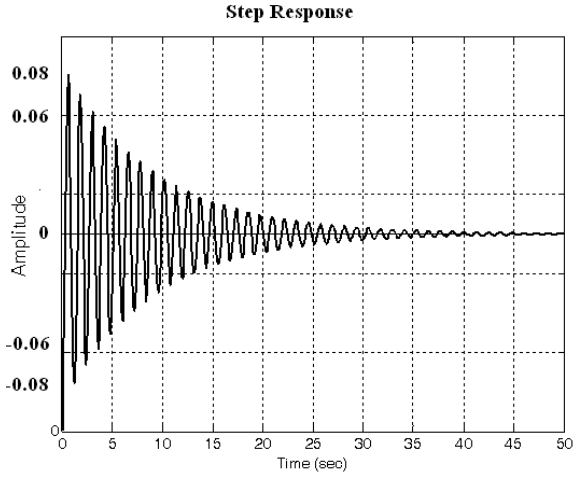
بعد از انجام مرحله مدل‌سازی سیستم، باید پاسخ پله سیستم را به‌دست آوریم. برای انجام این کار تمام بلوک های کشیده شده در شکل قبل را انتخاب کنید و با کلیک راست کردن بر روی صفحه و انتخاب subsystem بلوک ها را در داخل یک زیر سیستم قرار دهید. بعد از اینکه مدل دینامیکی سیستم را شبیه سازی کردیم باید پاسخ سیستم حلقه باز را بررسی کنیم. بنابراین در ورودی سیستم یک ورودی پله ناشی از کنترلر که مقدار نهایی آن 1 بوده و یک ورودی پله ناشی از اغتشاشات وارده از طرف جاده به سیستم که مقدار نهایی آن نیز برابر 0.1 است، قرار دهید. در خروجی نیز یک Scope بگذارید.



سپس در محیط MATLAB مقادیر زیر را وارد کنید و برنامه را اجرا کنید.

```
m1=2500;m2=320;k1=80000;k2=500000;b1 = 350;b2 = 15020;
```

خروجی برنامه به صورت زیر است:



شکل (۴-۲۵) پاسخ سیستم به ورودی ثابت کنترلر

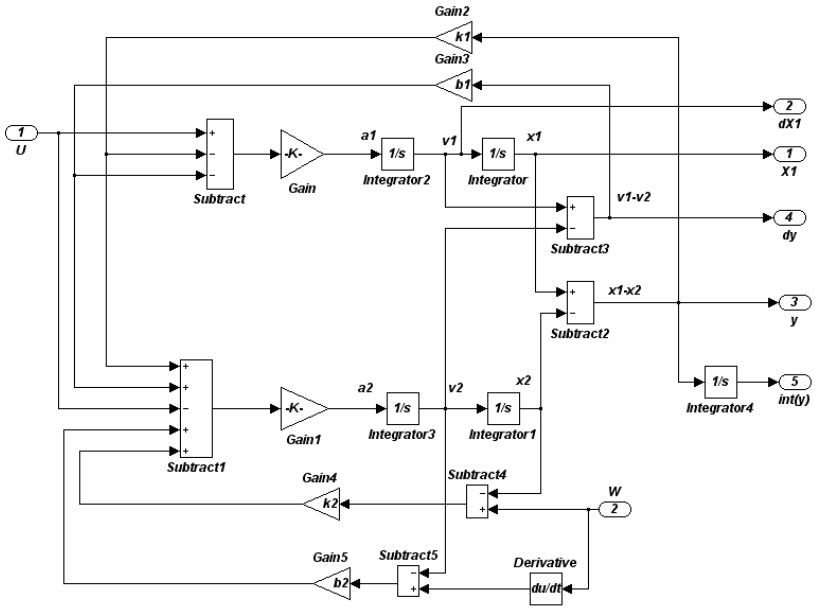
۴-۷-۱ پیاده سازی کنترلر با فیدبک تمام متغیر های حالت

همانطور که در بخش فضای حالت توضیح داده شد، در این مثال برای حذف خطای حالت ماندگار نیاز داریم تا یک حالت دیگر به سیستم اضافه کنیم. $\int X_1 - X_2 = \int y_1$

چون در واقعیت، خودرو به نقطه تعادل صفر می رسد، پس متغیر های حالت جدید به صورت زیر خواهند بود:

$X_1, X_2, y_1, y_2, \int y_1$

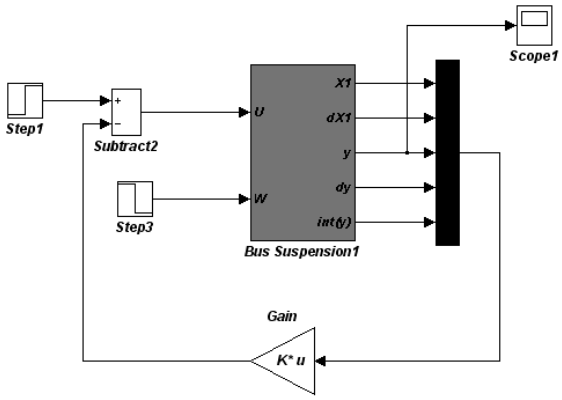
پس بلوک مربوط به سیستم تعلیق را باز کرده، و تغییرات زیر را اعمال می کنیم:



شکل (۴-۲۶) مدل دینامیکی سیستم

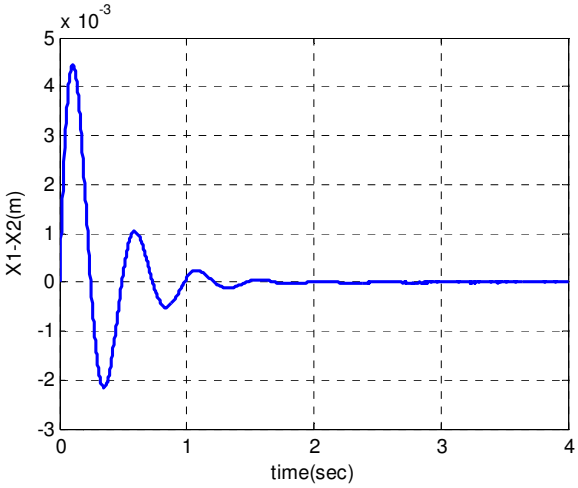
پس در این حالت بلوک Bus Suspension، دو ورودی و پنج خروجی دارد. در خروجی سیستم یک Mux قرار دهید تا برداری از خروجی‌ها را داشته باشید. سپس بلوک بهره را در خروجی آن قرار دهید. مقدار ماتریس بهره K را به صورت زیر در محیط MATLAB وارد کنید:

```
K= [ 0 2.3e6 5e8 0 8e6 ];
```



شکل (۴-۲۷) مدل سیستم کنترلی در سیمولینک

بعد از اجرای برنامه در مدت زمان 4 ثانیه، خروجی زیر برای سیستم تحت کنترل حاصل می شود،که نشان می-دهد سیستم زمان نشستی حدود 2 ثانیه داشته، و مقدار حداکثر جهش کمتر از 4.3mm است.



شکل (۴-۲۸) سیستم با کنترلر فیدبک حالت

پروژه پنجم

سیستم کنترل و تعادل پاندول وارونه

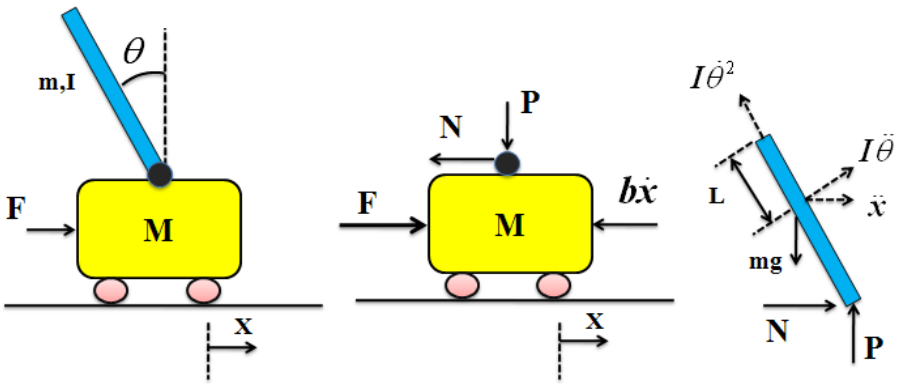
اهداف این پروژه

- مدل‌سازی
- طراحی کنترلر PID
- رسم مکان هندسی ریشه‌ها (Root Locus)
- پاسخ فرکانسی
- طراحی فضای حالت
- طراحی کنترلر دیجیتالی
- طراحی کنترلر با فیدبک کلیه متغیرهای حالت در محیط سیمولینک

۵ – ۱ مدل‌سازی

۵-۱-۱ مدل سازی سیستم براساس اصول فیزیکی و استخراج معادلات سیستم

مطابق شکل (۵-۱) یک پاندول معکوس به جرم m و به طول l متر روی ارابه‌ای به جرم M قرار دارد. این ارابه در جهت افقی حرکت می‌کند. نیروی کنترل F باعث حرکت جرم M ودر نتیجه باعث تعادل پاندول معکوس می‌گردد. در این سیستم فرض کنید اصطکاک با سرعت رابطه مستقیم دارد. در شکل زیر نمودار آزاد سیستم رسم شده است.



شکل (۵- ۱) نمودار آزاد پاندول معکوس

برای این سیستم فرض می‌کنیم:

جرم ارابه $M=0.5\ Kg$

جرم پاندول معکوس $m=0.5Kg$

اصطکاک ارابه $b=0.1N.s/m$

طول پاندول معکوس از مرکز جرم تا محور دوران $L=0.3m$

اینرسی پاندول $I=0.006Kg.m^2$

نیروی اعمالی به ارابه :F

موقعیت ارابه: x

زاویه پاندول نسبت به محور قائم :θ

۵-۱-۱-۱ روش اول مدل‌سازی براساس قانون دوم نیوتن

قانون دوم نیوتن را برای ارابه و پاندول در راستای افقی می‌نویسیم:

$$\begin{cases}\sum F_x = M\ddot{x}_M \Rightarrow F - N - b\dot{x} = M\ddot{x} \quad (I)\\ \sum F_x = m\ddot{x}_m \Rightarrow N = m\ddot{x} + ml\ddot{\theta}.Cos\theta - ml\dot{\theta}^2 Cos\theta \quad (II)\end{cases}$$
$$(I),(II) \Rightarrow (M+m)\ddot{x} + b\dot{x} + ml\ddot{\theta}Cos\theta - ml\dot{\theta}^2 Sin\theta = F \qquad (۱-۵)$$

معادله حرکت را در جهت عمود بر پاندول می‌نویسیم و سپس متغیرهای P,N را حذف می‌کنیم.

$$\begin{cases}\sum F_N = m\ddot{x}_N \Rightarrow PSin\theta + NCos\theta - mgSin\theta = ml\ddot{\theta} + m\ddot{x}Cos\theta \quad (i)\\ \sum M = I\ddot{\theta} \Rightarrow -PlSin\theta - NlCos\theta = I\ddot{\theta} \Rightarrow PSin\theta + NCos\theta = -\frac{I\ddot{\theta}}{l} \quad (ii)\end{cases}$$
$$(i),(ii) \Rightarrow (I+ml^2)\ddot{\theta} + mglSin\theta = -ml\ddot{x}Cos\theta \qquad (۲-۵)$$

از آنجا که نرم افزار MATLAB با سیستم‌های خطی کار می‌کند باید این سیستم غیرخطی را حول و حوش $\theta = \pi$ خطی سازی کنیم. بنابراین در این معادلات $Sin(\theta) \simeq 0\ , Cos(\theta) \simeq -1$. بعد از خطی‌سازی به معادلات حرکت زیر خواهیم رسید:

$$\begin{cases}(I+ml^2)\ddot{\theta} - mgl\theta = ml\ddot{x}\\ (M+m)\ddot{x} + b\dot{x} + ml\ddot{\theta} = F\end{cases} \qquad (۳-۵)$$

۵-۱-۱-۲ روش دوم مدل‌سازی توسط معادله لاگرانژ

متغیرهای تعمیم یافته که حرکت سیستم را به طور کامل توصیف می‌کنند عبارتند از :x,θ

انرژی جنبشی کل سیستم برابر است با:

$$T = \frac{1}{2}M\dot{x}^2 + \frac{1}{2}m\left[\left\{\frac{d}{dt}(x + \frac{l}{2}Sin\theta)\right\}^2 + \left\{\frac{d}{dt}(\frac{l}{2}Cos\theta)\right\}^2\right]$$
$$= \frac{1}{2}M\dot{x}^2 + \frac{1}{2}m[(\dot{x} + \frac{l}{2}\dot{\theta}Cos\theta)^2 + (-\frac{l}{2}\dot{\theta}Sin\theta)^2] \qquad (۴-۵)$$

از طرفی انرژی پتانسیل تنها در اثر تغییر ارتفاع در جرم پاندول ایجاد می‌شود. اگر سطح انرژی پتانسیل ارابه را U_o در نظر بگیریم انرژی پتانسیل مجموعه عبارت است از:

$$U = U_0 + mg\frac{l}{2}Cos\theta \qquad (۵-۵)$$

از این‌رو تابع لاگرانژ سیستم عبارت است از:

$$L = T - U = \frac{1}{2}M\dot{x}^2 + \frac{1}{2}m[(\dot{x} + \frac{l}{2}\dot{\theta}Cos\theta)^2 + (-\frac{l}{2}\dot{\theta}Sin\theta)^2] - U_0 - mg\frac{l}{2}Cos\theta \qquad (۶-۵)$$

نیروی تعمیم یافته در جهت x عبارت است از F ولی به علت عدم تحریک مستقل پاندول، نیروی تعمیم یافته‌ای در جهت θ وجود نخواهد داشت. حال معادله لاگرانژ را در امتداد x,θ می‌نویسیم:

$$\begin{aligned}
&(I): \frac{d}{dt}(\frac{\partial L}{\partial \dot{x}}) - \frac{\partial L}{\partial x} = F \\
&\Rightarrow (M+m)\ddot{x} + b\dot{x} + ml\ddot{\theta}Cos\theta - ml\dot{\theta}^2 Sin\theta = F \\
&(II): \frac{d}{dt}(\frac{\partial L}{\partial \dot{\theta}}) - \frac{\partial L}{\partial \theta} = 0 \\
&\Rightarrow (I + ml^2)\ddot{\theta} - mglSin\theta = ml\ddot{x}Cos\theta
\end{aligned} \qquad (۷-۵)$$

با محاسبه ماتریس‌های جاکوبین در نقطه کاری، معادلات خطی شده به صورت زیر در خواهند آمد:

$$LTI\ System: \begin{cases} (I + ml^2)\ddot{\theta} - mgl\theta = ml\ddot{x} \\ (M+m)\ddot{x} + b\dot{x} + ml\ddot{\theta} = F \end{cases} \qquad (۸-۵)$$

۵-۱-۲ تابع تبدیل سیستم

برای به‌دست آوردن تابع تبدیل سیستم نیاز داریم از معادله مدل سازی (۵-۸) تبدیل لاپلاس بگیریم. توجه کنید زمانی که در حال پیدا کردن تابع تبدیل سیستم هستیم شرایط اولیه را برابر صفر در نظر می‌گیریم.

$$\begin{cases} (I + ml^2)\theta(s).s^2 - mgl\theta(s) = mlX(s).s^2 \\ (M+m)X(s).s^2 + bX(s).s + ml\theta(s).s^2 = F(s) \end{cases} \qquad (۹-۵)$$

به‌دلیل اینکه خروجی مطلوب ما زاویه پاندول می‌باشد، از رابطه اول مقدار $X(s)$ را به‌دست می‌آوریم و در رابطه دوم جایگزین می‌کنیم.

$$\begin{aligned}
&X(s) = \frac{-mgl + (I + ml^2).s^2}{mls^2}\theta(s) \Rightarrow X(s) = -\frac{g}{s^2} + \frac{(I+ml^2)}{ml}\theta(s) \\
&(M+m)(-\frac{g}{s^2} + \frac{(I+ml^2)}{ml})\theta(s).s^2 + b(-\frac{g}{s^2} + \frac{(I+ml^2)}{ml})\theta(s).s + ml\theta(s).s^2 = F(s)
\end{aligned} \qquad (۱۰-۵)$$

با مرتب کردن معادله بالا داریم:

$$\frac{\theta(s)}{F(s)}=\frac{\frac{ml}{q}s^2}{s^4+\frac{b(I+ml^2)}{q}s^3-\frac{(M+m)mgl}{q}s^2-\frac{bmgl}{q}s} \qquad (۱۱-۵)$$

که در آن:

$$q=[(M+m)(I+ml^2)-(ml)^2] \qquad (۱۲-۵)$$

با توجه به تابع تبدیل می‌بینیم که یک صفر و قطب در مبدا مختصات قرار گرفته است که با هم ساده می‌شوند:

$$\frac{\theta(s)}{F(s)}=\frac{\frac{ml}{q}s}{s^3+\frac{b(I+ml^2)}{q}s^2-\frac{(M+m)mgl}{q}s-\frac{bmgl}{q}} \qquad (۱۳-۵)$$

۵-۱-۳ مدل فضای حالت سیستم

معادلات خطی شده را می‌توان به فرم فضای حالت نوشت.

$$\begin{bmatrix}\dot{x}\\ \ddot{x}\\ \dot{\theta}\\ \ddot{\theta}\end{bmatrix}=\begin{bmatrix}0 & 1 & 0 & 0\\ 0 & \frac{-b(I+ml^2)}{I(M+m)+Mml^2} & \frac{m^2gl^2}{I(M+m)+Mml^2} & 0\\ 0 & 0 & 0 & 1\\ 0 & \frac{-mbl}{I(M+m)+Mml^2} & \frac{mgl(M+m)}{I(M+m)+Mml^2} & 0\end{bmatrix}\begin{bmatrix}x\\ \dot{x}\\ \theta\\ \dot{\theta}\end{bmatrix}+\begin{bmatrix}0\\ \frac{(I+ml^2)}{I(M+m)+Mml^2}\\ 0\\ \frac{ml}{I(M+m)+Mml^2}\end{bmatrix}F \qquad (۱۴-۵)$$

$$y=\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 0 & 1 & 0\end{bmatrix}\begin{bmatrix}x\\ \dot{x}\\ \theta\\ \dot{\theta}\end{bmatrix}+\begin{bmatrix}0\\ 0\end{bmatrix}F$$

موقعیت ارابه و زاویه پاندول، خروجی سیستم می‌باشند. بنابراین ماتریس C یک ماتریس 2×4 می‌باشد. در مدل تابع تبدیل ما سیستم‌های SISO را بررسی می‌کنیم، ولی اگر سیستم به فرم فضای حالت نوشته شود می‌توانیم سیستم‌های چند ورودی و چند خروجی را نیز بررسی نمائیم.

۵-۱-۴ قیدهای حاکم بر طراحی

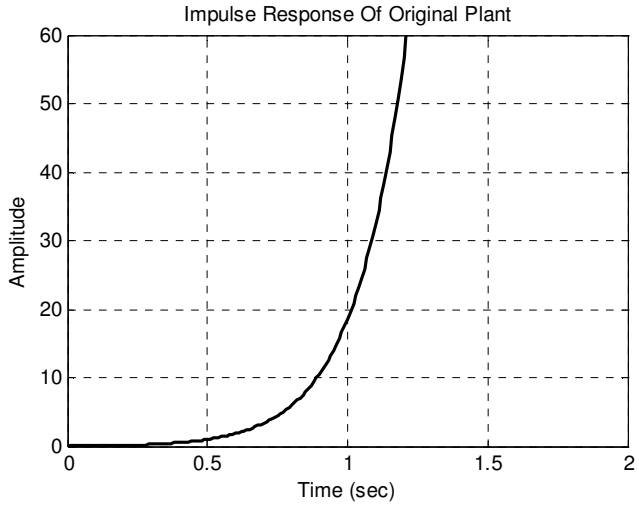
قدم بعدی در مدل سازی سیستم این است که بدانیم این سیستم قرار است چه شرایطی داشته باشد، یعنی قیدهای حاکم بر سیستم باید مشخص شوند.

- در اینجا فرض براین است که سیستم از حالت تعادل شروع به حرکت می‌کند.
- نیروی ضربه‌ای برابر 1N است.
- پاندول و ارابه باید در زمان کمتر از 5 ثانیه به وضعیت عمودی بر گردند $Settling\ time < 5\,\text{sec}$.
- پاندول نباید بیشتر از 0.35 رادیان (20deg) از وضعیت عمودی دور شود $Overshoot < 20\,\text{deg}$.
- زمان خیز برای این پاندول باید کمتر از 0.5sec باشد.

۵-۱-۵ پاسخ سیستم حلقه باز

حال بهتر است اجازه دهید پاسخ سیستم حلقه باز را به ورودی پله بررسی کنیم. برای این کار باید صورت و مخرج تابع تبدیل را به صورت برداری در محیط MATLAB وارد کنیم. دستورات زیر را در یک m-file جدید بنویسید:

```
%figure 1 Transfer function
M=0.5;m=0.2;b=0.1;i=0.006;g=9.81;l=0.3;
q=[(M+m)*(i+m*l^2)-(m*l)^2];
num=[m*l/q];
den=[1 b*(i+m*l^2)/q  -m*g*l*(M+m)/q  -m*g*l*b/q];
t=0:0.01:2;
figure(1)
impulse(num,den,t)
axis([0 2 0 60]);
```

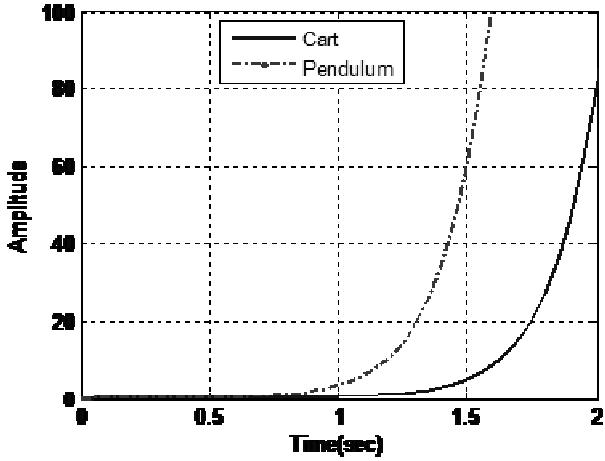


شکل (۵-۲) پاسخ پله سیستم حلقه باز

از روی پاسخ کاملاً مشخص است که سیستم در حالت حلقه باز ناپایدار است و پاندول دراثر ضربه وارده می‌افتد. MATLAB ماتریس‌های A,B,C,D معادلات حالت را به شما می‌دهد. برای انجام این کار دستورات زیر را به ادامه m-file اضافه کنید تا پاسخ سیستم (موقعیت ارابه و زاویه پاندول) را به ورودی پله 0.2m که به ارابه اعمال می‌شود مشاهده کنید.

```
%figure (2) State space
p = i*(M+m)+M*m*l^2; %denominator for the A and B matricies
A = [0      1              0           0;
     0 -(i+m*l^2)*b/p   (m^2*g*l^2)/p   0;
     0      0              0           1;
     0 -(m*l*b)/p        m*g*l*(M+m)/p  0]
```

```
B = [     0;
     (i+m*l^2)/p;
          0;
        m*l/p]
C = [1 0 0 0;
     0 0 1 0]
D = [0;
     0]
T=0:0.05:10;
U=0.2*ones(size(T));
[Y,X]=lsim(A,B,C,D,U,T);
figure(2)
plot(T,Y)
axis([0 2 0 100])
legend('Cart','Pendulum')
```



شکل (۵-۳) پاسخ پله سیستم حلقه باز

توجه: حل‌های ارائه شده در قسمت کنترلرهای PID و مکان هندسی ریشه‌ها برای مساله پاندول معکوس قابل پیاده سازی نیستند. زمانی که ما این مساله را در قالب SISO قرار می‌دهیم، در آن صورت چشم‌پوشی از موقعیت ارابه، پاندول می‌تواند در موقعیت عمودی قرار بگیرد. (ارابه حرکت بدون شتاب دارد یعنی با سرعت ثابت حرکت می‌کند)

۵-۲ طراحی کنترلر PID

همان‌طور که در بخش اول بررسی شد تابع تبدیل سیستم به صورت زیر به‌دست می‌آید:

$$\frac{\theta(s)}{F(s)} = \frac{\frac{ml}{q}s}{s^3 + \frac{b(I+ml^2)}{q}s^2 - \frac{(M+m)mgl}{q}s - \frac{bmgl}{q}} \qquad (۱۵-۵)$$

که در آن:

$$q = [(M + m)(I + ml^2) - (ml)^2]$$

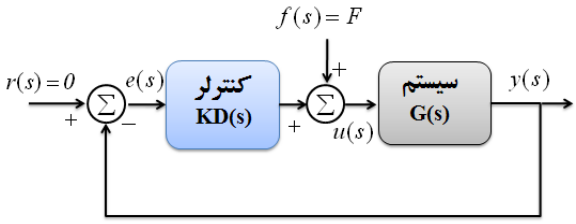
و همان‌طور که قبلاً گفته شده است، هدف ما این است که زمان نشست کمتر از 5 ثانیه بوده و پاندول بیشتر از $0.05\,rad$ نسبت به محور عمودی جابه‌جا نشود. برای به‌دست آوردن تابع تبدیل سیستم در محیط نرم‌افزار دستورات زیر را وارد کنید. توجه کنید که مقادیر صورت و مخرج را به صورت برداری وارد نمائید.

```
M=0.5;m=0.2;b=0.1;i=0.006;g=9.81;l=0.3;
q = (M+m)*(i+m*l^2)-(m*l)^2; %simplifies input
num = [m*l/q  0]
den = [1  b*(i+m*l^2)/q  -(M+m)*m*g*l/q  -b*m*g*l/q]
```

ماتریس صورت و مخرج به صورت زیر به‌دست می‌آید:

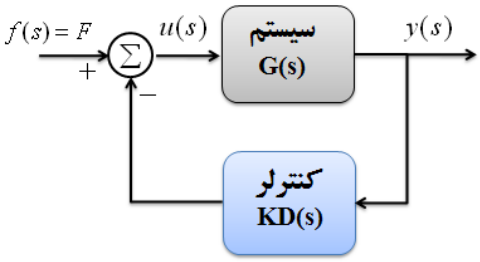
```
num =
         4.5455         0
den =
         1.0000    0.1818  -31.1818   -4.4545
```

کنترل این سیستم تا حدودی با مسائل کنترلی که قبلاً با آنها روبه‌رو بودیم متفاوت می‌باشد. از آنجایی که ما می‌خواهیم موقعیت پاندول را کنترل کنیم باید بعد از اعمال یک اغتشاش اولیه به پاندول، دوباره به وضعیت عمودی برگردد. توجه کنید که در اینجا سیگنال مرجعی که قرار است آن را دنبال کنیم برابر صفر است. نیروی اعمال شده به گاری می‌تواند به صورت یک ضربه در نظر گرفته شود. شماتیک این سیستم کنترلی به صورت زیر می‌باشد:



شکل (۵-۴) بلوک نمودار سیستم کنترلی

بلوک نمودار بالا را می‌توان به صورت زیر مرتب کرد:



شکل (۵-۵) بلوک نمودار سیستم کنترلی با آرایش جدید

تابع تبدیل سیستم حلقه بسته به صورت زیر به‌دست می‌آید:

$$\frac{y(s)}{F(s)}=\frac{G(s)}{1+KD(s).G(s)}=\frac{\frac{num}{den}}{1+\frac{K.(numPID).(num)}{(denPID).(den)}}=\frac{num.(denPID)}{(denPID).(den)+K.(numPID).(num)} \quad (۱۶-۵)$$

برای پیدا کردن این تابع تبدیل دستورات زیر را به ادامه m-file اضافه کنید:

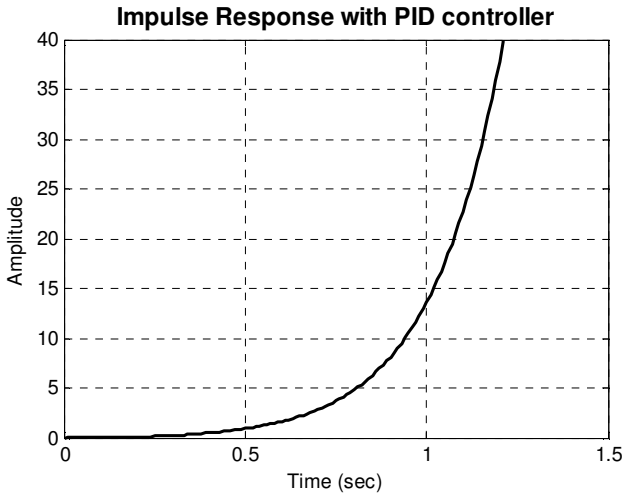
```
kd = 1;k = 1; ki = 1;
numPID = [kd k ki];
denPID = [1 0];
numc = conv(num,denPID)
denc = polyadd(conv(denPID,den),conv(numPID,num))
```

برای استفاده از تابع Polyadd باید در یک m-file جدید دستورات زیر را وارد کرده، آن را ذخیره کنید:

```
function[poly]=polyadd(poly1,poly2)
%polyadd(poly1,poly2) adds two polynominals possibly of
uneven length
if length(poly1)<length(poly2)
  short=poly1;
  long=poly2;
else
  short=poly2;
  long=poly1;
end
mz=length(long)-length(short);
if mz>0
  poly=[zeros(1,mz),short]+long;
else
  poly=long+short;
end
```

حال پاسخ سیستم حلقه بسته را بررسی می‌کنیم:

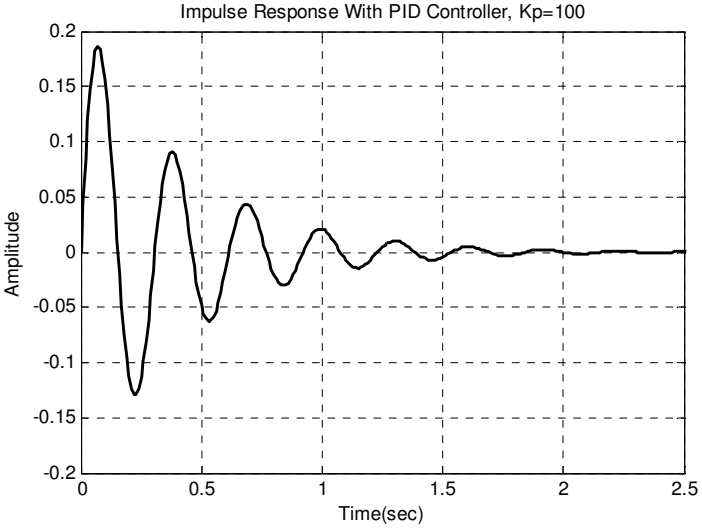
```
t=0:0.01:5;
impulse(numc,denc,t)
axis([0 1.5 0 40])
```



شکل (۵-۶) پاسخ سیستم حلقه بسته با کنترلر P

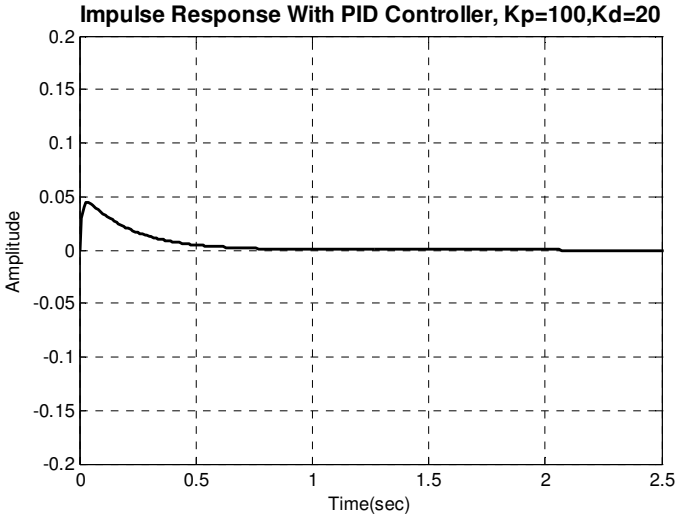
این سیستم پایدار نیست. حال بهره تناسبی را افزایش دهید مثلاً $k_p = 100$ را در نظر بگیرید و محور را به صورت زیر تغییر دهید:

```
axis([0 2.5 -.2 .2])
```



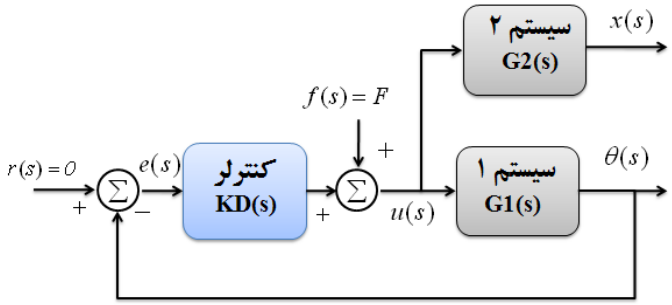
شکل (۵-۷) پاسخ سیستم حلقه بسته با کنترلر P با بهره تناسبی ۱۰۰

زمان نشست 2 ثانیه مناسب است و خطای حالت ماندگار نیز صفر است ولی مقدار جهش زیاد است و باید آن را کاهش دهیم. برای حل این مشکل، مقدار بهره Kd را افزایش داده و مقدار بهره را برابر 20 در نظر می‌گیریم تا نتیجه بهتری حاصل شود.



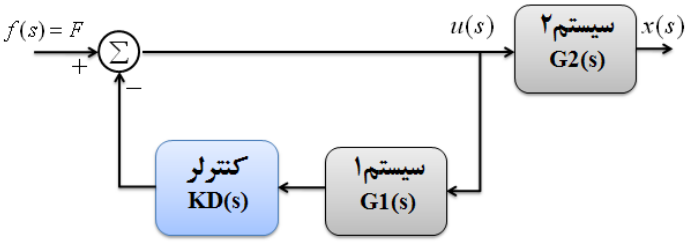
شکل (۵-۸) پاسخ سیستم حلقه بسته با کنترلر PD

حال باید ببینیم چه اتفاقی سر موقعیت ارابه می‌افتد. بلوک نمودار (۵-۵)کامل نمی‌باشد. باید بررسی کنیم وقتی زاویه پاندول کنترل می‌شود ارابه به چه موقعیتی می‌رود:



شکل(۵-۹) بلوک نمودار سیستم کنترلی همراه با در نظرگرفتن موقعیت گاری

بلوک نمودار بالا را می‌توان به صورت زیر مرتب کرد:



شکل(۵-۱۰) بلوک نمودار سیستم کنترلی همراه با در نظرگرفتن موقعیت گاری با آرایش جدید

تابع تبدیل سیستم حلقه بسته برای کنترل موقعیت گاری در اثر نیروی ضربه‌ای با یک کنترلر PID به صورت زیر می‌باشد:

$$X(s)=\frac{G_2(s)}{1+K.D(s).G_1(s)}.F(s)=\frac{\frac{num2}{den2}}{1+\frac{K.(numPID).(num1)}{(denPID).(den1)}}.F(s) \quad (۱۷-۵)$$

از آنجایی که معادله مشخصه برای هر دو خروجی یکسان است ($den2=den1$) پس داریم:

$$X(s)=\frac{G_2(s)}{1+K.D(s).G_1(s)}.F(s)=\frac{(num2).(denPID)}{(denPID).(den1)+K.(numPID).(num1)}.F(s) \quad (۱۸-۵)$$

تاکنون تابع تبدیل سیستم کلی را داشته ولی نیاز داریم تابع تبدیل موقعیت پاندول را نیز به‌دست آوریم. پس با لاپلاس‌گیری از رابطه (۵-۳) تابع تبدیل که به صورت زیر است، به‌دست می‌آید:

$$\frac{X(s)}{F(s)}=\frac{\frac{I+ml^2}{q}s^2-\frac{mgl}{q}}{s^3+\frac{b(I+ml^2)}{q}s^2-\frac{(M+m)mgl}{q}s-\frac{bmgl}{q}} \quad (۱۹-۵)$$

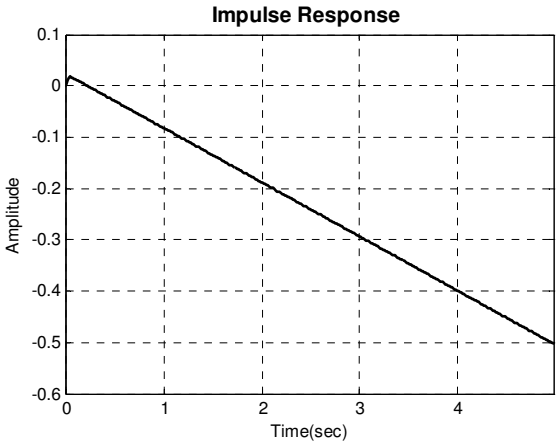
که در آن:

$$q=[(M+m)(I+ml^2)-(ml)^2] \quad (۲۰-۵)$$

برای رسم پاسخ سیستم حلقه بسته دستورات زیر را در m-file جدید بنویسید:

```
M=0.5;m=0.2;b=0.1;i=0.006;g=9.81;l=0.3;
q = (M+m)*(i+m*l^2)-(m*l)^2;   %simplifies input
num1 = [m*l/q  0  0];
den1 = [1  b*(i+m*l^2)/q  -(M+m)*m*g*l/q  -b*m*g*l/q  0];

num2 = [(i+m*l^2)/q  0  -m*g*l/q];
den2 = den1

kd = 20;k = 100; ki = 1;
numPID = [kd k ki];
denPID = [1 0];
numc = conv(num2,denPID);
denc = polyadd(conv(denPID,den2),conv(numPID,num1));
t=0:0.01:5;
impulse(numc,denc,t)
```

با اجرای برنامه به پاسخ زیر می‌رسیم:

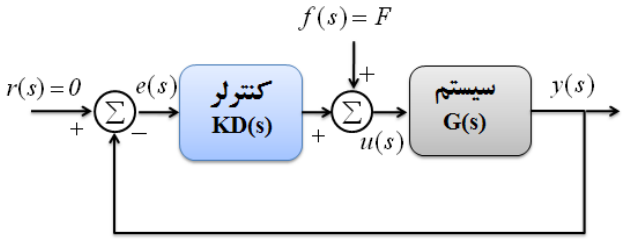


شکل (۵-۱۱) موقعیت ارابه در اثر نیروی ضربه‌ای وارده

همانطور که از پاسخ بر می‌آید گاری با یک سرعت ثابت در جهت عکس حرکت می‌کند. هرچند کنترلر PID برای پایدارسازی زاویه پاندول مناسب است ولی موقعیت گاری تثبیت شده نیست.

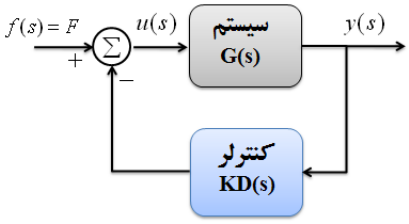
۵ -۳رسم مکان هندسی ریشه‌ها (Root Locus)

کنترل این سیستم تا حدودی متفاوت با دیگر سیستم‌های کنترلی است. از آنجایی که باید پاندول بعد از اعمال یک اغتشاش ورودی دوباره به حالت اول بازگردد، پس سیگنال مرجع که باید دنبال شود، سیگنال صفر است. نیرویی که به ارابه اعمال می‌شود می‌تواند به عنوان یک اغتشاش ضربه‌ای در نظرگرفته شود. شماتیک این سیستم کنترلی به صورت زیر می‌باشد.



شکل (۵-۱۲) بلوک نمودار سیستم کنترلی

با مرتب کردن این بلوک نمودار داریم:



شکل (۵-۱۳) بلوک نمودار سیستم کنترلی با آرایش جدید

و تابع تبدیل حلقه بسته به صورت زیر به‌دست می‌آید:

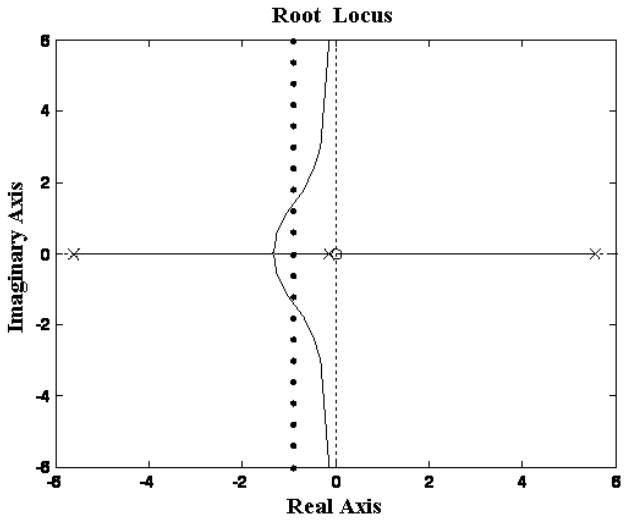
$$\frac{y(s)}{F(s)}=\frac{G(s)}{1+KD(s).G(s)}=\frac{\frac{num}{den}}{1+\frac{K.(numLead).(numLag)(num)}{(denLead).(denLag)(den)}}$$

$$=\frac{num.(denLead).(denLag)}{(denLead).(denLag).(den)+K.(numLead).(numLag).(num)} \qquad (۲۱-۵)$$

۵-۳-۱ طراحی مکان هندسی ریشه‌ها

در قدم اول مکان هندسی ریشه‌های سیستم کنترلی بدون جبران کننده، یعنی خود سیستم به تنهایی را در نظر می‌گیریم. می‌دانیم که زمان نشست باید کمتر از 5 ثانیه باشد، بنابراین مقدار $\zeta\omega=\frac{4.6}{5}=0.92$ خواهد بود. دستورات زیر را در یک m-file وارد کنید تا مکان هندسی ریشه‌های سیستم به‌دست آید:

```
M=0.5;m=0.2;b=0.1;i=0.006;g=9.81;l=0.3;
q = (M+m)*(i+m*l^2)-(m*l)^2;    %simplifies input
num = [m*l/q  0]
den = [1  b*(i+m*l^2)/q  -(M+m)*m*g*l/q  -b*m*g*l/q]
rlocus(num,den)
sigrid(0.92)
axis([-6 6 -6 6])
```

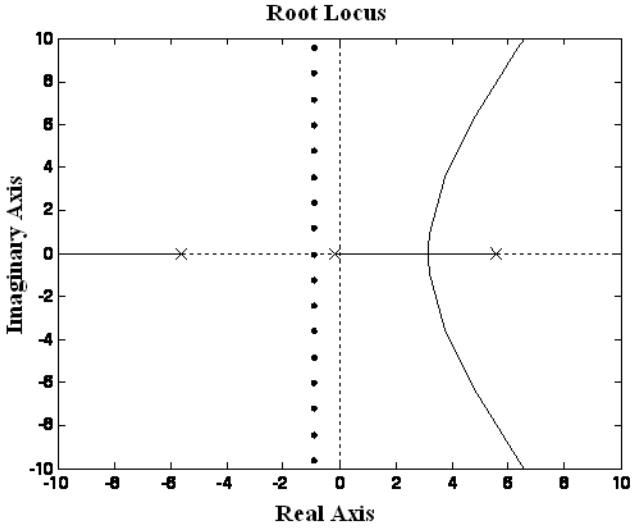


شکل (۵-۱۴) مکان هندسی ریشه‌ها

همان‌طور که در شکل مشخص است یکی از قطب‌های تابع تبدیل حلقه بسته در سمت راست محور موهومی قرار گرفته است. این به معنای ناپایداری سیستم حلقه بسته است. برای حل این مشکل در ابتدا باید یک قطب

در مبدا اضافه کنیم تا اثر صفر و قطب در مبدا خنثی شود. ثانیاً باید چندین صفر وقطب در سمت راست محور قرار بگیرند تا مکان هندسی ریشه‌ها را به سمت چپ بکشانند تا طراحی تمام شود. پس دستورات زیر را در ادامه m-file اضافه کنید.

```
p1 = 0;
dentemp = [1 p1];
num2 = num;
den2 = conv(den, dentemp);
rlocus(num2,den2)
sigrid(0.92)
axis([-10 10 -10 10])
```



شکل (۵-۱۵) مکان هندسی ریشه‌ها همراه با خنثی سازی صفر مبدا

حال قصد داریم که شاخه‌های مکان هندسی ریشه‌ها را به سمت راست بکشیم. پس دستورات زیر را به ادامه m-file اضافه کنید.

```
roots(num2)
roots(den2)
```

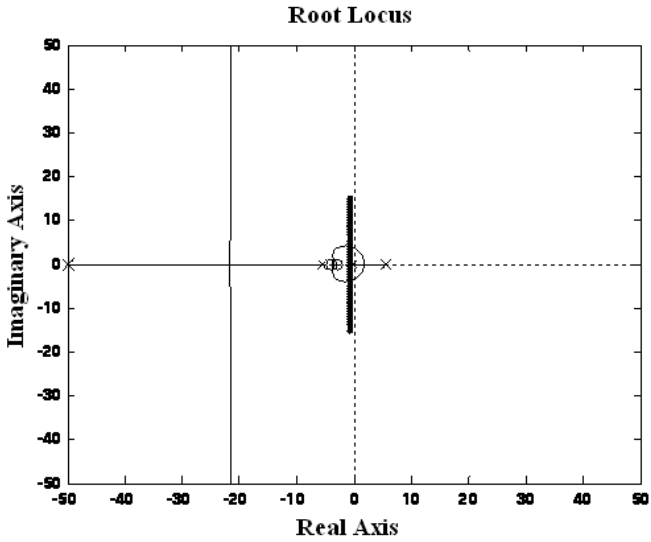
ملاحظه می‌کنید که چهار قطب و فقط یک صفر وجود دارد. پس سه مجانب خواهیم داشت. یکی در راستای محور حقیقی در جهت منفی و دو تای دیگر که با زاویه 120 درجه نسبت به هم قرار گرفته اند. در مدل فضای حالت یک صفر و قطب اضافی خواهیم داشت. باید تعداد مجانب‌ها را از سه به دو کاهش دهیم به گونه‌ای که تعداد صفرها نسبت به قطب‌ها یکی بیشتر اضافه شود.

۵-۳-۲ کنترلر Lead-Lag

برای حل این مساله یک قطب در فاصله دور از محور موهومی و به دور از صفرها و قطب‌ها در سمت چپ قرار می‌دهیم و دو تا صفر اضافه شده را نزدیک محور موهومی قرار می‌دهیم. پس قطبی که در صفر قرار داده بودیم و یکی از صفرها نقش جبران ساز Lag و قطبی که در فاصله دور قرار گرفته و صفر دیگر نقش جبران ساز Lead را بازی خواهد کرد. دستورات زیر را در یک M-file وارد کنید:

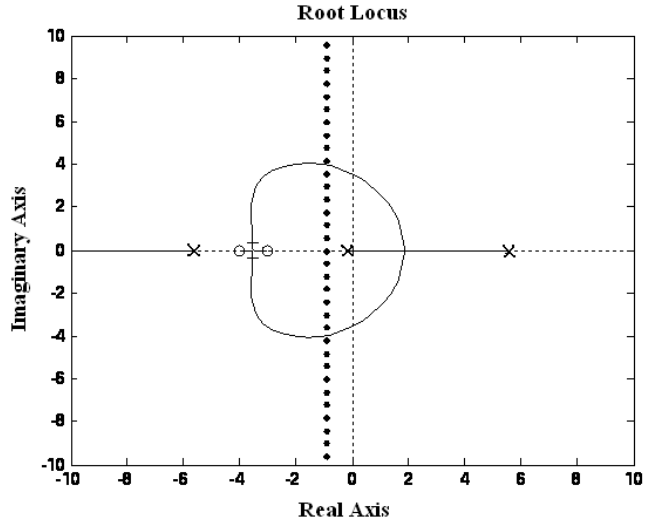
```
M=0.5;m=0.2;b=0.1;i=0.006;g=9.81;l=0.3;
q = (M+m)*(i+m*l^2)-(m*l)^2;   %simplifies input

num = [m*l/q  0];
den = [1  b*(i+m*l^2)/q  -(M+m)*m*g*l/q  -b*m*g*l/q];
z1 = 3;
p1 = 0;
z2 = 4;
p2 = 50;

numlag = [1 z1];
denlag = [1 p1];
numlead = [1 z2];
denlead = [1 p2];

num3 = conv(conv(num, numlead), numlag);
den3 = conv(conv(den, denlead), denlag);

rlocus(num3,den3)
sigrid(0.92)
axis([-50 50 -50 50])
figure
rlocus(num3,den3)
sigrid(0.92)
axis([-10 10 -10 10])
[k,poles]=rlocfind(num3,den3)
figure

numc = conv(conv(num,denlead),denlag);
denc = polyadd(k*num3,den3);
impulse(numc,denc)
axis([0 2 -0.05 0.05])
```

توجه کنید که محل صفرها و قطب‌ها با سعی و خطا به‌دست می‌آیند. تنها چیزی که باید در ذهنمان باشد این است که یکی از قطب‌ها در صفر و دیگری در فاصله دور باشد و صفرها نیز باید کوچک باشند. با اجرای برنامه نمودار (۵-۱۶) حاصل می‌شود:



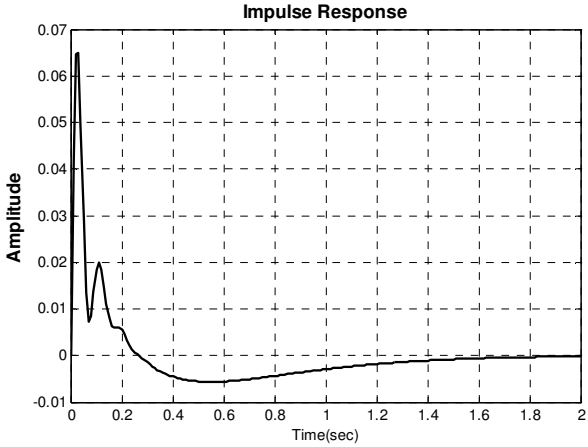
شکل (۵-۱۶) مکان هندسی ریشه‌ها همراه با کنترلر Lead-Lag

شکل (۵-۱۷) در حوالی مرجع بزرگنمایی بیشتری دارد و مکان صفرها و قطب‌ها را دقیق‌تر نمایش می‌دهد.



شکل (۵-۱۷) مکان هندسی ریشه‌ها همراه با کنترلر Lead-Lag

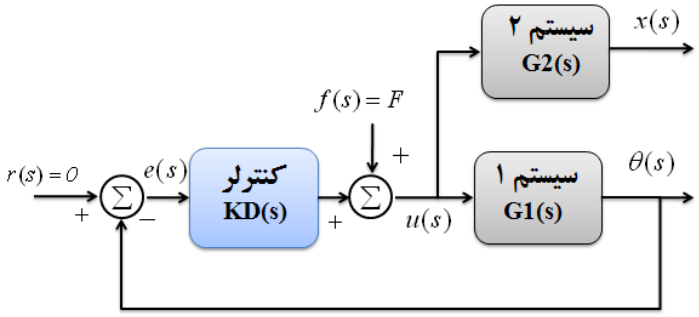
وقتی برنامه اجرا شود پیغامی ظاهر خواهد شد که از شما یک نقطه پرسیده می‌شود. پس نقاط نشان داده شده در شکل (۵-۱۷) را انتخاب کنید تا پاسخ سیستم حلقه بسته به اغشاشات ضربه‌ای وارده به‌دست آید.



شکل (۵-۱۸) پاسخ سیستم حلقه بسته به اغشاشات ضربه‌ای وارده

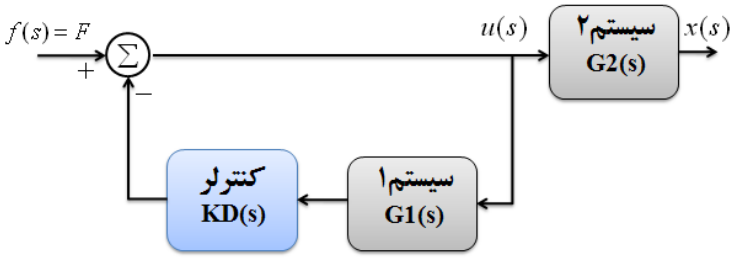
چه اتفاقی سر موقعیت ارابه می‌افتد؟

حال باید ببینیم چه اتفاقی سر موقعیت ارابه می‌افتد. همانطورکه در قسمت کنترلر PID گفته شد، بلوک نمودار (۵-۱۳) کامل نمی‌باشد و علاقمندیم بدانیم وقتی زاویه پاندول کنترل می‌شود ارابه به چه موقعیتی می‌رسد. پس بلوک نمودار سیستم واقعی را به صورت زیر رسم می‌کنیم.



شکل (۵-۱۹) بلوک نمودار سیستم کنترلی همراه با در نظر گرفتن موقعیت گاری

بلوک نمودار بالا را می‌توان به صورت زیرمرتب کرد:



شکل (۵-۲۰) بلوک نمودار سیستم کنترلی همراه با در نظر گرفتن موقعیت گاری با آرایش جدید

تابع تبدیل سیستم برای موقعیت ارابه به صورت زیر به‌دست می‌آید.

$$X(s)=\frac{G_2(s)}{1+K.D(s).G_1(s)}.F(s)=\frac{\frac{num2}{den2}}{1+\frac{K.(numLead).(numLag).(num1)}{(denLead).(denLag).(den1)}}.F(s) \quad (۲۲-۵)$$

$$X(s)=\frac{G_2(s)}{1+K.D(s).G_1(s)}.F(s)$$
$$=\frac{(den1)(num2).(denLead).(denLag)}{(denLead).(denLag).(den1)(den2)+K.(numLead).(numLag).(num1)(den2)}.F(s) \quad (۲۳-۵)$$

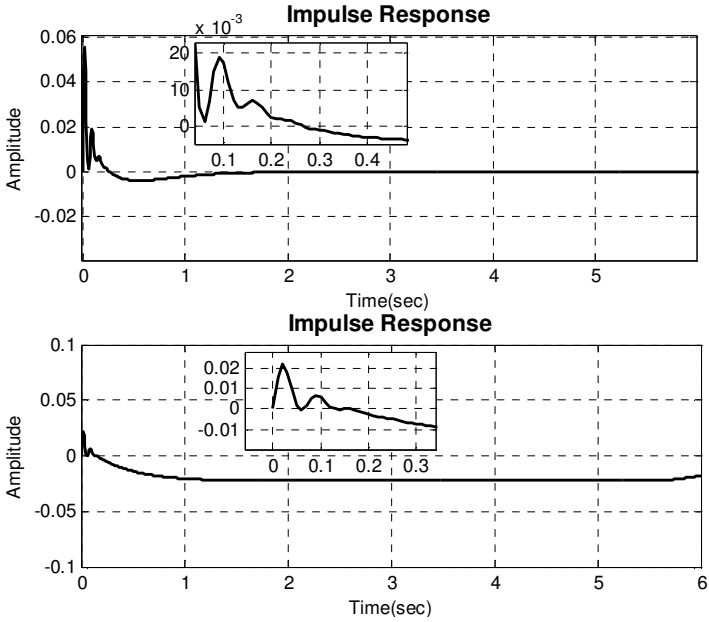
از آنجایی که معادله مشخصه برای هر دو خروجی یکسان است ($den2 = den1$) پس داریم:

$$\frac{X(s)}{F(s)}=\frac{(num2).(denLead).(denLag)}{(denLead).(denLag).(den1)+K.(numLead).(numLag).(num1)} \quad (۲۴-۵)$$

دستورات زیر را در یک M-file جدید وارد کنید:

```
M=0.5;m=0.2;b=0.1;i=0.006;g=9.81;l=0.3;
q = (M+m)*(i+m*l^2)-(m*l)^2;   %simplifies input
num1 = [m*l/q  0  0];
den1 = [1  b*(i+m*l^2)/q  -(M+m)*m*g*l/q  -b*m*g*l/q  0];
num2 = [(i+m*l^2)/q  0  -m*g*l/q];
den2 = den1
z1 = 3;p1 = 0;
z2 = 4;p2 = 50;
numlag = [1 z1];
denlag = [1 p1];
numlead = [1 z2];
denlead = [1 p2];
num3 = conv(conv(num1, numlead), numlag);
den3 = conv(conv(den1, denlead), denlag);
subplot(1,1,1);rlocus(num3,den3)
axis([-10 10 -10 10])
[k,poles]=rlocfind(num3,den3)
numc = conv(conv(num1,denlead),denlag);
denc = polyadd(k*num3,den3);
t=0:0.01:6;
subplot(2,1,1);
impulse(numc,denc,t)
axis([0 6 -0.05 0.05])
num4 = conv(num2,den3);
den4 = polyadd(conv(den1,den3),k*conv(den1,num3));
subplot(2,1,2);
impulse(num4,den4,t)
axis([0 6 -0.1 0.1])
```

بعد از اجرای برنامه همان نقطه قبلی را انتخاب کنید تا برای سیستم حلقه بسته، پاسخ‌های زیر حاصل شود:



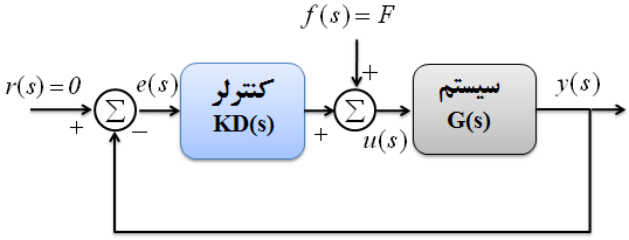
شکل (۵-۲۱) موقعیت پاندول و ارابه

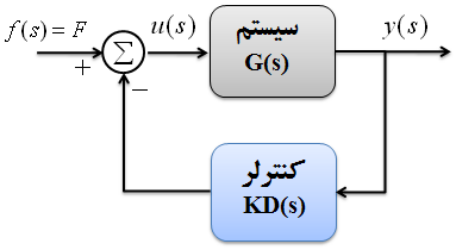
شکل بالایی موقعیت پاندول و شکل پائینی موقعیت ارابه را نشان می‌دهد. همان‌طور که شکل پائینی نشان می‌دهد ارابه برای 5 ثانیه اول پایدار است ولی بعد از 5 ثانیه به صورت ناپایدار در می‌آید. البته این امکان وجود دارد که اصطکاک موجود در چرخ‌ها که از آن چشم‌پوشی کردیم باعث پایداری ارابه گردد.

۵ –۴پاسخ فرکانسی

۵-۴-۱ پاسخ حلقه بسته بدون جبران کننده

در این قسمت مایلیم با استفاده از نمودار نایکوئیست یک کنترلر برای پاندول معکوس در برابر نیروی ضربه‌ای وارد بر ارابه طراحی کنیم (نمی‌توانیم از نمودار بود استفاده کنیم چون سیستم در حالت حلقه باز ناپایدار است). با توجه به بلوک نمودارهای زیر داریم:





شکل (۵-۲۲) بلوک نمودار کنترلی پاندول معکوس

بنابراین دستورات زیر را وارد کنید:

```
M=0.5;m=0.2;b=0.1;i=0.006;g=9.81;l=0.3;
q = (M+m)*(i+m*l^2)-(m*l)^2;   %simplifies input
num = [m*l/q  0  0]
den = [1  b*(i+m*l^2)/q  -(M+m)*m*g*l/q  -b*m*g*l/q  0]
x = roots(num)
y = roots(den)
```

```
x =
     0
     0
y =
     0
   -5.6041
    5.5651
   -0.1428
```

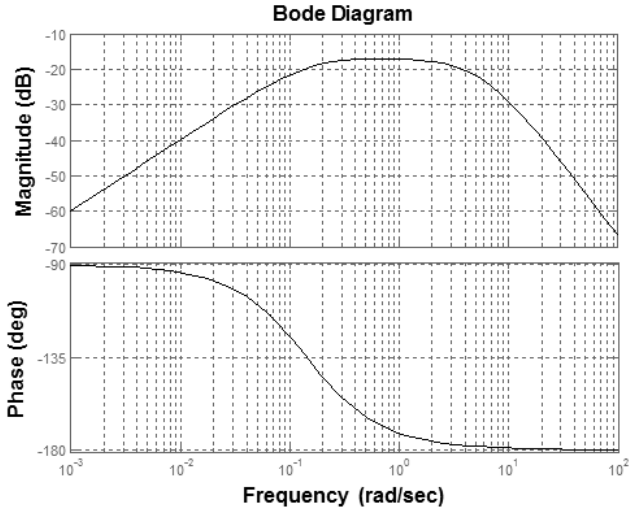
همانطور که می‌بینید در مبدا حذف صفر و قطب وجود دارد، از طرفی یک قطب مثبت در سمت راست محور موهومی وجود دارد و این به آن معنی است که نیاز به دوران خلاف جهت عقربه‌های ساعت حول 1- داریم تا سیستم حلقه بسته پایداری داشته باشیم. m-file زیر برای طراحی کنترلر بسیار مفید می‌باشد.

```
function[ ] = pend()
%define TF
num =  [4.5455          0          0];
den =[1.0000    0.1818  -31.1818   -4.4545          0];
figure(1)
%ask user for controller
numc = input('numc?.........');
denc = input('denc?.........');
k    = input('K?............');
%view compensated system bode
bode(k*conv(numc,num), conv(denc,den))
%view compensated system nyquist
figure(2)
subplot (2,1,1)
nyquist(k*conv(numc,num), conv(denc,den))
```

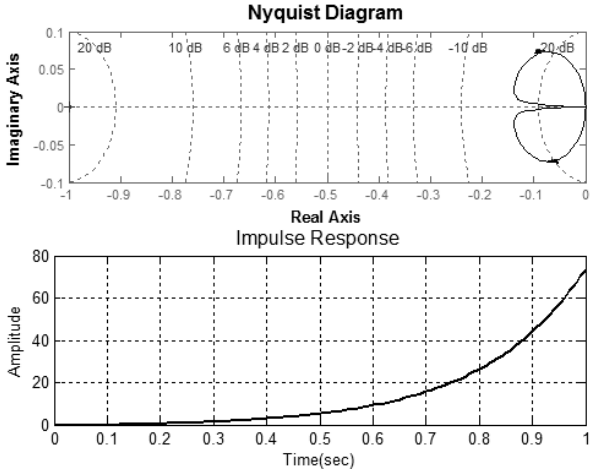
```
%view compensated CL system impulse response
subplot(2,1,2)
clnum = conv(num,denc);
temp1 = k*conv(numc,num);
temp2 = conv(denc, den);
clden = polyadd(temp1,temp2);
impulse (clnum,clden)
```

در محیط MATLAB تابع pend را فراخوانی کرده و مقادیر زیر را وارد کنید تا نمودار زیر ظاهر شود.

```
>> pend
numc?.........1
denc?.........1
K?............1
```



شکل (۵-۲۳) نمودار بود سیستم حلقه بسته بدون جبران ساز

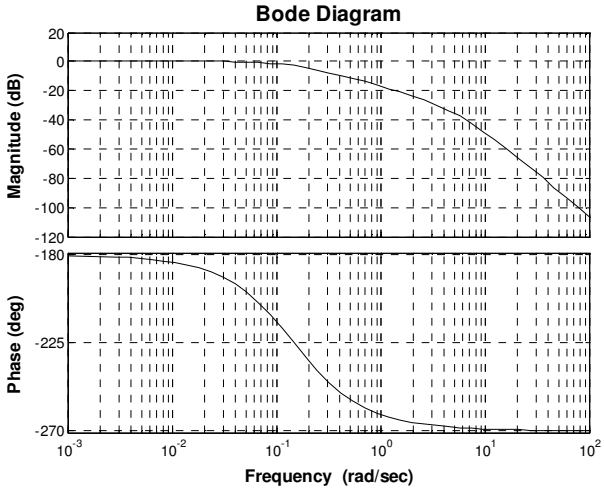


شکل (۵-۲۴) نمودار نایکوئیست و پاسخ سیستم حلقه بسته بدون جبران ساز

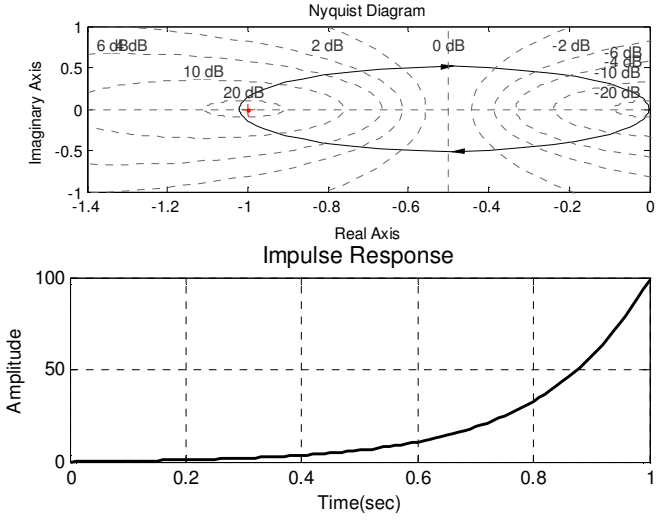
۵-۴-۲ پاسخ حلقه بسته با جبران کننده

همانطور که از شکل‌های بالا بر می‌آید سیستم حلقه بسته بدون جبران کننده ناپایدار است (منحنی حول 1- دورانی ندارد). اولین کاری که باید انجام دهیم این است که از یک انتگرال گیر برای حذف صفر استفاده کنیم. در نتیجه ما دو صفر و دو قطب در مبدا خواهیم داشت. M-file قبلی را یک بار دیگر اجرا کنید و مقادیر زیر را وارد نمائید.

```
numc?.........1
denc?.........[1 0]
K?............1
```



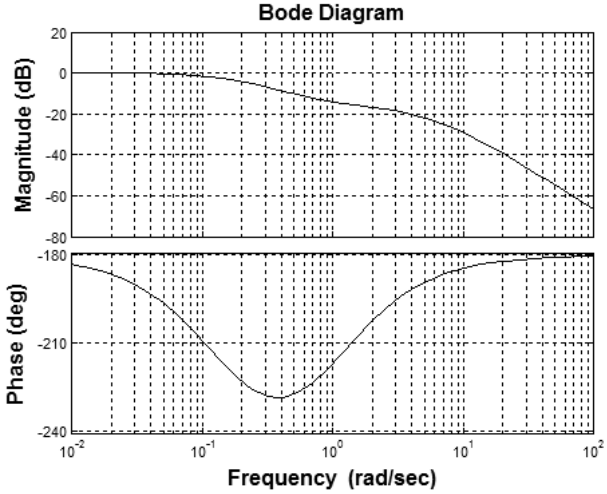
شکل (۵-۲۵): نمودار بود سیستم حلقه بسته با جبران ساز



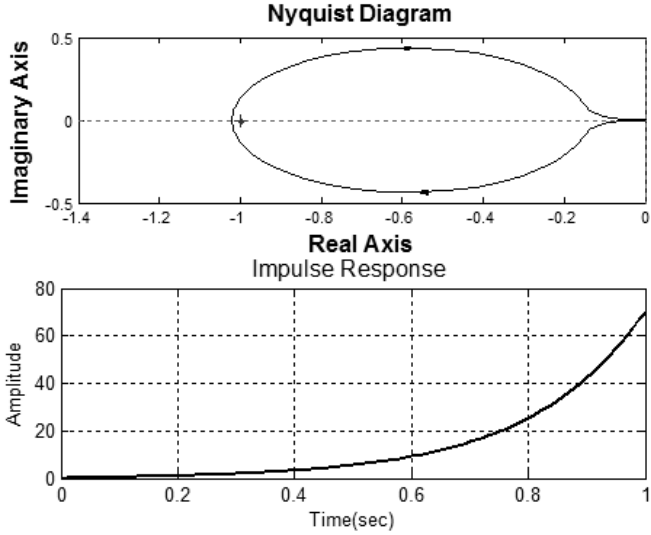
شکل (۵-۲۶) نمودار نایکوئیست و پاسخ سیستم حلقه بسته با جبران ساز

توجه کنید که نمودار نایکوئیست یک دور حول 1- در جهت عقربه‌های ساعت چرخیده است. بنابراین در حال حاضر دو قطب در سمت راست محور موهومی داریم. بنابراین نیاز است که فازی به منظور دوران منحنی در خلاف جهت عقربه‌های ساعت اضافه کنیم. این کار را با اضافه کردن یک صفر به کنترلر انجام می‌دهیم. برای شروع صفر را در 1- قرار می‌دهیم.

```
numc?.........[1 1]
denc?.........[1 0]
K?............1
```



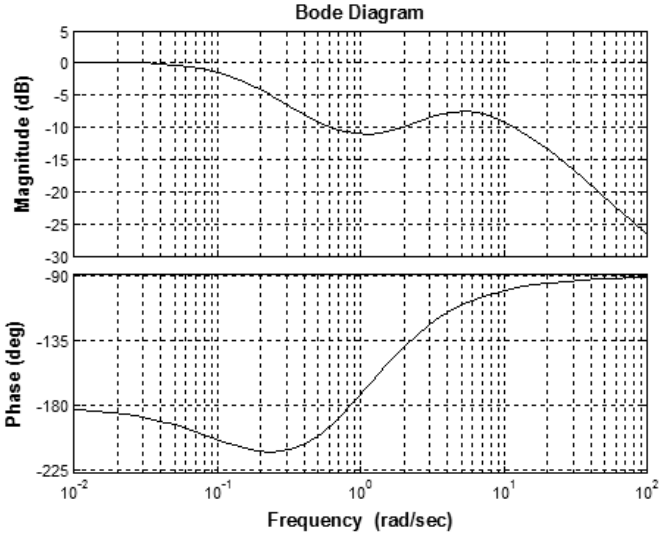
شکل (۵–۲۷) نمودار بود سیستم حلقه بسته با جبران ساز



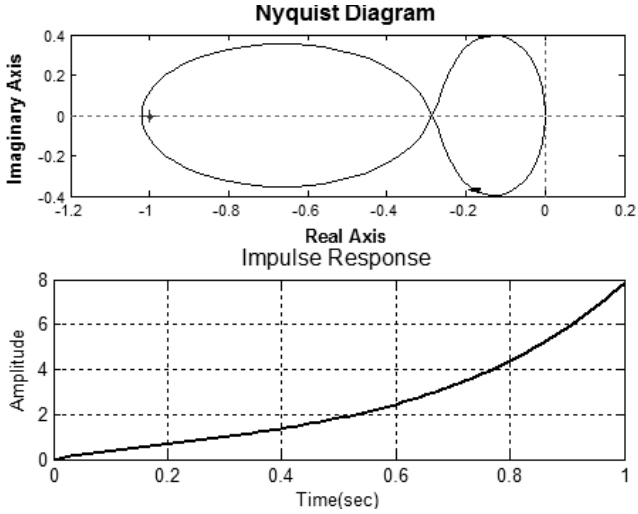
شکل (۵–۲۸) نمودار نایکوئیست و پاسخ سیستم حلقه بسته با جبران ساز

همانطور که می‌بینید این فاز اضافه شده کافی نبوده و هنوز دوران منحنی حول 1- در جهت ساعتگرد است. بنابراین نیاز داریم یک صفر دیگر اضافه کنیم. پس برنامه را یک بار دیگر اجرا کرده و مقادیر زیر را وارد کنید:

```
numc?.........conv([1 1],[1 1])
denc?.........[1 0]
K?............1
```



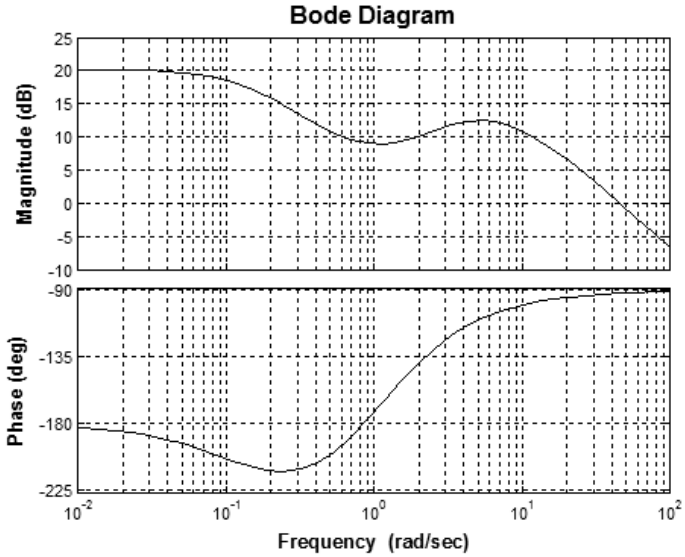
شکل (۵–۲۹) نمودار بود سیستم حلقه بسته با جبران ساز



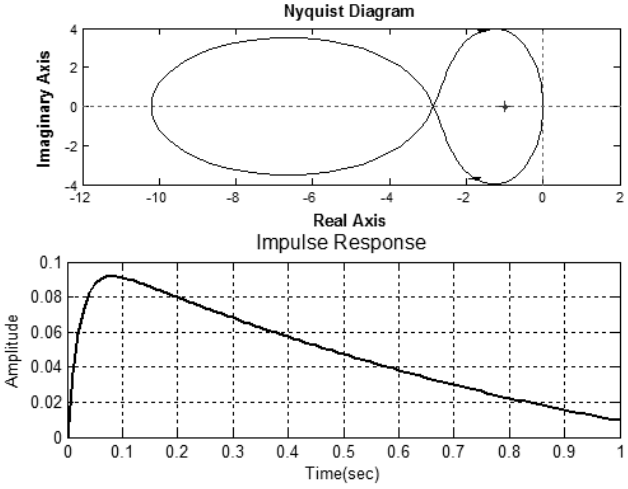
شکل (۵–۳۰) نمودار نایکوئیست و پاسخ سیستم حلقه بسته با جبران ساز

هنوز دوران منحنی حول 1- در جهت ساعتگرد است. حال اگر یک بهره به سیستم اضافه کنیم می‌توانیم با کشیدن منحنی نایکوئیست به سمت چپ، سیستم را پایدار کنیم و یک دوران خلاف جهت عقربه‌های ساعت حول 1- داشته باشیم. پس برنامه را یک بار دیگر اجرا کرده، مقادیر زیر را وارد کنید:

```
numc?.........conv([1 1],[1 1])
denc?.........[1 0]
K?............10
```



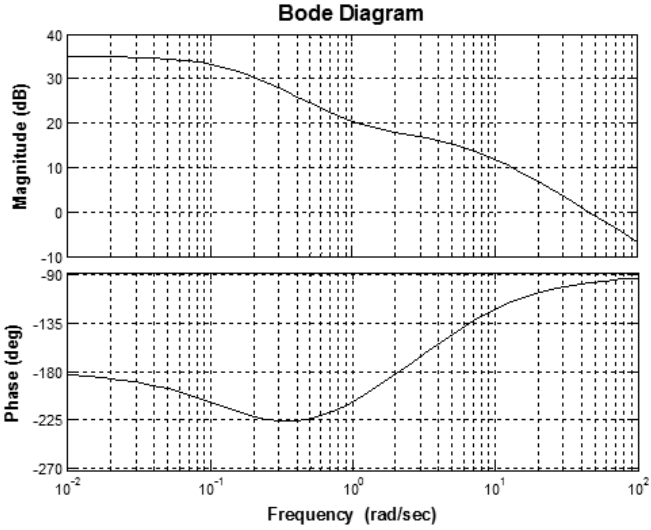
شکل (۵-۳۱) نمودار بود سیستم حلقه بسته با جبران ساز



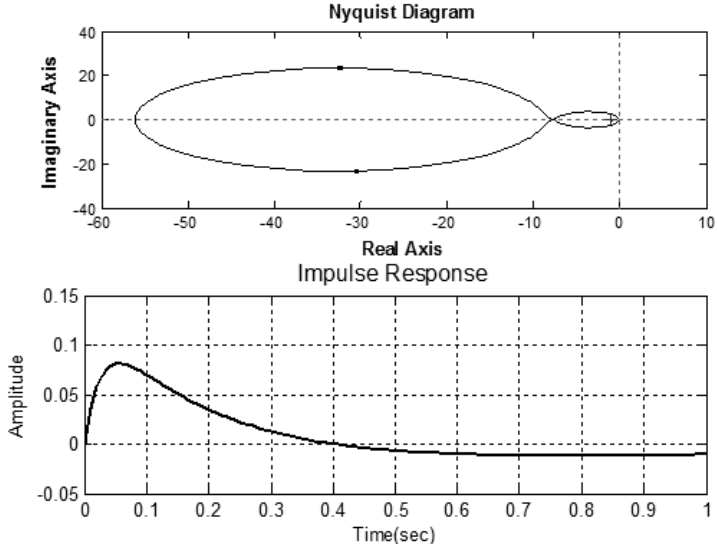
شکل (۵-۳۲) نمودار نایکوئیست و پاسخ سیستم حلقه بسته با جبران ساز

همانطور که از شکل مشخص است سیستم پایدار می‌باشد. حال می‌توانیم روی پاسخ متمرکز شویم و جواب را بهبود ببخشیم. برای این کار می‌توانیم قطب را اصلاح کنیم. توجه داشته باشید که قطب‌های کوچک نزدیک مبدا روی جواب سیستم در فرکانس‌های پائین تاثیر می‌گذارد و قطب‌های دور از مبدا روی پاسخ سیستم در فرکانس‌های بالا تاثیر می‌گذارد. با اجرای برنامه بالا مقادیر زیر را برای صفرها و قطب‌ها وارد کنید.

```
numc?.........conv([1 1.1],[1 5])
denc?.........[1 0]
K?............10
```



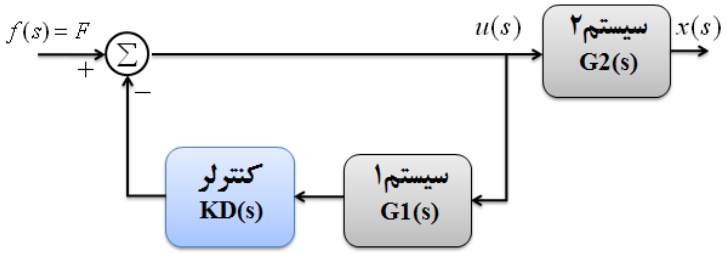
شکل (۵-۳۳) نمودار بود سیستم حلقه بسته با جبران ساز



شکل (۵-۳۴) نمودار نایکوئیست و پاسخ سیستم حلقه بسته با جبران ساز

چه اتفاقی سر موقعیت ارابه می‌افتد ؟

حال باید ببینیم چه اتفاقی سر موقعیت ارابه می‌افتد. بلوک نمودار (۵-۲۲) کامل نمی‌باشد و علاقمندیم بدانیم وقتی زاویه پاندول کنترل می‌شود ارابه به چه موقعیتی می‌رسد. پس بلوک نمودار سیستم واقعی را به صورت شکل (۵-۳۵) رسم می‌کنیم.



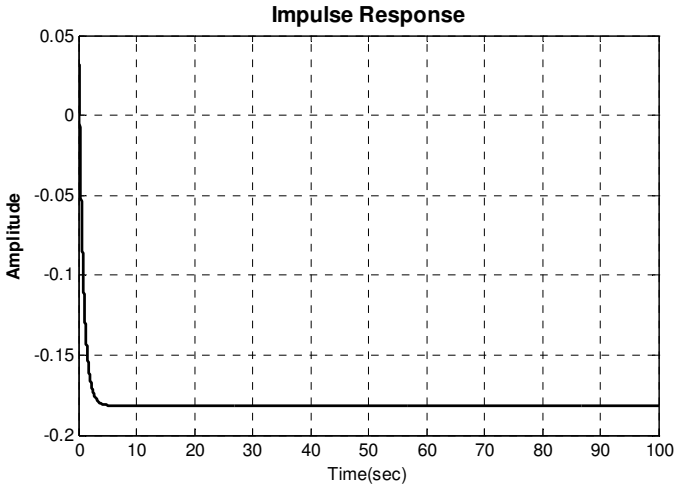
شکل (۵-۳۵) بلوک نمودار سیستم کنترلی همراه با در نظر گرفتن موقعیت گاری

تابع تبدیل سیستم برای موقعیت ارابه به صورت زیر است:

$$\frac{X(s)}{F(s)} = \frac{(num2).(denController)}{(denController).(den1) + K.(numController).(num1)}$$

دستورات زیر را در یک M-file جدید وارد کرده و آن را اجرا کنید.

```
M=0.5;m=0.2;b=0.1;i=0.006;g=9.81;l=0.3;
q = (M+m)*(i+m*l^2)-(m*l)^2;    %simplifies input
num1 = [m*l/q  0  0];
den1 = [1  b*(i+m*l^2)/q  -(M+m)*m*g*l/q  -b*m*g*l/q  0];
num2 = [(i+m*l^2)/q  0  -m*g*l/q];
den2 = den1;
k = 10;
numcontroller = conv([1 1.1],[1 5]);
dencontroller = [1 0];
numc = conv(num2,dencontroller);
denc=polyadd(conv(dencontroller,den1),k*conv(numcontroller,nu
m1));
t=0:0.01:100;
impulse(numc,denc,t)
```

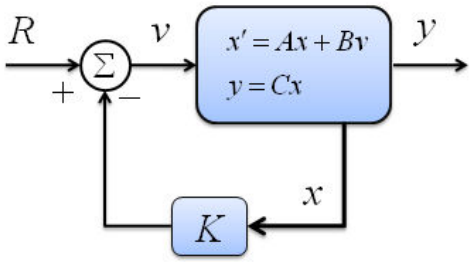


شکل (۵-۳۶) موقعیت ارابه بعد از اعمال نیروی ضربه‌ای

همان‌طور که در شکل بالا نشان داده شده است، ارابه در جهت منفی حرکت کرده، در فاصله 18cm- متوقف می‌شود. این طراحی برای کنترلر واقعی بسیار خوب کار می‌کند.

۵ –۵ طراحی فضای حالت

در این قسمت قصد داریم با استفاده از روش فیدبک، تمام متغیرهای حالت یک کنترلر مساله پاندول معکوس را طراحی کنیم. هدفمان این است که پاندول و ارابه در زمان کمتر از 5 ثانیه متوقف شوند و پاندول نباید بیشتر از 0.35 رادیان (20deg) از وضعیت عمودی دور شود و زمان خیز برای این پاندول باید کمتر از 0.5sec باشد. شماتیک این روش کنترلی در زیر آورده شده است:



شکل (۵-۳۷) کنترل به روش فیدبک کلیه متغیرهای حالت

در این مساله R نشان دهنده ورودی پله اعمالی به ارابه است. متغیرهای حالت عبارتند از موقعیت و سرعت ارابه و زاویه و سرعت زاویه‌ای پاندول. خروجی شامل موقعیت ارابه و زاویه پاندول است. می‌خواهیم بعد از اینکه ورودی پله به ارابه اعمال شد پاندول بعد از یک تغییر مکان زاویه‌ای دوباره به موقعیت عمودی برگردد و ارابه به موقعیت جدید فرمان داده شده برود.

اولین قدم در طراحی کنترلر تعیین قطب‌های سیستم‌های حلقه باز می‌باشد. دستورات زیر را در یک m-file وارد کنید.

```
M = 0.5;m = 0.2;b = 0.1;i = 0.006;g = 9.8;
l = 0.3;
p = i*(M+m)+M*m*l^2; %denominator
A = [0      1              0           0;
     0 -(i+m*l^2)*b/p (m^2*g*l^2)/p   0;
     0      0              0           1;
     0 -(m*l*b)/p     m*g*l*(M+m)/p 0];
B = [0; (i+m*l^2)/p; 0; m*l/p];
C = [1 0 0 0;
     0 0 1 0];
D = [0;0];
p = eig(A)
```

```
p =
   0
   5.5651
   -0.1428
   -5.6041
```

همانطور که از روی قطب‌ها مشخص است یک قطب 5.56 در سمت راست محور موهومی قرار گرفته است. بنابراین سیستم در حالت حلقه باز ناپایدار است.

۵-۵-۱ طراحی LQR

قدم بعدی در طراحی کنترلر این است که فرض کنید تمام متغیرهای حالت را می‌توانید اندازه گیری کرده، بردار K را پیدا کنید و از این طریق می‌توانید قانون کنترل فیدبک را بنویسید. توجه کنید که این کار را می‌توانید به چندین روش انجام دهید. اگر شما قطب‌های حلقه بسته مطلوب را بدانید، در آن صورت می‌توانید از دستور place یا acker استفاده کنید. روش دیگر این است که از روش LQR استفاده کنید که به شما یک کنترلر بهینه را می‌دهد. این روش به شما اجازه می‌دهد که ماتریس کنترل بهینه‌ای را پیدا کنید که تعادلی بین خطای سیستم و تلاش کنترلی ایجاد شود. در این روش به سه پارامتر نیاز داریم:

ماتریس R، ماتریس Q ، و ماتریس وزنی ρ. برای ساده سازی $R=\rho=1$ و $Q=C^T\times C$ را در نظر می‌گیریم. این روش به شما این امکان را می‌دهد که بتوانید هر دو خروجی، یعنی موقعیت ارابه و زاویه پاندول را کنترل کنید. دستور زیر را وارد محیط MATLAB کنید:

$Q=C^T\times C$

$ans =$

$$\begin{matrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{matrix}$$

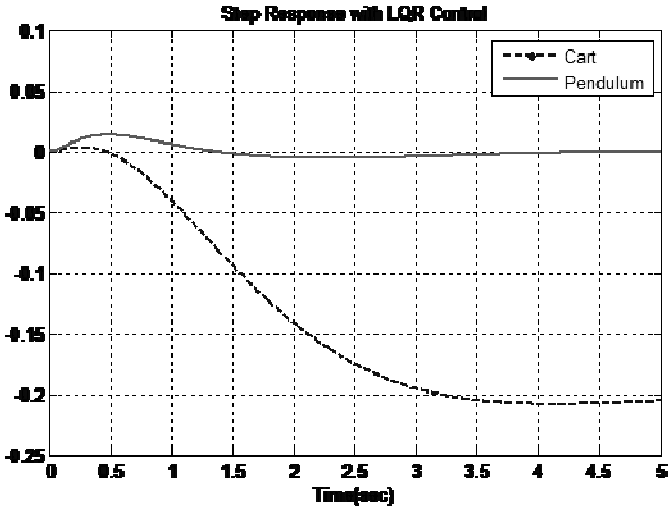
$$\begin{matrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{matrix}$$

مولفه (۱و۱) ضریب وزنی است که به موقعیت ارابه اختصاص دارد و مولفه (۳و۳) ضریب وزنی است که به زاویه پاندول اختصاص داده می‌شود. پس حالا می‌توانیم ماتریس K را به‌دست آوریم. دستورات زیر را به ادامه m-file اضافه کنید.

```
x=1;
y=1;
Q=[x 0 0 0;0 0 0 0;0 0 y 0;0 0 0 0];
R = 1;
K = lqr(A,B,Q,R)
Ac = [(A-B*K)];
Bc = [B];
Cc = [C];
Dc = [D];
T=0:0.01:5;
U=0.2*ones(size(T));
[Y,X]=lsim(Ac,Bc,Cc,Dc,U,T);
plot(T,Y)
legend('Cart','Pendulum')
```

ماتریس k برابر است با:

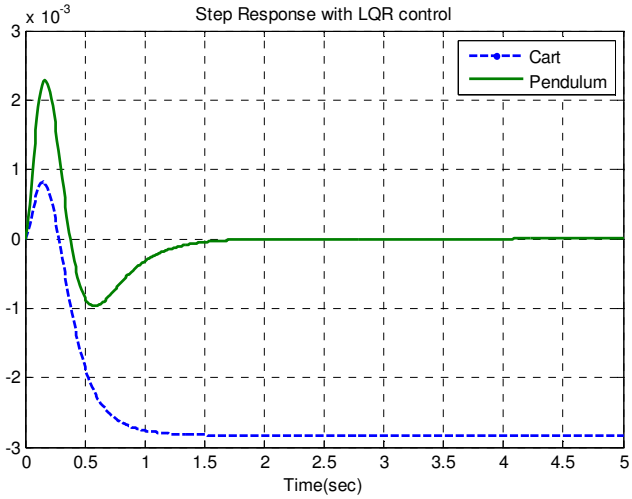
```
K =
  -1.0000   -1.6567   18.6854    3.4594
```



شکل (۵-۳۸) کنترل به روش فیدبک کلیه متغیرهای حالت با روش LQR

مقدار جهش ارابه و پاندول خوب و مناسب است ولی مقدار زمان نشست و زمان خیز باید کمتر شود. همچنین ارابه باید در جهت دیگری حرکت کند. حال ما باید روی مساله بهبود زمان نشست و زمان خیز و ثابت کردن خطای حالت ماندگار متمرکز شویم. اجازه بدهید که مقدار فاکتور وزنی را افزایش دهیم تا مقدار زمان نشست و زمان خیز کاهش پیدا کند. بنابراین به m-file برگردید و مقادیر x=5000,y=100 را تغییر دهید. برنامه را دوباره اجرا کنید.

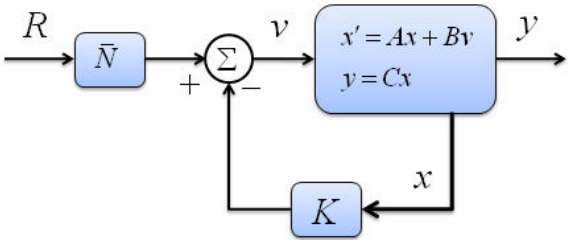
```
K =
  -70.7107  -37.8345  105.5298   20.9238
```



شکل (۵-۳۹) پاسخ پله سیستم با روش کنترلی LQR

۵-۵-۲ افزودن ورودی مرجع

در این قسمت قصد داریم از دست خطای حالت ماندگار خلاص شویم. بدون شباهت به بقیه روش‌های طراحی دیگر، در روش فیدبک کلیه متغیرهای حالت، خروجی سیستم با ورودی مرجع مقایسه نمی‌شود بلکه فیدبک همه متغیرها ضربدر بهره با ورودی مرجع مقایسه می‌گردد و یک مقدار به‌دست می‌آید. برای به‌دست آوردن خروجی دلخواه باید این مقدار مساوی صفر باشد. این کار با استفاده از ضریب پیشخور Nbar انجام می‌شود:



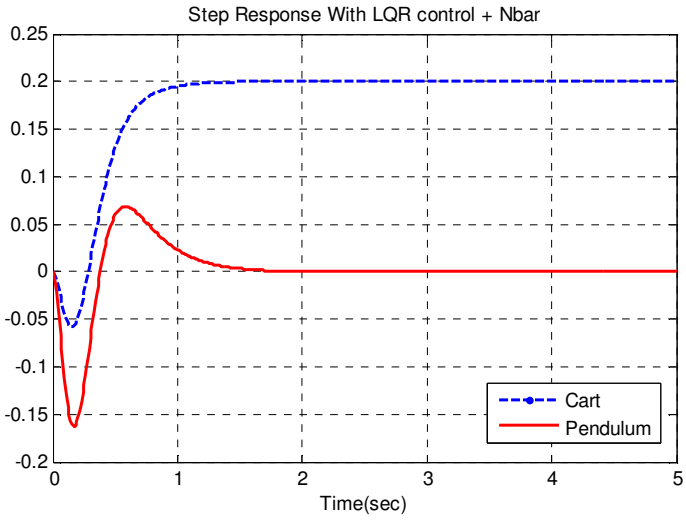
شکل (۵-۴۰) کنترل به روش فیدبک کلیه متغیرهای حالت همراه با ورودی مرجع

```
Cn=[1 0 0 0];
```

```
Nbar=rscale(A,B,Cn,0,K)
Bcn=[Nbar*B];
[Y,X]=lsim(Ac,Bcn,Cc,Dc,U,T);
plot(T,Y)
legend('Cart','Pendulum')
```

Nbar =

-70.7107

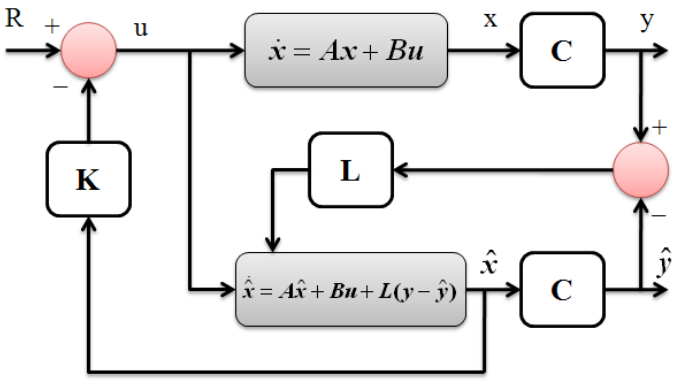


شکل (۵-۴۱) پاسخ پله سیستم با روش LQR همراه با ورودی مرجع

همان‌طور که از شکل(۵-۴۱) مشخص است تمام معیارهای طراحی ما برآورده شده است و خطای حالت ماندگار نیز به صفر می‌رسد.

۵-۵-۳ طراحی رویتگر

پاسخ به‌دست آمده مناسب است اما ما با این فرض جلو رفتیم که تمام حالت‌ها را در اختیار داریم ولی این فرض اشتباه است. برای جبران آن از یک تخمین زننده مرتبه کامل برای تخمین متغیرها استفاده می‌کنیم. شماتیک این رویتگر به صورت زیر می‌باشد.



شکل (۵-۴۲) سیستم حلقه بسته همراه با رویتگر

در ابتدا باید قطب‌های کنترلر بدون رویتگر را پیدا کنیم. بنابراین دستور p=eig(Ac) را وارد می‌کنیم. درنتیجه مقادیر زیر حاصل می‌شود.

```
p =
-8.4910 + 7.9283i
-8.4910 - 7.9283i
-4.7592 + 0.8309i
-4.7592 - 0.8309i
```

سپس نیاز داریم بهره L رویتگر را تعیین کنیم. از آنجا که می‌خواهیم دینامیک رویتگر بسیار سریعتر از خود سیستم باشد، نیاز داریم که قطب‌های رویتگر را حداقل 4-10 بار دورتر از قطب‌های غالب سیستم در نظر بگیریم.

```
P = [-40 -41 -42 -43];
L = place(A',C',P)'
```

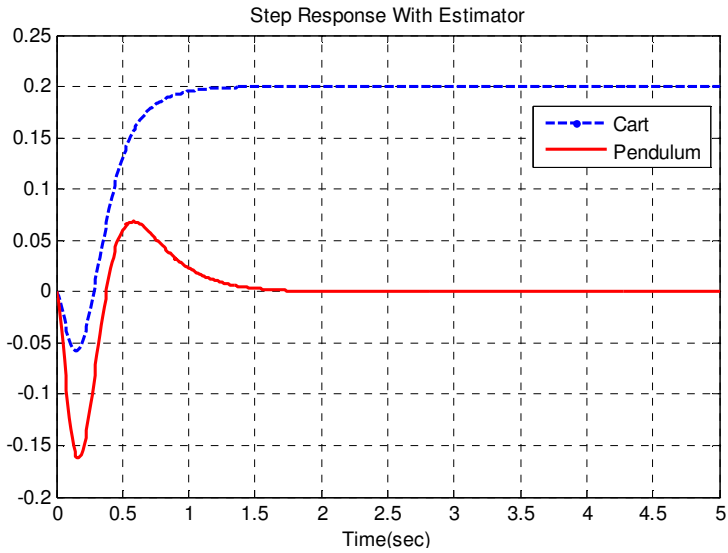
برای طراحی رویتگر از هر دو خروجی استفاده می‌کنیم. اگر فقط زاویه پاندول را به عنوان خروجی در نظر بگیریم در آن صورت سیستم رویت پذیر نخواهد بود. این کار را می‌توانید در MATLAB با محاسبه rank(obsv(A,C(2,:))) چک کنید. اگر شما فقط زاویه پاندول را اندازه‌گیری کنید نمی‌توانید دقیقاً تعیین کنید که موقعیت ارابه کجاست.

```
L =1.0e+003 *
      0.0826   -0.0010
      1.6992   -0.0402
     -0.0014    0.0832
     -0.0762    1.7604
```

خیلی مرسوم است که معادلات ترکیبی سیستم و رویتگر با استفاده از متغیرهای حالت اصلی X و خطای حالت $e = x - \hat{x}$ نوشته شود. پس از آن از فیدبک حالت $u = -K\hat{x}$ استفاده می‌کنیم. بعد از کمی محاسبات

جبری به یک معادله ترکیبی از حالت‌ها و خطای حالت‌ها به همراه فیدبک تمام متغیرهای حالت و یک رویتگر می‌رسیم. دستورات زیر را ادامه m-file وارد کنید.

```
Ace = [A-B*K              B*K;
       zeros(size(A)) (A-L*C)];
Bce = [       B*Nbar;
      zeros(size(B))];
Cce = [Cc zeros(size(Cc))];
Dce = [0;0];
T = 0:0.01:5;
U = 0.2*ones(size(T));
[Y,X] = lsim(Ace,Bce,Cce,Dce,U,T);
plot(T,Y)
legend('Cart','Pendulum')
```



شکل (۵-۴۳) پاسخ پله سیستم با استفاده از تخمین‌گر حالت

همانطور که مشاهده می‌شود جواب همانند حالت قبل است و تمام ملزومات طراحی برآورده شده است.

۵ -۶طراحی کنترلر دیجیتالی

در این قسمت قصد داریم با استفاده از روش فضای حالت یک کنترلر دیجیتالی برای مساله پاندول معکوس طراحی کنیم. هدفمان این است که پاندول و ارابه در زمان کمتر از 5 ثانیه متوقف شوند و پاندول نباید بیشتر از 0.35 رادیان (20deg) از وضعیت عمودی دور شود وزمان خیز برای این پاندول باید کمتر از 0.5sec باشد.

۵-۶-۱ فضای حالت گسسته

اولین کاری که ما باید انجام دهیم این است که مدل فضای حالت پیوسته را به فضای گسسته ببریم. پس از دستور c2dm استفاده می‌کنیم. در این قسمت به آرگومانهای زیر نیاز داریم:

ماتریس‌های A،B،C،وD، زمان نمونه برداری، و روش طراحی. زمان نمونه برداری باید کوچکتر از $1/30\times BW$ باشد. از طرفی روش ZOH را به‌عنوان روش طراحی انتخاب می‌کنیم. فرض می‌کنیم که فرکانس پهنای باند تقریباً 1rad/sec است، پس زمان نمونه برداری را 1/100 در نظر می‌گیریم. حال می‌توانیم از دستور c2dm استفاده کنیم و مدل فضای حالت گسسته را به‌دست آوریم. پس دستورات زیر را در یک m-file وارد کنید.

```
M = .5;m = 0.2;b = 0.1;i = 0.006;g = 9.8;
l = 0.3;
p = i*(M+m)+M*m*l^2; %denominator for the A and B matricies
A = [0      1              0           0;
     0 -(i+m*l^2)*b/p   (m^2*g*l^2)/p   0;
     0      0              0           1;
     0 -(m*l*b)/p        m*g*l*(M+m)/p  0];
B = [     0;
    (i+m*l^2)/p;
          0;
      m*l/p];
C = [1 0 0 0;
     0 0 1 0];
D = [0;
     0];
Ts=1/100;
[F,G,H,J]=c2dm (A,B,C,D,Ts,'zoh')
```

مدل فضای حالت گسسته به صورت زیر به‌دست می‌آید:

$$\begin{bmatrix}\dot{x}(k)\\ \ddot{x}(k)\\ \dot{\varphi}(k)\\ \ddot{\varphi}(k)\end{bmatrix}=\begin{bmatrix}1 & 0.01 & 0.0001 & 0\\ 0 & 0.998 & 0.0267 & 0.0001\\ 0 & 0 & 1.0016 & 0.01\\ 0 & -0.004 & 0.3119 & 1.0016\end{bmatrix}\begin{bmatrix}x(k-1)\\ \dot{x}(k-1)\\ \varphi(k-1)\\ \dot{\varphi}(k-1)\end{bmatrix}+\begin{bmatrix}0.0001\\ 0.0182\\ 0.0002\\ 0.0454\end{bmatrix}[u(k-1)]$$

$$y=\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 0 & 1 & 0\end{bmatrix}\begin{bmatrix}x(k-1)\\ \dot{x}(k-1)\\ \varphi(k-1)\\ \dot{\varphi}(k-1)\end{bmatrix}+\begin{bmatrix}0\\ 0\end{bmatrix}\times[u(k-1)]$$

۵-۶-۲بررسی کنترل پذیری و رویت پذیری

قدم بعدی این است که کنترل پذیری و رویت پذیری را بررسی کنیم. ماتریس کنترل پذیری به صورت زیر می‌باشد:

$$C=[G\ FG\ F^2G\ \ldots\ F^{n-1}G]$$

که رتبه ماتریس باید n باشد. ماتریس رویت پذیری نیز به صورت زیر تعریف می‌شود که رتبه آن نیز باید برابر n باشد.

$$O=[H\ HF\ HF^2\ \ldots\ HF^{n-1}]^T$$

از آنجایی که هر دو ماتریس C,O 4×4 می‌باشند پس مرتبه هر دو ماتریس باید 4 باشد.

```
co = ctrb (F,G);
ob = obsv (F,H);
Controllability = rank (co)
Observability = rank (ob)
```

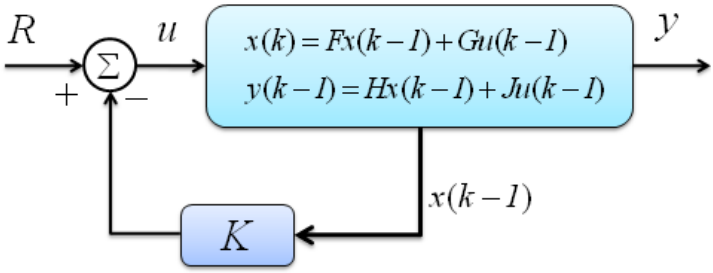
Controllability = 4

Observability = 4

چون هر دو ماتریس کنترل پذیری و رویت پذیری مرتبه کامل هستند پس سیستم گسسته هم کنترلر پذیر است و هم رویت پذیر.

۵-۶-۳طراحی کنترلر با استفاده از جایابی قطب‌ها

شماتیک روش فیدبک تمام متغیرهای حالت به صورت زیر می‌باشد:



شکل (۵-۴۴) کنترل به روش فیدبک کلیه متغیرهای حالت

که در بلوک بالا:

K : ماتریس کنترل

X : ماتریس متغیرهای حالت

U : ورودی کنترلی

R : ورودی مرجع

برای حل این مساله از روش دیگری به نام LQR استفاده می‌کنیم. این روش به شما اجازه می‌دهد تا ماتریس کنترل بهینه‌ای را پیدا کنید که تعادلی بین خطای سیستم و تلاش کنترلی ایجاد شود. در این روش به سه پارامتر نیاز داریم:

ماتریس R، ماتریس Q و ماتریس وزنی ρ.برای ساده سازی فرض می‌کنیم $R=1$ و $Q=H^T\times H$. دستور زیر را وارد محیط MATLAB کنید:

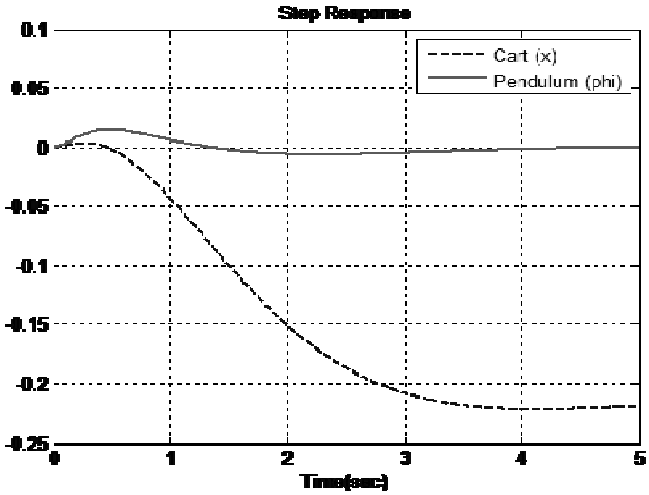
Q=H'*H

ans =

1	0	0	0
0	0	0	0
0	0	1	0
0	0	0	0

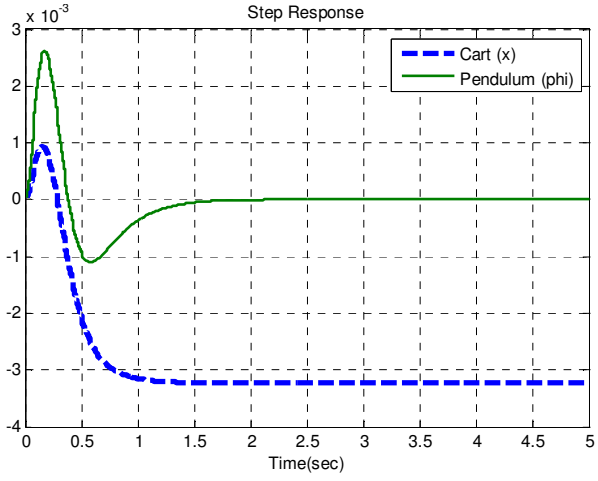
در ابتدا ماتریس وزنی را برابر $\rho = 1$ در نظر می‌گیریم. مولفه (۱و۱) ضریب وزنی است که به موقعیت ارابه و مولفه (۳و۳) ضریب وزنی است که به زاویه پاندول اختصاص داده می‌شود. این فاکتورهای وزنی در ابتدا به صورت مجزا انتخاب می‌شوند. پس اکنون می‌توانیم ماتریس K را به‌دست آوریم. دستورات زیر را به ادامه m-file اضافه کنید.

```
T=0:0.01:5;
U=0.2*ones(size(T));

F = [1.0000   0.0100   0.0001   0.0000;
          0   0.9982   0.0267   0.0001;
         0   0.0000   1.0016   0.0100;
          0  -0.0045   0.3119   1.0016];
G =  [0.0001;
      0.0182;
      0.0002;
      0.0454];
H =  [1    0    0    0;
      0    0    1    0];

J =   [0;
      0];
x=1;    % weighting factor for the cart position
y=1;    % weighting factor for the pendulum angle

Q=[x 0 0 0;
   0 0 0 0;
   0 0 y 0;
   0 0 0 0];

R = 1;
K = dlqr(F,G,Q,R)
[Y,X]=dlsim(F-G*K,G,H,J,U);
stairs(T,Y)
legend('Cart (x)','Pendulum (phi)')
```



شکل (۵-۴۵) پاسخ پله سیستم با روش LQR

مقدار جهش ارابه و پاندول خوب و مناسب است ولی مقدار زمان نشست و زمان خیز باید کمتر شود. همچنین ارابه باید در جهت دیگری حرکت کند. حال ما باید روی مساله بهبود زمان نشست و زمان خیز و ثابت کردن خطای حالت ماندگار متمرکز شویم. حال اجازه بدهید که مقدار فاکتور وزنی را افزایش دهیم تا مقدار زمان نشست و زمان خیز کاهش پیدا کند. بنابراین به m-file برگردید و مقادیرx=5000,y=100 را تغییر دهید و برنامه را دوباره اجرا کنید.

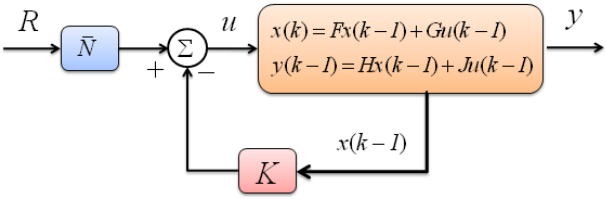


شکل (۵-۴۶) پاسخ پله سیستم با روش LQR با تغییر ضرایب وزنی

از روی نمودار مشخص است که تمام ملزومات طراحی ما برآورده شده است به‌جز خطای حالت ماندگار موقعیت ارابه x که باید یک ضریب پیشخور به ابتدای آن اضافه کنیم.

۵-۶-۴ تعریف ورودی مرجع

متمایز از بقیه روش‌های طراحی دیگر در روش فیدبک تمام متغیرهای حالت، خروجی سیستم با ورودی مرجع مقایسه نمی‌شود بلکه خروجی ضربدر بهره با ورودی مرجع مقایسه می‌گردد. برای به‌دست آوردن خروجی دلخواه باید خروجی با ورودی مرجع مساوی باشد. این کار با استفاده از ضریب پیشخور Nbar انجام می‌شود:

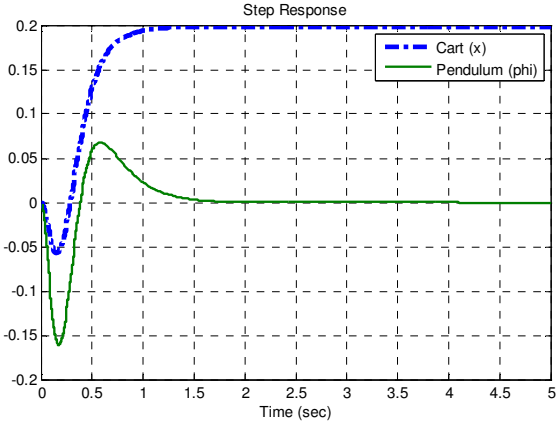


شکل (۵-۴۷) کنترل به روش فیدبک کلیه متغیرهای حالت با ورودی مرجع

پس دستورات زیر را به ادامه m-file اضافه کنید و پاسخ پله وارده به ارابه را به‌دست آورید.

```
x=5000; %weighting factor for the cart position
y=100;  %weighting factor for the pendulum angle
Q=[x 0 0 0;
   0 0 0 0;
   0 0 y 0;
   0 0 0 0];
R = 1;
K = dlqr(F,G,Q,R)
Nbar=-61.55;
[Y,X]=dlsim(F-G*K,G*Nbar,H,J,U);
stairs(T,Y)
legend('Cart (x)','Pendulum (phi)')
```

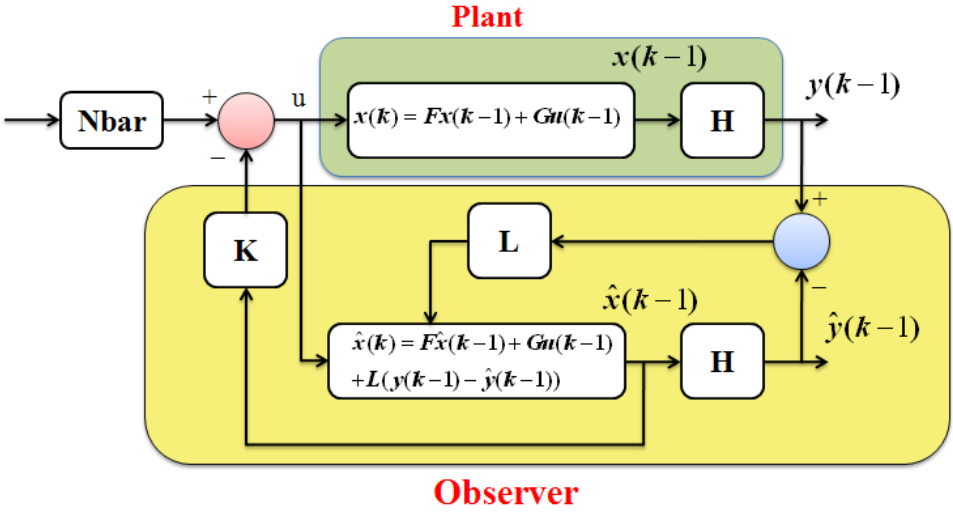
توجه: مقدار Nbar را باید با سعی و خطا پیدا کنیم. با چند بار سعی و خطا کردن به مقدار 61- می‌رسیم.



شکل (۵-۴۸) پاسخ پله سیستم با روش LQR با در نظر گرفتن ورودی مرجع

۵-۶-۵ طراحی رویتگر

پاسخ بالا تمام ملزومات طراحی ما را برآورده می‌کند. به هر حال ما فرض کردیم که تمام حالت‌ها قابل اندازه-گیری هستند. این فرض برای تمام سیستم‌ها معتبر نیست. در این قسمت ما قصد داریم از یک تکنیک برای تخمین متغیرهای حالت استفاده کنیم. این سیستم دینامیکی که برای تخمین متغیرهای حالت استفاده می‌شود رویتگر نامیده می‌شود. در این قسمت ما از یک رویتگر مرتبه کامل برای تخمین متغیرهایی حالتی که قابل اندازه گیری نیستند استفاده می‌کنیم. شماتیک رویتگر سیستم به صورت زیر می‌باشد:



شکل (۵-۴۹) سیستم حلقه بسته همراه با رویتگر دیجیتالی

برای طراحی رویتگر اول نیاز داریم که ماتریس L را به‌دست آوریم و برای پیدا کردن این ماتریس هم نیاز داریم که قطب‌های سیستم حلقه بسته بدون رویتگر یعنی $F-G\times K$ را به‌دست آوریم.

```
poles = eig (F-G*K)
poles =
0.9156 + 0.0729i
0.9156 - 0.0729i
0.9535 + 0.0079i
0.9535 - 0.0079i
```

می‌خواهیم قطب‌های رویتگر را در جایی قرار دهیم که رویتگر خیلی سریعتر از سیستم بدون رویتگر کار کند. بنابراین اجازه بدهید که قطب‌های رویتگر را خیلی دورتر در سمت چپ قرار دهیم؛ به طور مثال در [-0.3 -0.32 -0.33].

البته می‌توانیم بعداً این قطب‌ها را تغییر دهیم. از دستور place برای پیدا کردن ماتریس L استفاده می‌کنیم.

```
F = [1.0000    0.0100    0.0001    0.0000;
          0    0.9982    0.0267    0.0001;
          0    0.0000    1.0016    0.0100;
          0   -0.0045    0.3119    1.0016];
```

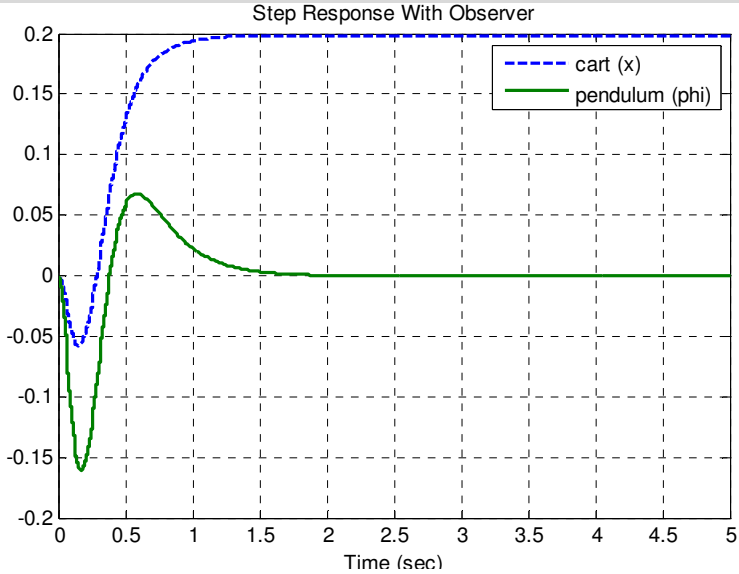
```
H =  [1    0    0    0;
      0    0    1    0];

P = [-0.3 -0.31 -0.32 -0.33];

L = place (F',H',P)'
```

```
L =
    2.6310   -0.0105
  172.8146   -1.3468
   -0.0129    2.6304
   -2.2954  173.2787
```

برای پیدا کردن پاسخ سیستم با استفاده از یک رویتگر دستورات زیر را در یک m-file جدید وارد کنید.

```
T=0:0.01:5;
U=0.2*ones(size(T));

F = [1.0000   0.0100   0.0001   0.0000;
          0   0.9982   0.0267   0.0001;
          0   0.0000   1.0016   0.0100;
          0  -0.0045   0.3119   1.0016];
G =  [0.0001;0.0182;0.0002;0.0454];
H =  [1    0    0    0;0    0    1    0];
J =   [0; 0];

x=5000; %weighting factor for the cart position
y=100;  %weighting factor for the pendulum angle
Q=[x 0 0 0;0 0 0 0;0 0 y 0;0 0 0 0];

R = 1;
K = dlqr(F,G,Q,R)
Nbar = -61.55;
L =   [2.6310    -0.0105;
      172.8146    -1.3468;
       -0.0129     2.6304;
       -2.2954   173.2787];

Fce = [F-G*K           G*K;
       zeros(size(F))  (F-L*H)];

Gce = [G*Nbar;
       zeros(size(G))];

Hce = [H zeros(size(H))];
```

```
Jce = [0;0];

[Y,X] = dlsim (Fce,Gce,Hce,Jce,U);
stairs (T,Y)
legend ('cart (x)','pendulum (phi)')
```



شکل (۵-۵۰) پاسخ سیستم حلقه بسته همراه با رویتگر دیجیتالی

همان‌طور که از شکل مشخص است تمام ملزومات برآورده می‌شود.

۵ -۷ طراحی کنترلر با فیدبک تمامی متغیرهای حالت در محیط سیمولینک

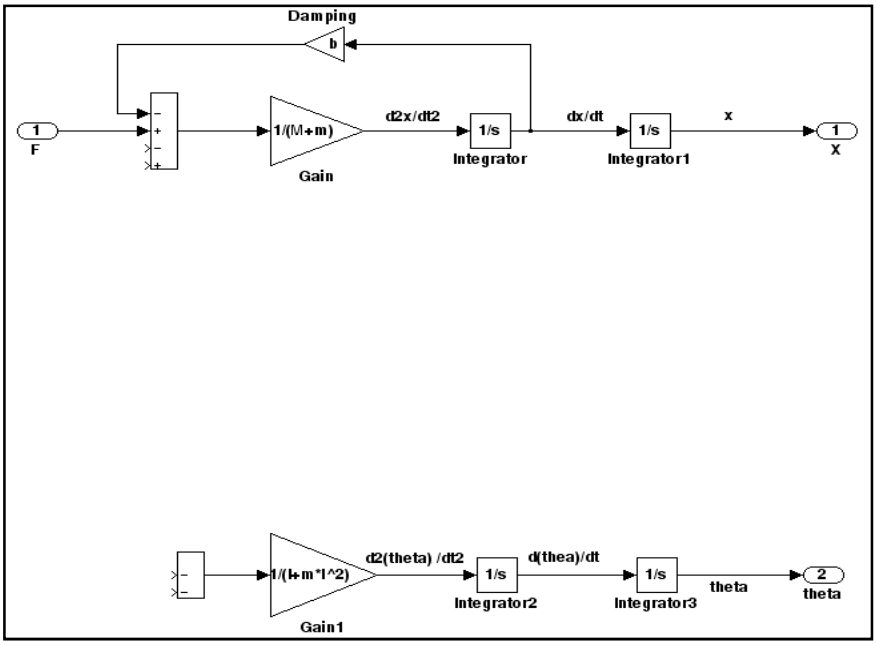
همانطور که قبلاً گفته شد، معادلات حاکم بر سیستم، معادلات نیوتن می‌باشد که می‌توان این معادلات دینامیکی را به صورت زیر نوشت:

$$(M+m)\ddot{x}+b\dot{x}+ml\ddot{\theta}Cos\theta-ml\dot{\theta}^2Sin\theta=F$$

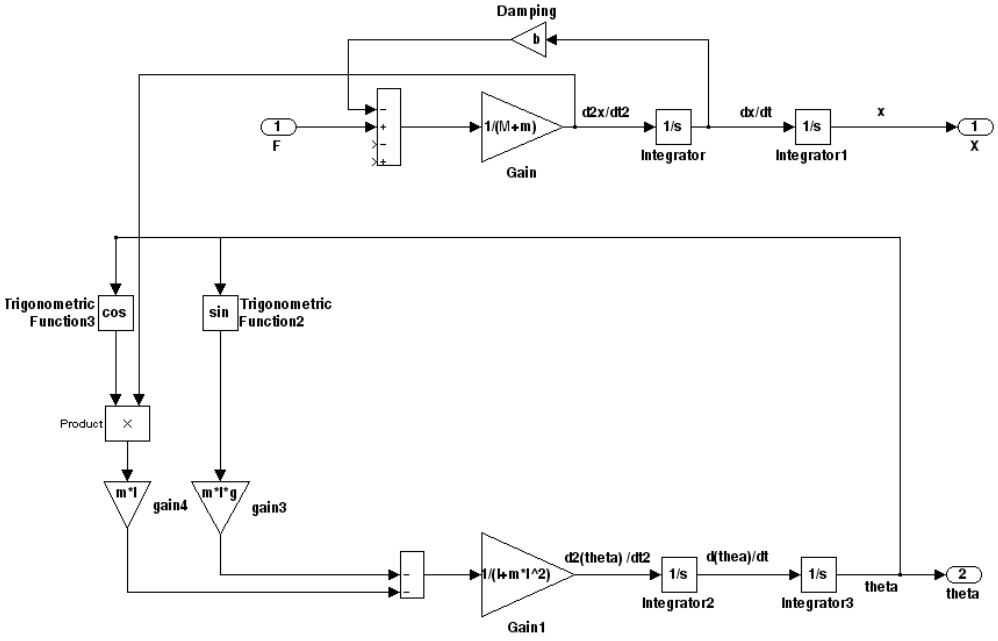
$$(I+ml^2)\ddot{\theta}+mglSin\theta=-ml\ddot{x}Cos\theta$$

در این قسمت قصد داریم با استفاده از روش فیدبک کلیه متغیرهای حالت،کنترلری برای سیستم خطی شده طراحی کنیم. ولی برای تمرین بیشتر ابتدا مدل غیرخطی سیستم را در این محیط شبیه سازی می‌کنیم تا با نحوه مدل سازی یک سیستم غیرخطی پیچیده در محیط سیمولینک بیشتر آشنا شویم.

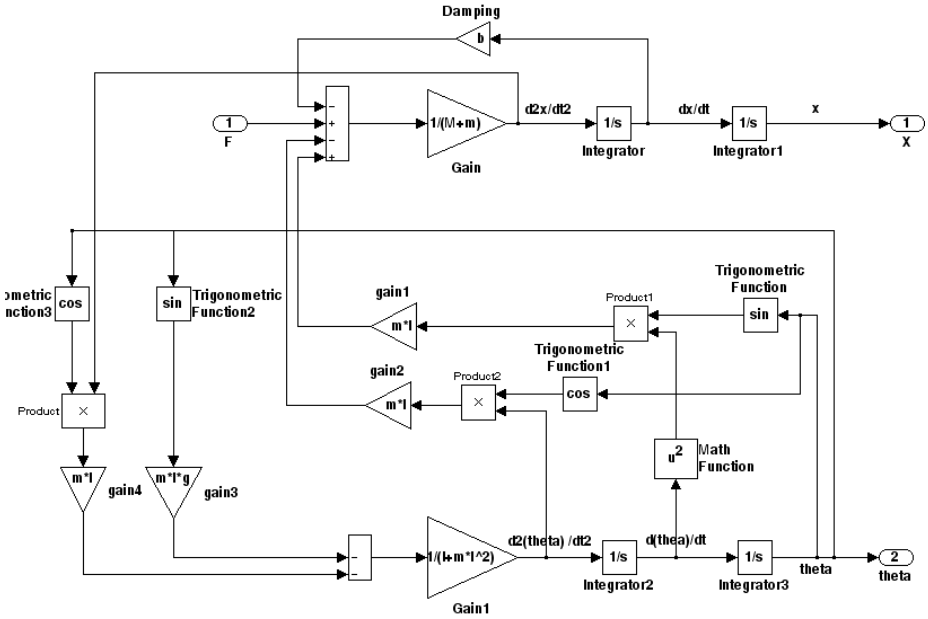
در ابتدا باید در محیط سیمولینک مدل دینامیکی را بسازیم. بنابراین مطابق شکل‌های زیر بلوک‌های مورد نظر را از کتابخانه سیمولینک وارد پنجره مدل سازی کرده و با استفاده از سیگنال آنها را به هم ارتباط دهید. مراحل مدل سازی سیستم غیرخطی به صورت زیر می‌باشد:



شکل (۵-۵۱) مرحله اول شبیه سازی مدل دینامیکی

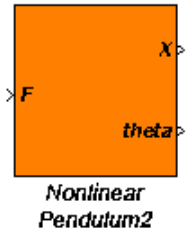


شکل (۵-۵۲) مرحله دوم شبیه سازی مدل دینامیکی



شکل (۵-۵۳) مرحله سوم شبیه سازی مدل دینامیکی

حال تمام بلوک‌های کشیده شده در شکل قبل را انتخاب کنید و با کلیک راست کردن بر روی صفحه و انتخاب subsystem، بلوک‌ها را در داخل یک زیر سیستم قرار دهید.

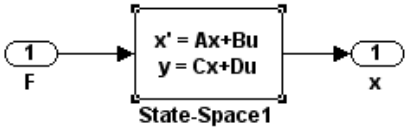


به ادامه مساله، یعنی طراحی کنترلر برای سیستم خطی شده بر می‌گردیم. با توجه به معادلات فضای حالت زیر مدل دینامیکی را در محیط سیمولینک ایجاد می‌کنیم.

$$\begin{bmatrix}\dot{x}\\ \ddot{x}\\ \dot{\theta}\\ \ddot{\theta}\end{bmatrix}=\begin{bmatrix}0 & 1 & 0 & 0\\ 0 & \frac{-b(I+ml^2)}{I(M+m)+Mml^2} & \frac{m^2gl^2}{I(M+m)+Mml^2} & 0\\ 0 & 0 & 0 & 1\\ 0 & \frac{-mbl}{I(M+m)+Mml^2} & \frac{mgl(M+m)}{I(M+m)+Mml^2} & 0\end{bmatrix}\begin{bmatrix}x\\ \dot{x}\\ \theta\\ \dot{\theta}\end{bmatrix}+\begin{bmatrix}0\\ \frac{(I+ml^2)}{I(M+m)+Mml^2}\\ 0\\ \frac{ml}{I(M+m)+Mml^2}\end{bmatrix}F$$

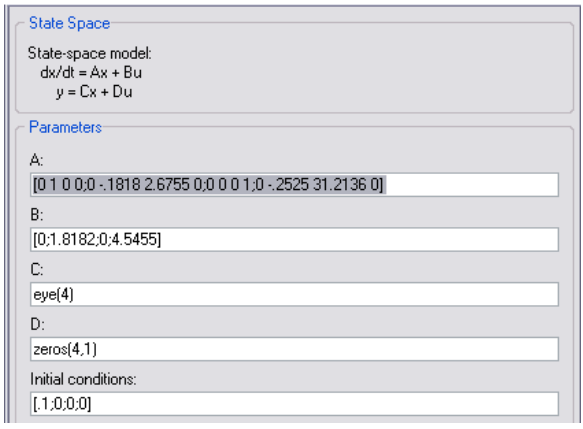
$$y=\begin{bmatrix}1&0&0&0\\0&0&1&0\end{bmatrix}\begin{bmatrix}x\\\dot{x}\\\theta\\\dot{\theta}\end{bmatrix}+\begin{bmatrix}0\\0\end{bmatrix}F$$

```
A = [0      1              0          0;
     0 -(i+m*l^2)*b/p  (m^2*g*l^2)/p   0;
     0      0              0          1;
     0 -(m*l*b)/p      m*g*l*(M+m)/p  0];
B = [0;  (i+m*l^2)/p;  0;  m*l/p];
C = [1 0 0 0;
     0 0 1 0];
D = [0;0];
```

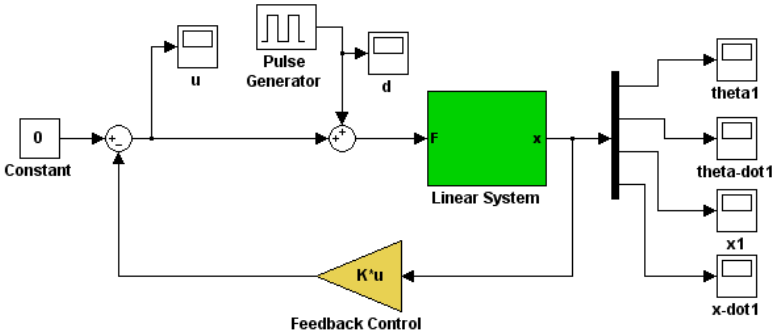


روی state space دوبار کلیک کنید و مقادیر زیر را برای ماتریس‌های A,B,C,D بنویسید.

```
A =
         0    1.0000         0         0
         0   -0.1818    2.6727         0
         0         0         0    1.0000
         0   -0.4545   31.1818         0
 B =[0;1.8182;0;4.5455]
C = [1 0 0 0;0 0 1 0];
D = [0;0];
```

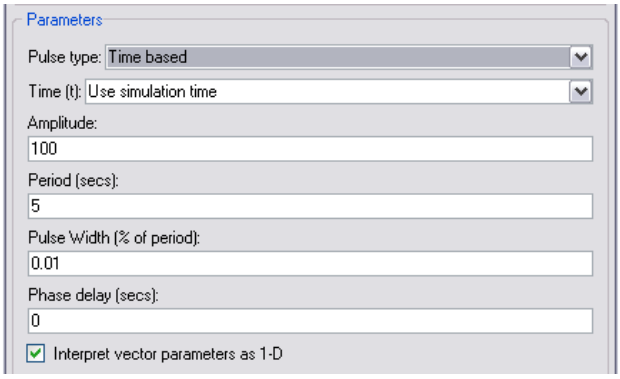


با کلیک راست کردن بر روی صفحه و انتخاب Subsystem، بلوک‌ها را در داخل یک زیر سیستم قرار دهید. سپس سیستم حلقه بسته زیر را ایجاد کنید.

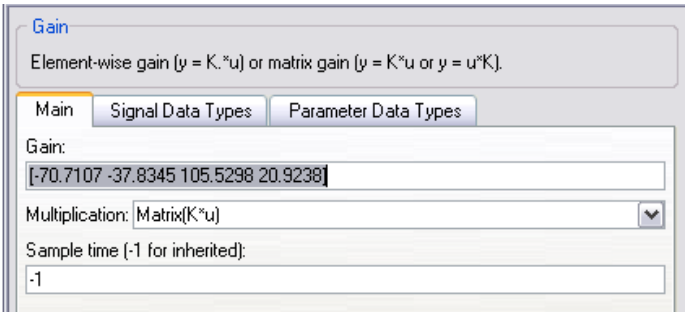


شکل (۵-۵۴) شبیه سازی مدل دینامیکی

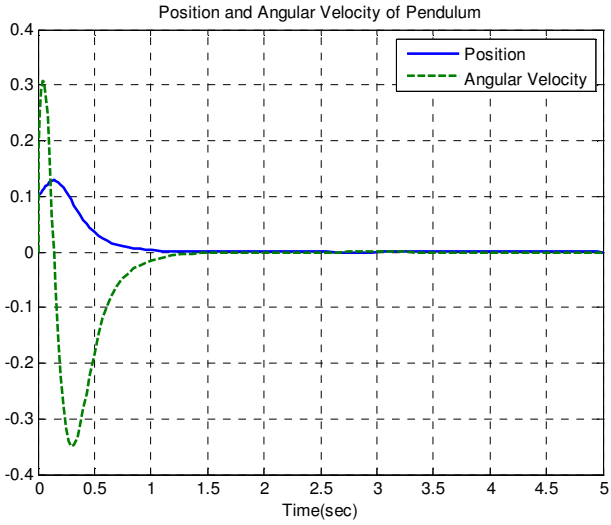
روی Pulse Generator دوبار کلیک کنید و تنظیمات زیر را انجام دهید.



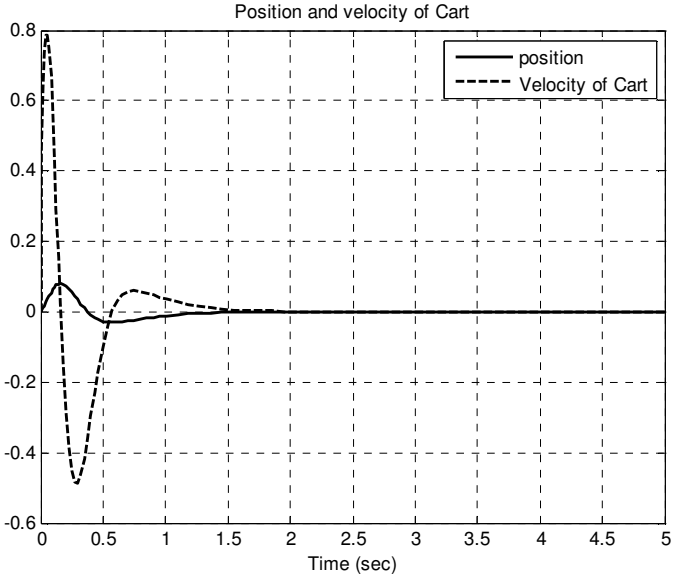
برای کنترلر از آنچه که در قسمت طراحی فیدبک حالت آورده شده یعنی کنترل LQR استفاده کرده و مقدار K را به جای بهره فیدبک قرار دهید. شرایط اولیه را به صورت [0.1;0;0;0] قرار دهید؛ به این معنا که پاندول از موقعیت 0.1 rad رها شده است.



با اجرای برنامه خروجی‌های زیر حاصل می‌شود.



شکل (۵-۵۵) موقعیت و سرعت زاویه ای



شکل (۵-۵۶) موقعیت و سرعت ارابه

پروژه ششم

سیستم کنترل موقعیت هواپیما در صفحه قائم

اهداف این پروژه

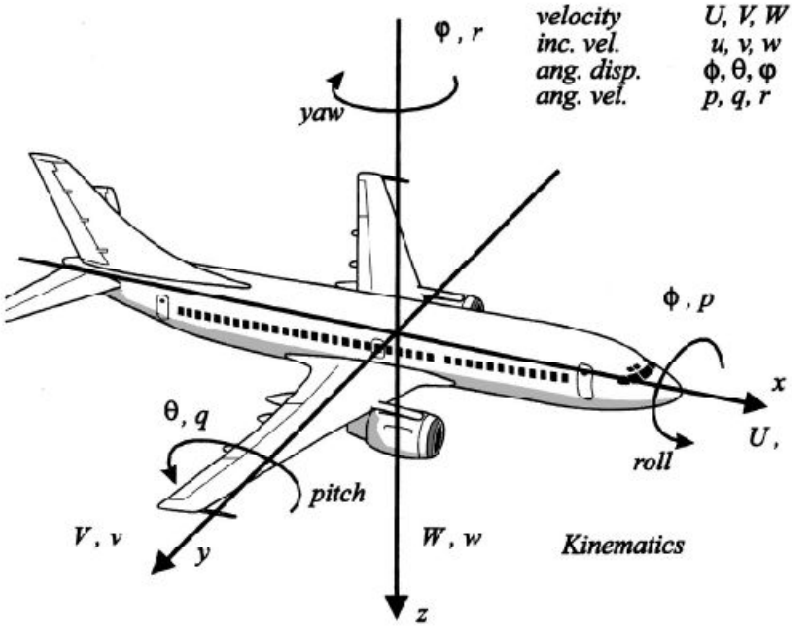
- **مدل سازی**
- **طراحی کنترلر PID**
- **رسم مکان هندسی ریشه‌ها (Root Locus)**
- **پاسخ فرکانسی**
- **طراحی فضای حالت**
- **طراحی کنترلر دیجیتالی**
- **طراحی کنترلر به روش فیدبک کلیه متغیرهای حالت در محیط سیمولینک**

۶-۱ مدل سازی

۶-۱-۱ مدل سازی سیستم براساس اصول فیزیکی و استخراج معادلات سیستم

معادلات حاکم بر حرکت هواپیما بسیار پیچیده است و معادلات دیفرانسیل کوپل شده غیرخطی بر آن حاکم است. به هر حال تحت یکسری فرضیات معین این معادلات قابل دکوپله شدن و خطی شدن در جهت طولی و عرضی می‌باشند. کنترل پیچ[1]، یک مساله طولی است و در این مثال قصد داریم یک اوتوپایلوت طراحی کنیم به‌طوری‌که بتواند پیچ هواپیما (زاویه فراز) را کنترل کند. محورهای مختصات اصلی در هواپیما در شکل (۶-۱) نمایش داده شده است:

[1] Pitch angle



شکل (۶-۱) مدل هواپیما و محورهای مختصات اصلی

فرض کنید که هواپیما در یک کروز ماندگار، ارتفاع و سرعت ثابت قرار گرفته است. بنابراین نیروی محوری و درگ حذف می‌شوند و نیروی لیفت و وزن با هم در تعادل می‌باشند. همچنین فرض کنید که زاویه پیچ تحت هیچ شرایطی سرعت هواپیما را تغییر نمی‌دهد (این موضوع غیر واقعی است ولی تا حدودی مساله را ساده می-کند). با در نظر گرفتن این فرضیات، معادلات طولی حرکت هواپیما را می‌توان به صورت زیر نوشت:

$$\begin{cases} \dot{\alpha} = \mu\Omega\sigma[-(C_L + C_D)\alpha + (1/\mu - C_L)q - (C_W Sin\gamma_e)\theta + C_L] \\ \dot{q} = \dfrac{\mu\Omega}{2i_{yy}}\{[C_M - \eta(C_L + C_D)]\alpha + [C_M + \sigma C_M(1 - \mu C_L)]q + (\eta C_W Sin\gamma_e)\delta_e\} \\ \dot{\theta} = \Omega q \end{cases} \qquad (۱-۶)$$

که متغیرهای به‌کار رفته به ترتیب عبارتند از:

- α زاویه حمله، q سرعت تغییرات پیچ، θ زاویه پیچ، δ_e زاویه انحراف بالابر، C_T ضریب تراست، C_D ضریب درگ ، C_L ضریب لیفت، C_W ضریب وزنی، C_M ضریب ممان پیچ، γ_e زاویه مسیر پرواز
- در رابطه $\mu = \dfrac{\rho_e S\,\bar{c}}{4m}$: ρ_e چگالی هوا، S سطح مقطع بال، $\bar{c}$ متوسط طول قوس، m جرم هواپیما
- در رابطه $\Omega = 2U/\bar{c}$: U سرعت پرواز تعادلی[1]

[1] Equilibrium flight speed

- در رابطه $\sigma = 1/(1+\mu C_L)$: σ ثابت سیگما، i_{yy} ممان اینرسی نرمالیزه شده، $\eta = \mu\sigma C_L$ ثابت نو

توجه: در این سیستم زاویه انحراف بالابر[1] به عنوان ورودی سیستم و خروجی، زاویه پیچ سیستم می‌باشد.

قبل از اینکه تابع تبدیل سیستم را به‌دست آوریم از یک سری مقادیر عددی برای ساده شدن معادلات مدل سازی استفاده می‌کنیم. این مقادیر از داده‌های هواپیمای تجاری **بوئینگ**[2] گرفته شده است.

$$\begin{cases} \dot{\alpha} = -0.313\alpha + 56.7q + 0.232\delta_e \\ \dot{q} = -0.013\alpha - 0.426q + 0.0203\delta_e \\ \dot{\theta} = 56.7q \end{cases} \qquad (۲-۶)$$

۶-۱-۲ تابع تبدیل سیستم

برای به‌دست آوردن تابع تبدیل سیستم نیاز داریم از معادله مدل سازی (۶-۲) تبدیل لاپلاس بگیریم. توجه کنید زمانی که در حال پیدا کردن تابع تبدیل سیستم هستیم شرایط اولیه را برابر صفر در نظر می‌گیریم.

$$\begin{cases} s\alpha(s) = -0.313\alpha(s) + 56.7q(s) + 0.232\delta_e(s) \\ sq(s) = -0.013\alpha(s) - 0.426q(s) + 0.0203\delta_e(s) \\ s\theta(s) = 56.7q(s) \end{cases} \qquad (۳-۶)$$

$$\frac{\theta(s)}{\delta_e(s)} = \frac{1.51s + 0.1774}{s^3 + 0.739s^2 + 0.921s} \qquad (۴-۶)$$

۶-۱-۳ مدل فضای حالت سیستم

با توجه به معادله (۶-۳)، مدل فضای حالت را می‌توان به صورت زیر نوشت:

$$\begin{cases} \dot{x}(t) = Ax(t) + Bu(t) \\ y(t) = Cx(t) + Du(t) \end{cases} \Rightarrow \begin{cases} \dfrac{d}{dt}\begin{bmatrix} \alpha \\ q \\ \theta \end{bmatrix} = \begin{bmatrix} -0.313 & 56.7 & 0 \\ -0.013 & -0.42 & 0 \\ 0 & 56.7 & 0 \end{bmatrix}\begin{bmatrix} \alpha \\ q \\ \theta \end{bmatrix} + \begin{bmatrix} 0.232 \\ 0.02 \\ 0 \end{bmatrix}\delta_e \\ y = [0 \quad 0 \quad 1]\begin{bmatrix} \alpha \\ q \\ \theta \end{bmatrix} + [0]\delta_e \end{cases} \qquad (۵-۶)$$

۶-۱-۴ قیدهای حاکم بر طراحی

قدم بعدی در مدل‌سازی سیستم این است که بدانیم این سیستم قرار است چه شرایطی را داشته باشد، یعنی قیدهای حاکم بر سیستم باید مشخص شوند.

- مقدار جهش سیستم باید کمتر از 10% باشد.
- زمان خیز کمتر از 2 ثانیه باشد.
- زمان نشست باید کوچکتر از 10 ثانیه باشد.

[1] elevator deflection angle
[2] Boeing

- خطای حالت ماندگار باید کمتر از 2% باشد.

به‌طور مثال اگر ورودی سیستم 0.2rad (11 درجه) بود در آن صورت زاویه پیچ نباید از 0.22rad بیشتر شود از طرفی در مدت کمتر از 2 ثانیه به مقدار 0.2rad برسد و در مدت زمان 10 ثانیه مقدار ماندگار آن در بازه $(0.196 \sim 0.204)rad$ قرار بگیرد.

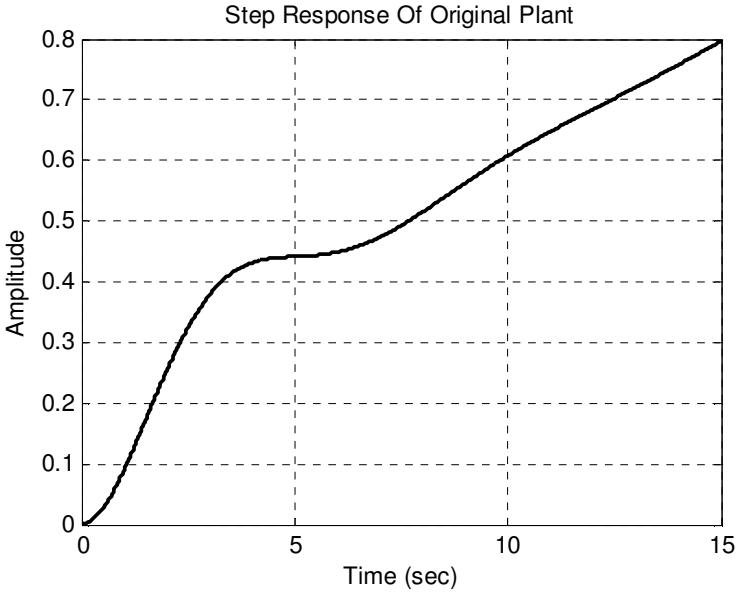
۶-۱-۵ پاسخ سیستم حلقه باز

دستورات زیر را در یک m-file جدید بنویسید:

```
%~~~~~~~~~~~open loop response~~~~~~~~~~
de=0.2;
num=[1.151 0.1774];
den=[1 0.739 0.921 0];
sys=tf(num,den);
step(de*sys)
title('Step Response Of Original Plant')
```

اگر مدل فضای حالت زیر را نیز وارد کنیم در هر دو حالت به یک جواب خواهیم رسید

```
A=[-0.313 56.7 0; -0.0139 -0.426 0; 0 56.7 0];
B=[0.232; 0.0203; 0];
C=[0 0 1];
D=[0];
Sys1=ss(A,B,C,D)
step(de*sys1)
```



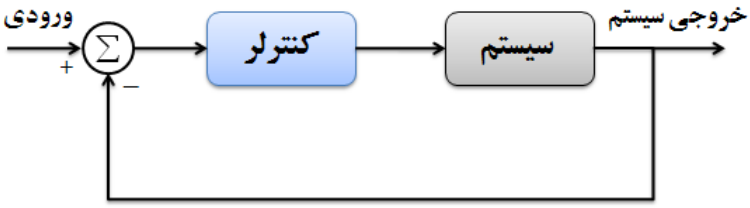
شکل (۶- ۲) پاسخ پله سیستم حلقه باز

۶-۲ طراحی کنترلر PID

از قسمت مدل سازی به یاد داریم که تابع تبدیل سیستم حلقه باز به صورت زیر به‌دست می‌آید:

$$\frac{\theta(s)}{\delta_e(s)} = \frac{1.51s + 0.1774}{s^3 + 0.739s^2 + 0.921s}$$

که ورودی زاویه انحراف بالابرنده به اندازه $0.2\,rad$ یا $11\deg$ و خروجی زاویه پیچ می‌باشد. بلوک نمودار سیستم کنترل PID با فیدبک واحد به صورت زیر می‌باشد:

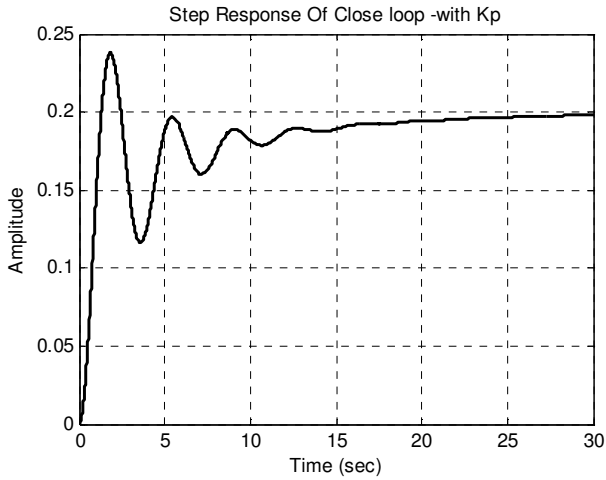


شکل (۶- ۳) بلوک نمودار سیستم کنترلی با فیدبک واحد

در ابتدا از یک کنترلر تناسبی استفاده می‌کنیم و تابع تبدیل سیستم حلقه بسته را به‌دست می‌آوریم. به‌دست آوردن این تابع تبدیل با دست تا حدودی وقت‌گیر است بنابراین برای حل این مشکل از دستور cloop استفاده می‌کنیم. در ابتدا قرار می‌دهیم $k_p = 2$ و خروجی آن را حساب می‌کنیم.

$$\frac{\theta(s)}{\delta_e(s)} = \frac{k_p.(1.51s + 0.17)}{s^3 + 0.739s^2 + (1.51K_p + 0.921)s + 0.17K_p} \qquad (۶\text{-}۶)$$

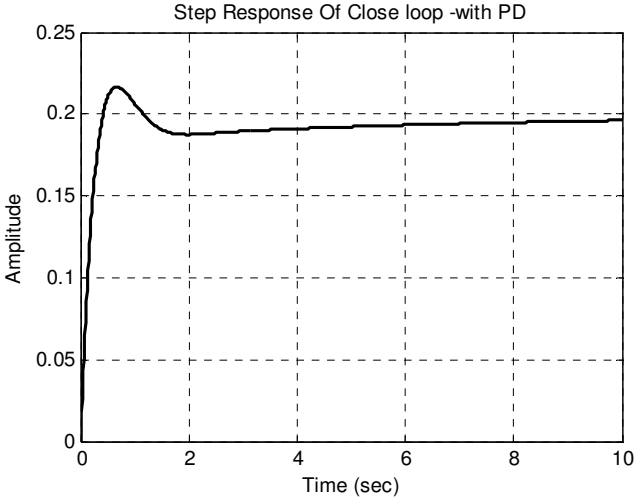
```
Kp=[1]; %Enter any numerical value for the proportional gain
num=[1.151 0.1774];
num1=conv(Kp,num);
den1=[1 0.739 0.921 0];
[numc,denc]=cloop (num1,den1)

de=0.2;
Kp=2;
numc=Kp*[1.151 0.1774];
denc=[1 0.739 1.151*Kp+0.921 0.1774*Kp];

t=0:0.01:30;
step (de*numc,denc,t)
```



شکل (۶-۴) پاسخ پله سیستم همراه با کنترل تناسبی

همان‌طور که از شکل بر می‌آید نیاز داریم که هم جهش و هم زمان نشست را بهبود دهیم. برای کاهش این دو مورد باید کنترلر مشتق‌گیر را به واحد کنترلی اضافه کنیم و تابع تبدیل سیستم حلقه بسته را به‌دست آوریم. پس دستورات زیر را در یک m-file جدید بنویسید:

```
de=0.2; kp=9;kd=4;
numpd=[kd kp];
num=[1.151 0.1774]; num1=conv(numpd,num);
den1=[1 0.739 0.921 0];
[numc,denc]=cloop (num1,den1)
t=0:0.01:10;
step (de*numc,denc,t)
```



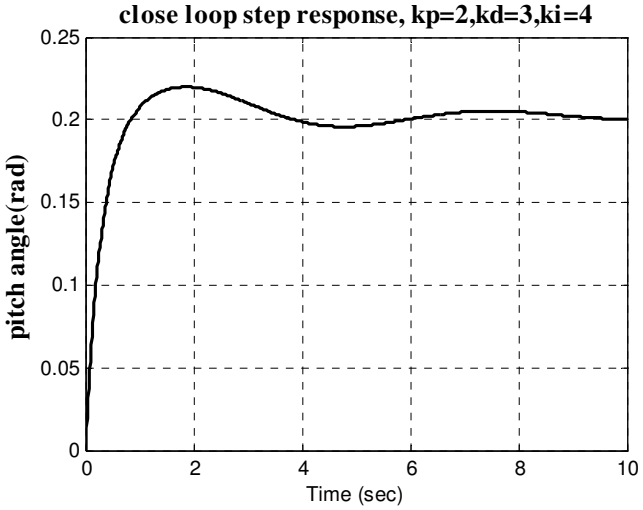
شکل (۶-۵) پاسخ پله سیستم همراه با کنترلر PD

پاسخ پله نشان می‌دهد که زمان خیز کمتر از 2 ثانیه و جهش کمتر از 10% و زمان نشست کمتر از 10 ثانیه است .

۶-۲-۱ کنترلر PID

تمام ملزومات طراحی با استفاده از کنترلر PD برآورده شده است. برای هموار کردن پیک جهش، یک کنترلر انتگرالی به واحد کنترلی اضافه می‌کنیم تا پیک شدید آن را حذف کند. بنابراین مقادیر بهره‌ها را به صورت $k_p = 2, k_I = 4, k_d = 3$ تنظیم کرده و پاسخ پله را بررسی می‌کنیم. دستورات زیر را در یک m-file جدید وارد کرده و برنامه را اجرا کنید.

```
de=0.2;
Kp=2; Kd=3; Ki=4;
numo=[1.151 0.1774];
deno=[1 0.739 0.921 0];
numpid=[Kd Kp Ki];
denpid=[1 0];

num1=conv(numo,numpid);
den1=conv(deno,denpid);

[numc,denc] = cloop(num1,den1);
t=0:0.01:10;
step (de*numc,denc,t)
title('close loop step response, kp=2,kd=3,ki=4')
ylabel('pitch angle(rad)')
```



شکل (۶- ۶) پاسخ پله سیستم همراه با کنترلر PID

۶-۳ رسم مکان هندسی ریشه‌ها (Root Locus)

از قسمت مدل‌سازی به یاد داریم که تابع تبدیل سیستم حلقه باز به صورت زیر به‌دست می‌آید:

$$\frac{\theta(s)}{\delta_e(s)} = \frac{1.51s + 0.1774}{s^3 + 0.739s^2 + 0.921s}$$

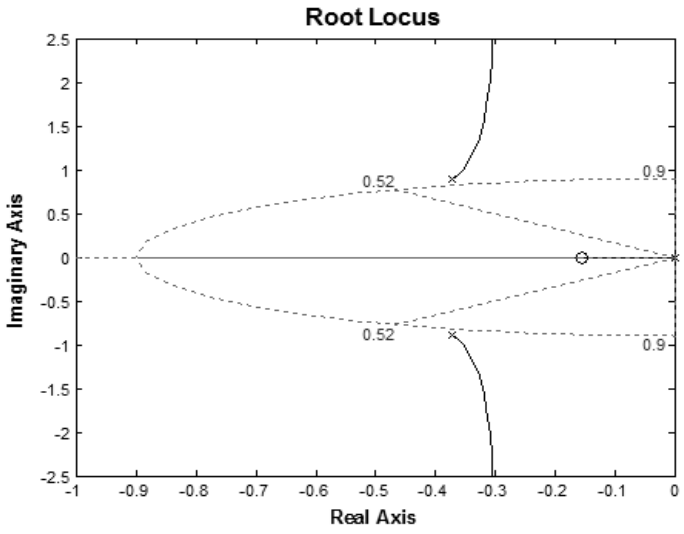
همان‌طور که می‌دانیم مکان هندسی ریشه‌ها تمام موقعیت‌های ممکن برای قطب‌های سیستم حلقه بسته را با یک کنترلر تناسبی نشان می‌دهد. از آنجایی که همه قطب‌ها پذیرفتنی نیستند بنابراین از دستور sgrid برای پیداکردن ناحیه مطلوب استفاده می‌نمائیم. همان‌طور که از شکل مشخص است، سیستم دو قطب در مبدا مختصات دارد که در راستای محور موهومی به سمت بینهایت می‌روند. با دستور sgrid می‌توانیم خطوط نسبت میرایی ثابت و فرکانس طبیعی را رسم کنیم. این دستور به دو آرگومان نیاز دارد نسبت میرایی $\zeta = 0.52$ که متناسب با جهش 10% است و فرکانس طبیعی $\omega_n = 0.9$.

$$\omega_n = \frac{1.8}{T_r}\ ,\quad M_p = e^{\frac{-\pi\zeta}{\sqrt{1-\zeta^2}}}\ ,\zeta\omega_n = \frac{4.6}{T_s} \qquad (۶-۷)$$

پس دستورات زیر را در ادامه m-file اضافه کنید:

```
num=[1.151 0.1774]; den=[1 0.739 0.921 0];
Wn=0.9; zeta=0.52;
rlocus (num,den)
sgrid (zeta,Wn)
axis ([-1 0 -2.5 2.5])
```

با اجرای دستورات بالا به نمودار زیر می‌رسید:



شکل (۶-۷) مکان هندسی ریشه‌ها

ناحیه بین دوخط چین، ناحیه‌ای است که در آن مقدار جهش کمتر از 10% و $\zeta \geq 0.52$، و نیز ناحیه خارج منحنی نیم بیضی، ناحیه‌ای می‌باشد که زمان نشست کمتر از سه ثانیه است و فرکانس طبیعی بزرگتر از 0.9 است. همان‌طورکه می‌بینیم، هیچ قسمتی از مکان هندسی ریشه‌ها در این ناحیه نیفتاده است. پس برای جبران اثر آن از یک کنترلر پیش‌فاز استفاده می‌کنیم تا مکان هندسی ریشه‌ها را به سمت چپ بکشاند و سیستم پایدار گردد.

۶-۳-۱ طراحی کنترلر پیش فاز

جبران کننده Lead مرتبه اول با استفاده از مکان هندسی ریشه‌ها طراحی می‌شود. این جبران کننده به فرم مکان هندسی ریشه‌ها به صورت زیر می‌باشد:

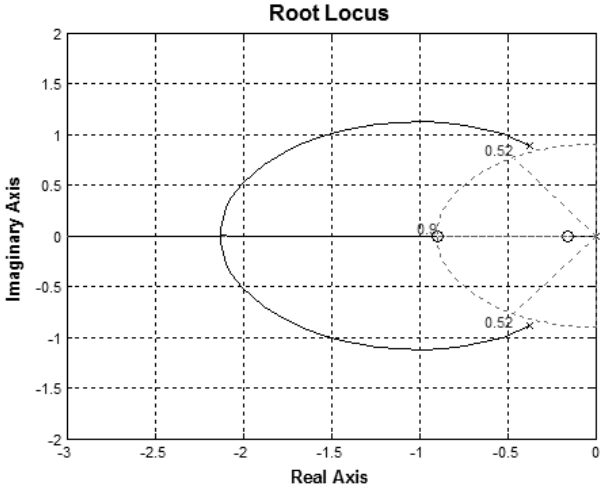
$$G(s) = K_c \frac{(s - z_0)}{(s - p_0)} \qquad (۶-۸)$$

که در آن بزرگی Z_0 کوچکتر از p_0 است. جبران کننده Lead باعث کشیده شدن مکان هندسی ریشه‌ها به سمت نیم صفحه چپ می‌گردد.

نکته: عموماً صفر را در همسایگی فرکانس طبیعی و قطب را در فاصله‌ای در حدود 3 الی 20 برابر مقدار موقعیت صفر قرار می‌دهند.

اجازه بدهید که Z_0=0.9 و P_0 =20. سپس تابع تبدیل سیستم حلقه بسته را همراه کنترلر Lead به‌دست می‌آوریم. دستورات زیر را در ادامه m-file اضافه کنید و برنامه را اجرا کنید:

```
num1=[1.151 0.1774]; den1=[1 0.739 0.921 0];
num2=[1 0.9];den2=[1 20];
num=conv(num1,num2); den=conv(den1,den2); Wn=0.9; zeta=0.52;
rlocus (num,den)
axis ([-3 0 -2 2])
sgrid (zeta,Wn)
```

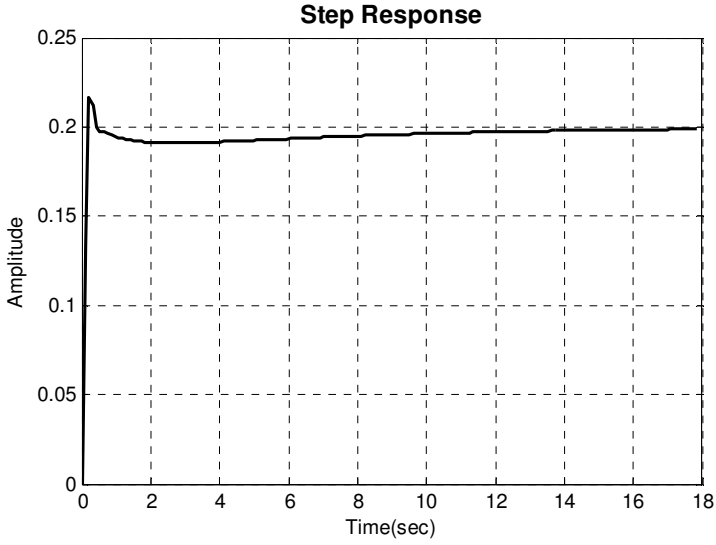


شکل (۶- ۸) مکان هندسی ریشه‌ها با جبران ساز پیش فاز

تاکنون مکان هندسی ریشه‌ها به سمت چپ انتقال پیدا کرده است. حال باید بهره‌ای را انتخاب کرده و به‌دنبال آن پاسخ پله سیستم را پیدا کنیم. برای انجام این کار از دستور rlocfind استفاده می‌کنیم. پس دستورات زیر را به ادامه m-file اضافه کنید:

```
[K, poles]=rlocfind (num,den)
de=0.2;
[numc,denc]=cloop (K*num,den,-1);
step (de*numc,denc)
```

نقطه ای نزدیک نقطه 1- روی محور حقیقی انتخاب کنید. این نقطه باید بهره ای حدود 200 داشته باشد. پاسخ پله به صورت زیر خواهد بود:



شکل (۶-۹) پاسخ پله سیستم حلقه بسته

۶-۳-۲ چگونه مسائل را با استفاده از مکان هندسی ریشه‌ها حل کنیم؟

۱- نمودار مکان هندسی ریشه‌ها را با استفاده از دستور sgrid برای تابع تبدیل سیستم رسم کنید.

۲- اگر لازم بود از یک کنترلر Lead یا Lag برای کشیدن منحنی به ناحیه مطلوب استفاده کنید.

۳- یک نقطه روی مکان هندسی انتخاب کنید و بهره متناظر با آن را به‌دست آورید.

۴- پاسخ پله سیستم را با بهره محاسبه شده به‌دست آورید.

۵- تعیین کنید چه چیزی برای تغییر پاسخ پله لازم است.

۶- در آن صورت کنترلر Lead یا Lag را تغییر دهید.

۷- مکان هندسی ریشه‌های جدید را در حالی‌که sgrid فعال است، به‌دست آورید.

۸- مرحله ۳ تا ۷ را تکرار کنید تا به جواب مورد نظر برسید.

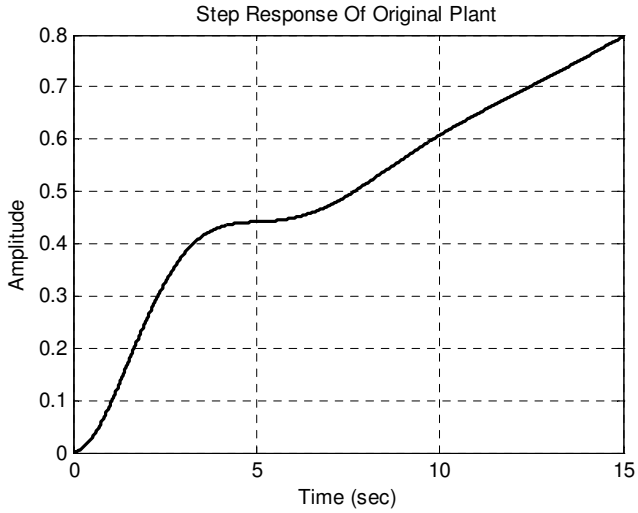
۶-۴ پاسخ فرکانسی

از بخش مدل سازی به یاد داریم که تابع تبدیل سیستم به صورت زیر به‌دست آمد:

$$\frac{\theta(s)}{\delta_e(s)} = \frac{1.51s + 0.1774}{s^3 + 0.739s^2 + 0.921s}$$

ورودی $\delta_e = 0.2rad$ و خروجی زاویه پیچ θ است. هدف ما از طراحی این کنترلر این است که به جهش کمتر از 10%، زمان خیز کمتر از 2 ثانیه، زمان نشست کمتر از 10 ثانیه، و خطای حالت ماندگار کمتر از 2% برسیم. می‌دانیم روش پاسخ فرکانسی بهترین روش طراحی برای سیستم‌هایی است که در حالت حلقه باز پایدار می‌باشند. پس در ابتدا پایداری سیستم حلقه باز را بررسی می‌کنیم. برنامه زیر را اجرا کنید:

```
de=0.2;
num=[1.151 0.1774];
den=[1 0.739 0.921 0];
step (de*num,den)
```

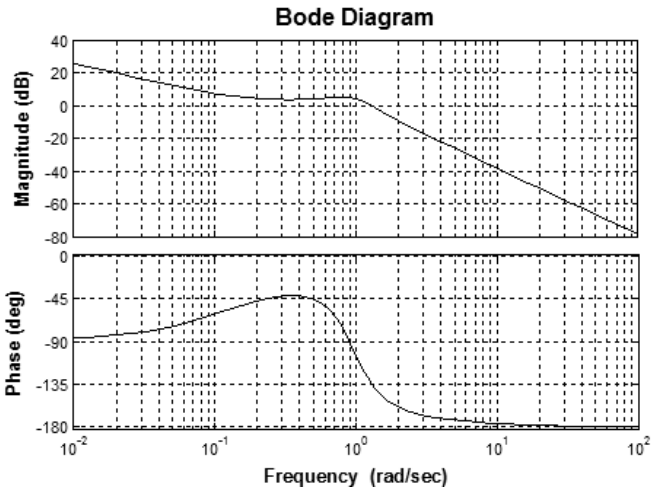


شکل (۶-۱۰) پاسخ پله سیستم حلقه باز

متاسفانه سیستم در حالت حلقه باز ناپایدار است ولی هنوز می‌توانیم از سیستم فیدبک با استفاده از روش پاسخ فرکانسی استفاده کنیم. البته در ابتدا باید نمودار بود سیستم حلقه باز را رسم نموده، آن را بررسی کنیم.

```
num=[1.151 0.1774];
den=[1 0.739 0.921 0];
bode (num,den)
```

هدف از طراحی این است که فرکانس طبیعی بزرگتر از 0.9 و ضریب میرایی $\zeta \geq 0.52$ باشد. با استفاده از دو معادله زیر به این نتیجه می‌رسیم که پهنای باند فرکانسی و حد فاز باید به ترتیب بزرگتر از 0.9 و 52 درجه باشد.

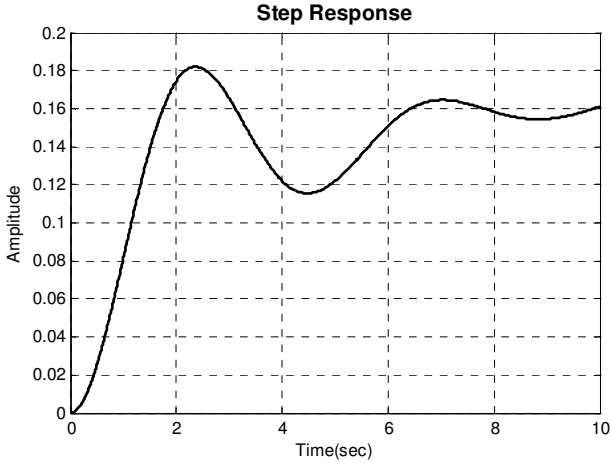


شکل (۶-۱۱) نمودار بود سیستم حلقه باز

$$T_r = \frac{1.8}{w_n} = \frac{1.8}{BW}$$
$$\zeta = \frac{PM(\deg ree)}{100} \qquad (۹-۶)$$

در حال حاضر فرکانس پهنای باند 1rad/s و حاشیه فاز 80 درجه را داریم و این مقادیر در ناحیه مطلوب قرار دارند. حال پاسخ پله سیستم حلقه بسته را بررسی می‌کنیم که باید به صورت زیر باشد:

```
[numc,denc]=cloop(num,den,-1);
de=0.2;
t=0:0.01:10;
step (de*numc,denc,t)
```



شکل (۶- ۱۲) پاسخ پله سیستم حلقه بسته

از نمودار مشاهده می‌شود که پاسخ حالت گذرا نامطلوب بوده و زمان نشست طولانی می‌باشد. برای بهبود پاسخ سیستم از یک جبران کننده Lead استفاده می‌کنیم.

۶-۴-۱ جبران کننده Lead

این جبران کننده یک فاز مثبت به سیستم اضافه می‌کند که این فاز اضافی باعث افزایش حد فاز می‌شود؛ بنابراین ضریب میرایی افزایش می‌یابد. زمان نشست با افزایش میرایی کاهش پیدا می‌کند. تابع تبدیل جبران کننده به صورت زیر می‌باشد:

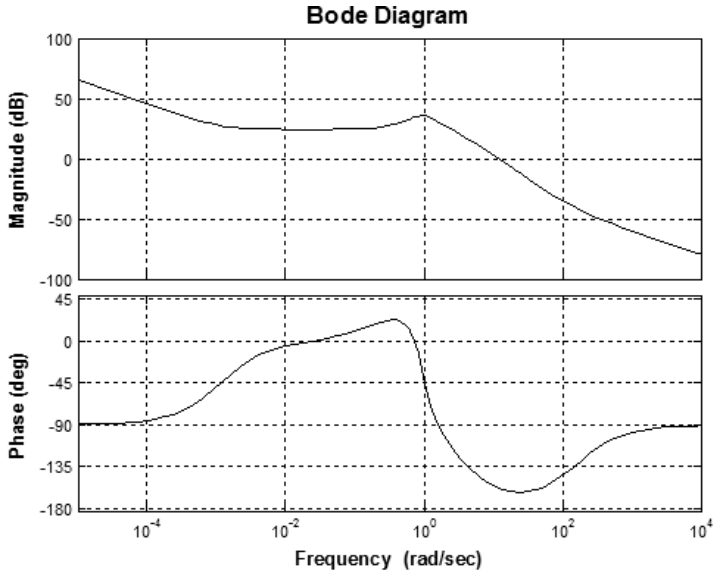
$$G_{lead}(s) = K_{lead}(\frac{1+Tas}{1+Ts}) \qquad (۱۰-۶)$$

که باید مقادیر K,T,a را پیدا کنیم.

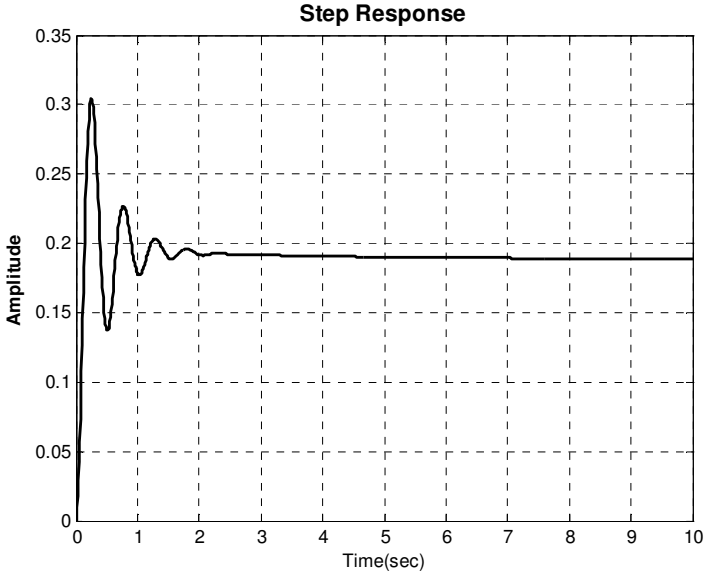
$$Sin\varphi = \frac{a-1}{a+1} \qquad (۱۱-۶)$$

از آنجایی‌که $\varphi \geq 52\deg$، در آن صورت باید $a \geq 8.43$ باشد. از طرفی پهنای باند فرکانسی باید بزرگتر از 0.9 باشد. $w_n = 1/T\sqrt{a}$ و این فرکانس به ما $T < 0.382$را می‌دهد. اجازه بدهید که برای این متغیرها مقادیر روبرو را در نظر بگیریم: $T = 0.3$, $a = 10$, $K = 0.1$ و دستورات زیر را در یک m-file وارد کنید:

```
num=[1 151 0.1774];
den=[1 0.739 0.921 0];

alead=10;
Tlead=0.3;
aleadtlead=alead*Tlead;
k=0.1;

numlead=k*[aleadtlead 1];
denlead=[Tlead 1];

num1=conv(num,numlead);
den1=conv(den,denlead);
bode(num1,den1)

[numc,denc]=cloop(num1,den1,-1);
de=0.2;
t=0:0.01:10;
figure
step (de*numc,denc,t)
```



شکل (۶- ۱۳) نمودار بود سیستم با اضافه شدن یک کنترلر Lead



شکل (۶- ۱۴) پاسخ پله سیستم حلقه بسته با کنترلر Lead

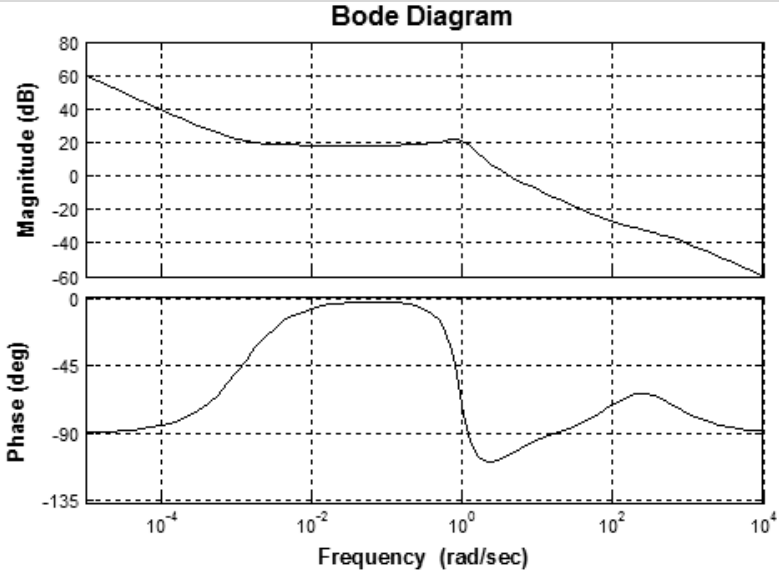
هرچند پهنای باند فرکانس و حد فاز افزایش پیدا کرده است ولی تمام معیارهای طراحی برآورده نشده است. حال اجازه بدهید که T را کاهش و a را افزایش بدهیم و مقادیر زیر را برای این سه متغیر در نظر بگیریم:

$K = 0.05\ ,\ a = 200\ \ ,T = 0.0025$

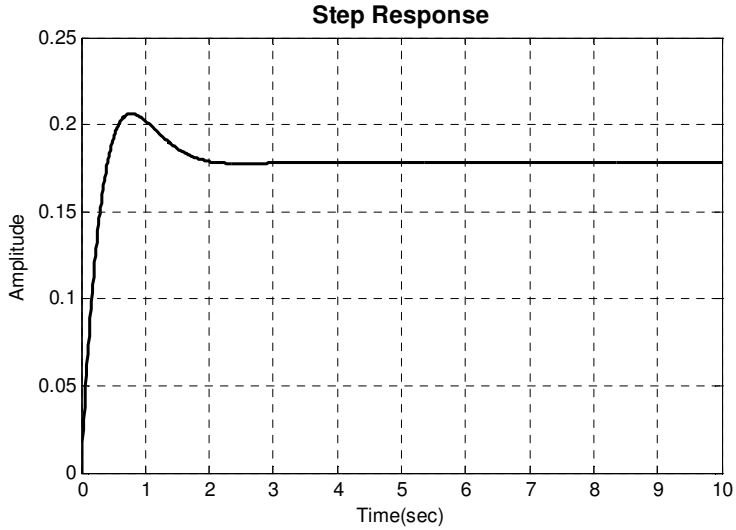
تابع تبدیل این کنترلر به صورت زیر خواهد بود:

$$G(s)=\frac{0.05(0.5s+1)}{(0.0025s+1)} \qquad (۱۲-۶)$$

```
num=[1 151 0.1774];
den=[1 0.739 0.921 0];
alead=200;
Tlead=0.0025;
aleadtlead=alead*Tlead;
k=0.05;
numlead=k*[aleadtlead 1];
denlead=[Tlead 1];
num1=conv(num,numlead);
den1=conv(den,denlead);
bode(num1,den1)
[numc,denc]=cloop(num1,den1,-1);
de=0.2;
t=0:0.01:10;
figure
step (de*numc,denc,t)
```



شکل (۶- ۱۵) نمودار بود سیستم با اضافه شدن یک کنترلر Lead



شکل (۶- ۱۶) پاسخ پله سیستم حلقه بسته با کنترلر Lead

از روی شکل مشخص است که تمام ملزومات طراحی برآورده شده است، ولی خطای حالت ماندگار مشکل دارد و باید آن را برطرف کرد. بنابراین از یک کنترلر Lag استفاده می‌کنیم.

۶-۴-۲ کنترلر Lag

کنترلر Lag خطای حالت ماندگار را کاهش می‌دهد. کنترلر Lag مرتبه اول به صورت زیر می‌باشد:

$$G_{lag}(s) = K_{lag} \Big/ a \left(\frac{1+Tas}{1+Ts}\right) \qquad (۶-۱۳)$$

خطای حالت ماندگار با فاکتور a_{lag} کاهش پیدا می‌کند. از نمودار بالا مشخص است که خطای حالت ماندگار حدود 10% است و باید 0.1 شود. توجه کنید که این جبران کننده پاسخ حالت گذرا را تغییر نمی‌دهد. با چند بار سعی و خطا، مقادیر زیر را برای متغیرها در نظر می‌گیریم:

$K = 1.5$, $a = 0.1$, $T = 20$

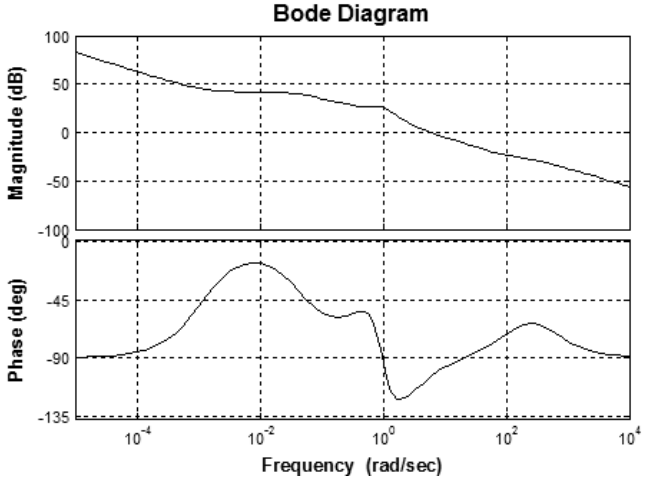
پس تابع تبدیل Lag به صورت زیر در می‌آید:

$$G_{lag}(s) = \frac{15(2s+1)}{(20s+1)}$$

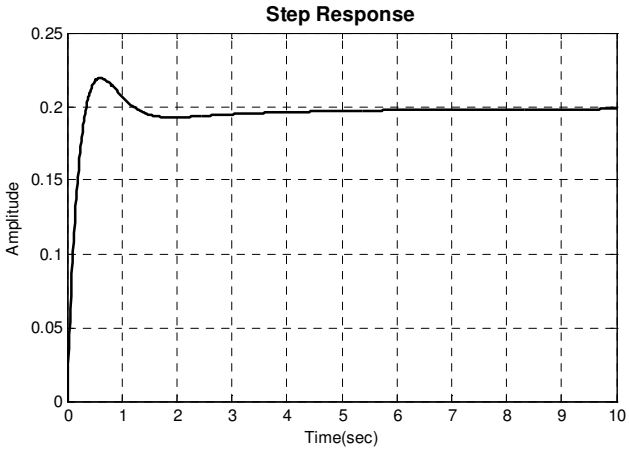
حال پاسخ به ورودی پله و نمودار بود را مشاهده می‌کنیم:

```
num=[1 151 0.1774];
den=[1 0.739 0.921 0];
alead=200;
```

```
Tlead=0.0025;
aleadtlead=alead*Tlead;
k=0.05;

numlead=k*[aleadtlead 1];
denlead=[Tlead 1];

num1=conv(num,numlead);
den1=conv(den,denlead);

Tlag=20; alag=0.1;
at=alag*Tlag;
k2=1.5;
numlag=k2/alag*[at 1];
denlag=[Tlag 1];

num2=conv(num1,numlag);
den2=conv(den1,denlag);
bode (num2,den2)
[numc2,denc2]=cloop(num2,den2,-1);
Figure
t=0:0.01:10;
step (0.2*numc2,denc2,t)
```



شکل (۶-۱۷) نمودار بود سیستم با اضافه شدن یک کنترلر Lead-Lag



شکل (۶-۱۸) پاسخ پله سیستم حلقه بسته با کنترلر Lead-Lag

همانطور که می‌بینید مشخصات حالت گذرا ثابت باقی مانده است ولی خطای حالت ماندگار کاهش پیدا کرده است.

۶-۵ طراحی فضای حالت

در قسمت مدل سازی، معادلات فضای حالت به صورت زیر به‌دست آمد:

$$\frac{d}{dt}\begin{bmatrix}\alpha\\ q\\ \theta\end{bmatrix}=\begin{bmatrix}-0.313 & 56.7 & 0\\ -0.013 & -0.42 & 0\\ 0 & 56.7 & 0\end{bmatrix}\begin{bmatrix}\alpha\\ q\\ \theta\end{bmatrix}+\begin{bmatrix}0.232\\ 0.02\\ 0\end{bmatrix}\delta_e$$

$$y=[0 \quad 0 \quad 1]\begin{bmatrix}\alpha\\ q\\ \theta\end{bmatrix}+[0]\delta_e$$

که ورودی، زاویه انحراف بالابرنده [1] به اندازه $0.2rad$ یا 11deg و خروجی زاویه پیچ می‌باشد. در ابتدا **کنترل پذیری و رویت پذیری** متغیرهای حالت را بررسی می‌کنیم:

ماتریس کنترل پذیری به صورت $C=\begin{bmatrix}B & AB & A^2B & \dots A^{n-1}B\end{bmatrix}$ تعریف می‌شود که باید مرتبه آن n باشد.

مرتبه یک ماتریس یعنی اینکه چند تا از سطرها یا ستون‌های آن مستقل از هم هستند.

در یک سیستم، تمام متغیرهای حالت باید رویت‌پذیر باشند. ماتریس رویت‌پذیری نیز به صورت $O=\begin{bmatrix}C & CA & CA^2 \dots CA^{n-1}\end{bmatrix}$ تعریف می‌شود که مرتبه آن نیز باید n باشد. از آنجایی که هر دو ماتریس C,O 3×3 می‌باشند پس مرتبه هر دو ماتریس نیز باید سه باشد. پس دستورات زیر را در یک m-file وارد کنید:

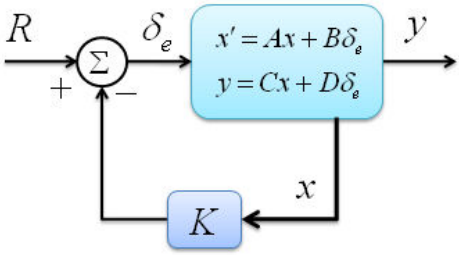
[1] Elevator deflection

```
A=[-0.313       56.7    0;
   -0.0139      -0.426  0;
   0            56.7    0];
B=[0.232;
   0.0203;
   0];
C=[0 0 1];
D=[0];

co=ctrb (A,B);
ob=obsv (A,C);
Controllability=rank(co)
Observability=rank(ob)
```

Controllability =3
Observability =3

۶-۵-۱ کنترل با استفاده از جایابی قطب‌ها

شماتیک این سیستم کنترلی به صورت زیر می‌باشد:



شکل (۶- ۱۹) کنترل به روش فیدبک کلیه متغیرهای حالت

که در بلوک بالا:

K: ماتریس کنترل، X : ماتریس متغیرهای حالت، $\delta_e = -K.X$: ورودی، R : ورودی مرجع

برای حل این مساله از روش دیگری به نام LQR استفاده می‌کنیم. این روش به شما اجازه می‌دهد که ماتریس کنترل بهینه ای را پیدا کنید به‌طوری‌که تعادلی بین خطای سیستم و تلاش کنترلی ایجاد شود.

در این روش به سه پارامتر نیاز داریم: ماتریس R، ماتریس Q و ماتریس وزنی ρ. برای ساده سازی $R = \rho = 1$ و $Q = C^T \times C$ در نظر می‌گیریم. فرامین زیر را در پنجره دستورات MATLAB وارد کنید:

```
C=[0 0 1];
Q=C'*C
```

حالا آماده‌ایم که ماتریس کنترلی را پیدا کنیم و پاسخ سیستم را ببینیم. اجازه بدهید فاکتور وزنی را برابر 50 انتخاب کنیم. دستورات زیر را در یک m-file وارد کنید:

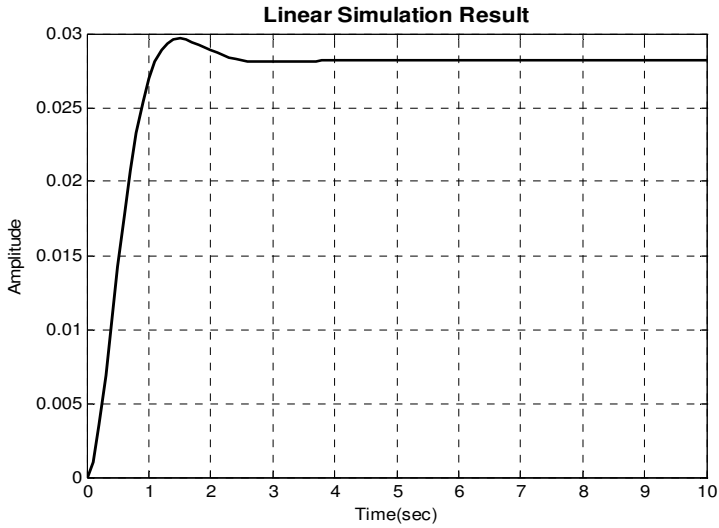
```
t=0:0.1:10;
de=0.2*ones(size(t));
```

```
yo=[0 0 0];

A=[-0.313  56.7  0;-0.0139  -0.426  0;0  56.7    0];
B=[0.232;0.0203;0];
C=[0 0 1];
D=[0];
p=50;
Q=[0 0 0;0 0 0;0 0 p];
[K]= lqr (A,B,Q,1)
lsim (A-B*K,B,C,D,de,t,yo)
```

بعد از اجرای برنامه پاسخ سیستم باید به صورت زیر باشد:

K =

-0.6435 169.6950 7.0711

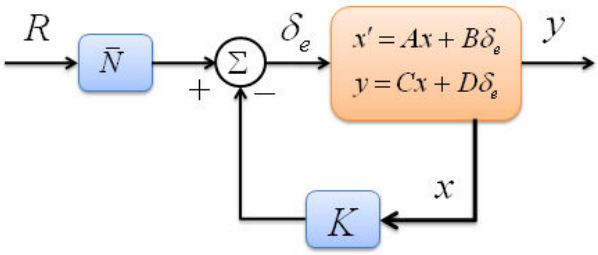


شکل (۶-۲۰) پاسخ پله سیستم با فیدبک کلیه متغیرهای حالت

مقدار جهش و زمان نشست رضایت بخش است ولی خطای حالت ماندگار زیاد است. پس یک ورودی مرجع به ابتدای آن اضافه می‌کنیم.

۶-۵-۲ اضافه کردن ورودی مرجع

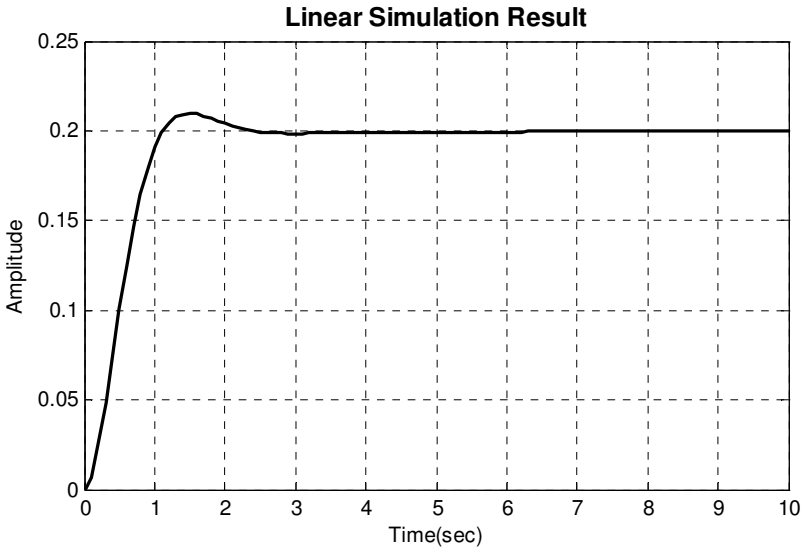
متمایز از بقیه روش‌های طراحی، در روش فیدبک تمام متغیرهای حالت، خروجی سیستم با ورودی مرجع مقایسه نمی‌شود، بلکه خروجی ضرب در بهره با ورودی رفرنس مقایسه می‌گردد. برای به‌دست آوردن خروجی دلخواه باید خروجی با ورودی مرجع مساوی باشد. این کار با استفاده از ضریب پیشخور Nbar انجام می‌شود.



شکل (۶- ۲۱) کنترل به روش فیدبک کلیه متغیرهای حالت با ورودی مرجع

فرامین زیر را در یک m-file جدید وارد کنید:

```
t=0:0.1:10;
de=0.2*ones(size(t));
yo=[0 0 0];
A=[-0.313  56.7  0;-0.0139  -0.426  0;0  56.7    0];
B=[0.232;0.0203;0];
C=[0 0 1];
D=[0];
x=50;
Q=[0 0 0;0 0 0;0 0 x];
[K]= lqr (A,B,Q,1)
Nbar = rscale(A,B,C,D,K)
lsim (A-B*K,B*Nbar,C,D,de,t,yo)
```



شکل (۶- ۲۲) پاسخ پله سیستم با فیدبک کلیه متغیرهای حالت با ورودی مرجع

دراین حالت خطای حالت ماندگار حذف شده، و تمام معیارهای طراحی برآورده می‌شود.

۶-۶ طراحی کنترلر دیجیتالی

در این قسمت هدف این است که از روش فضای حالت برای طراحی کنترلر دیجیتالی استفاده کنیم و به جهشی کمتر از 10%، زمان نشستی کمتر از 10 ثانیه، و زمان خیزی کمتر از 2 ثانیه، و خطای حالت ماندگاری کمتر از 2% برسیم.

۶-۶-۱ فضای حالت گسسته

اولین کاری که باید انجام دهیم این است که مدل فضای حالت پیوسته را به فضای گسسته ببریم. بنابراین از دستور c2dm استفاده می‌کنیم. در این قسمت به آرگومانهای زیر نیاز داریم:

ماترسی‌هایA،B،C،D، زمان نمونه‌برداری، و روش طراحی.

زمان نمونه‌برداری، باید کوچکتر از $1/(30\times BW)$ باشد. روش طراحی را ZOH انتخاب می‌کنیم. از نمودار بود به یاد داریم که پهنای باند فرکانسی تقریباً 2rad/sec است، پس زمان نمونه برداری را 1/100 در نظر می‌گیریم. حال می‌توانیم از دستور c2dm استفاده کنیم و مدل فضای حالت گسسته را به‌دست آوریم.

```
A = [-0.313  56.7 0;-0.013  -0.42 0;0 56.7 0];
B = [0.232;0.0203;0];
C=[0 0 1];
D=[0];
Ts=1/100;
[F,G,H,J] = c2dm (A,B,C,D,Ts,'zoh')
```

مدل فضای حالت گسسته به صورت زیر به‌دست می‌آید:

$$\begin{bmatrix}\alpha(k)\\ q(k)\\ \theta(k)\end{bmatrix}=\begin{bmatrix}0.996 & 0.056 & 0\\ -0.0001 & 0.995 & 0\\ 0 & 0.565 & 1\end{bmatrix}\begin{bmatrix}\alpha(k-1)\\ q(k-1)\\ \theta(k-1)\end{bmatrix}+\begin{bmatrix}0.0024\\ 0.0002\\ 0.0001\end{bmatrix}[\delta_e(k-1)]$$

$$y=[0\quad 0\quad 1]\begin{bmatrix}\alpha(k-1)\\ q(k-1)\\ \theta(k-1)\end{bmatrix}+[0]\times[\delta_e(k-1)]$$

۶-۶-۲ بررسی کنترل پذیری و رویت پذیری

قدم بعدی این است که کنترل‌پذیری و رویت‌پذیری را بررسی کنیم. ماتریس کنترل‌پذیری به صورت زیر می‌باشد:

$$C=[G\ FG\ F^2G\ \dots\ F^{n-1}G] \qquad (۱۴-۶)$$

که رتبه ماتریس باید n بوده، و رتبه ماتریس رویت‌پذیری نیز که به صورت زیر تعریف می‌شود باید برابر n باشد.

$$O=[H\ HF\ HF^2\ \dots\ HF^{n-1}]^T \qquad (۱۵-۶)$$

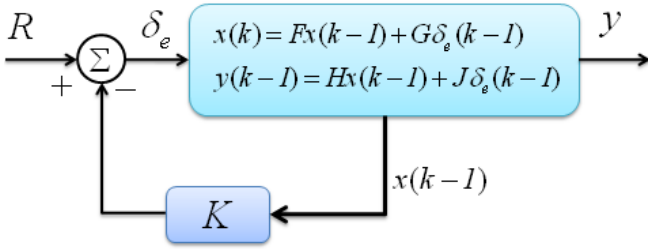
از آنجایی‌که هر دو ماتریس C,O 3×3 می‌باشند پس مرتبه هر دو ماتریس باید 3 باشد.

```
F = [0.996 0.0564 0;-0.0001  0.99 0;0 0.5658  1];
```

```
G = [0.0024;0.0002;0.0001];
H = [0     0    1];
J = [0];
co = ctrb (F,G);
ob = obsv (F,H);
Controllability = rank (co)
Observability = rank (ob)
```

۶-۶-۳ طراحی کنترلر با استفاده از جایابی قطب‌ها

شماتیک روش فیدبک تمام متغیرهای حالت به صورت زیر می‌باشد:



شکل (۶-۲۳) کنترل به روش فیدبک کلیه متغیرهای حالت

که در بلوک نمودار بالا:

K: ماتریس کنترل

X: ماتریس متغیرهای حالت

$\delta_e = -K.X$: ورودی

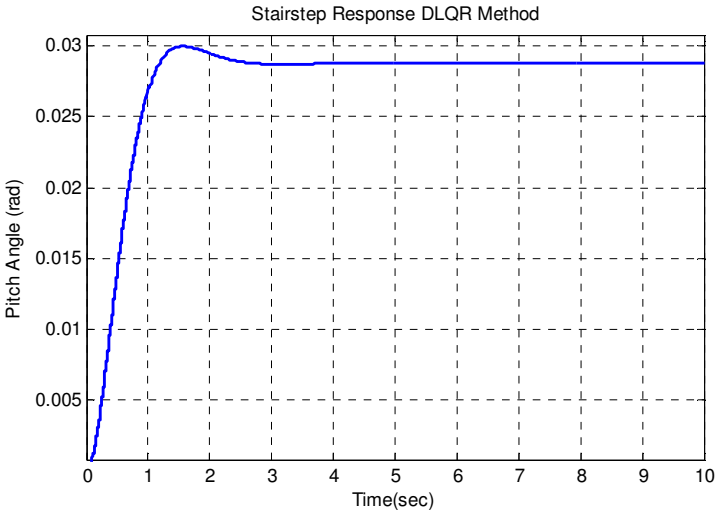
R: ورودی مرجع

برای حل این مساله از روش دیگری به نام LQR استفاده می‌کنیم. این روش به شما اجازه می‌دهد تا ماتریس کنترل بهینه‌ای را پیدا کنید که تعادلی بین خطای سیستم و تلاش کنترلی ایجاد شود. در این روش به سه پارامتر نیاز داریم: ماتریس R، ماتریس Q، و ماتریس وزنی ρ. برای ساده‌سازی $R=1$ فرض شده و $Q=C^T\times C$ را در نظر می‌گیریم. حال ماتریس k را به‌دست می‌آوریم. در ابتدا ماتریس وزنی را برابر $\rho=50$ در نظر گرفته، و دستورات زیر را در یک m-file وارد می‌کنیم:

```
t=0:0.01:10;
de=0.2*ones(size(t));
F = [0.9968              0.05649         0
    -0.0001              0.9957          0
     0                   0.5658          1];
G = [0.0024;
     0.0002;
     0.0001];
H = [0     0    1];
J = [0];
```

```
p=50;
Q = [0 0 0
     0 0 0
     0 0 p];
[K] = dlqr (F,G,Q,1)
[x] = dlsim  (F-G*K,G,H,J,de);
stairs  (t,x)
```

بعد از اجرای برنامه، پاسخ پله سیستم به صورت زیر به‌دست می‌آید:

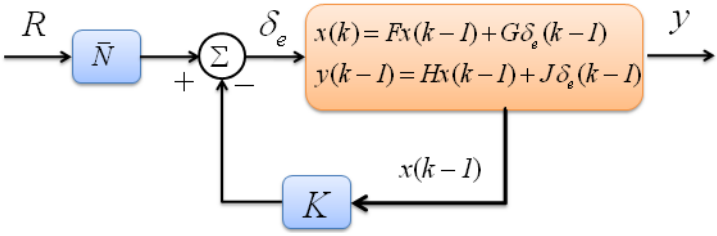


شکل (۶- ۲۴) پاسخ پله سیستم با فیدبک کلیه متغیرهای حالت

از روی پاسخ مشخص است که زمان نشست، جهش، و زمان خیز رضایت بخش است ولی خطای حالت ماندگار زیاد بوده و برای جبران آن از ورودی مرجع استفاده می‌کنیم.

۶-۶-۴ ورودی مرجع

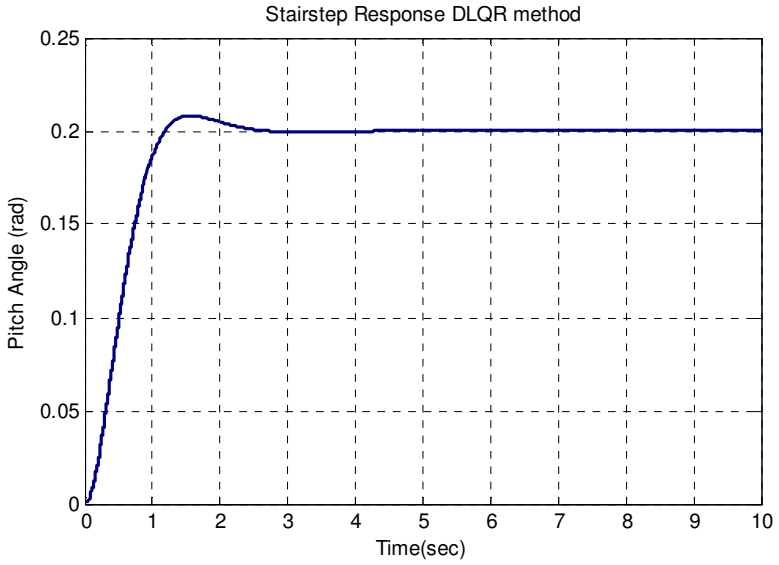
متمایز از بقیه روش‌های طراحی دیگر، در روش فیدبک تمام متغیرهای حالت، خروجی سیستم با ورودی مرجع مقایسه نمی‌شود بلکه خروجی ضرب در بهره، با ورودی مرجع مقایسه می‌گردد. برای به‌دست آوردن خروجی دلخواه باید خروجی با ورودی مرجع مساوی باشد. این کار با استفاده از ضریب پیشخور Nbar انجام می‌شود:



شکل (۶-۲۵) کنترل به روش فیدبک کلیه متغیرهای حالت با ورودی مرجع

بعد از چند بار سعی و خطا کردن مقدار Nbar را برابر 6.95 در نظر می‌گیریم.

```
t=0:0.01:10;
de=0.2*ones(size(t));
F = [0.9968              0.05649         0
    -0.0001              0.9957          0
     0                   0.5658          1];
G = [0.0024;
    0.0002;
     0.0001];
H = [0     0    1];
J = [0];
p=50;
Q = [0 0 0
     0 0 0
     0 0 p];
[K,S,E] = dlqr (F,G,Q,1)
Nbar = 6.95;
[x] = dlsim  (F-G*K,G*Nbar,H,J,de);
stairs  (t,x)
```



شکل (۶- ۲۶) پاسخ پله سیستم با فیدبک کلیه متغیرهای حالت با ورودی مرجع

از روی شکل کاملاً مشخص است که با استفاده از ورودی مرجع خطای حالت ماندگار حذف می‌شود.

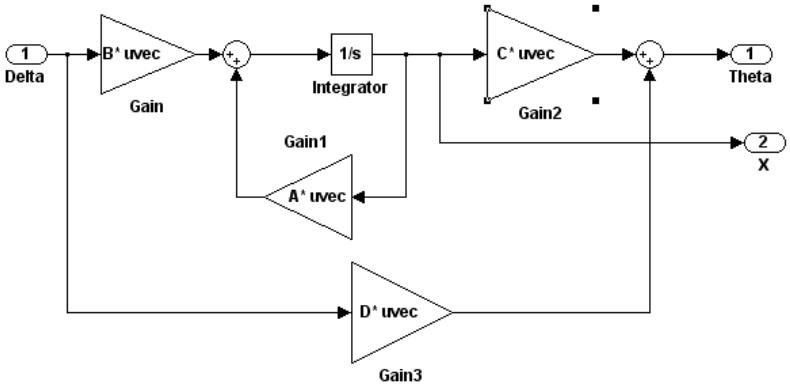
۶-۷ طراحی کنترلر به روش فیدبک کلیه متغیرهای حالت در محیط سیمولینک

همانطور که قبلاً گفته شد معادلات دینامیک هواپیما کاملاً پیچیده بوده و شامل شش معادله غیرخطی کوپل شده است. بنابراین با اعمال یکسری فرضیات مدل را ساده‌تر می‌کنیم و یک سیستم اتوپایلوت برای کنترل پیچ هواپیما طراحی می‌کنیم. با خطی کردن سیستم غیرخطی مدل فضای حالت سیستم به صورت زیر به‌دست آمد:

$$\frac{d}{dt}\begin{bmatrix}\alpha\\ q\\ \theta\end{bmatrix}=\begin{bmatrix}-0.313 & 56.7 & 0\\ -0.013 & -0.42 & 0\\ 0 & 56.7 & 0\end{bmatrix}\begin{bmatrix}\alpha\\ q\\ \theta\end{bmatrix}+\begin{bmatrix}0.232\\ 0.02\\ 0\end{bmatrix}\delta_e$$

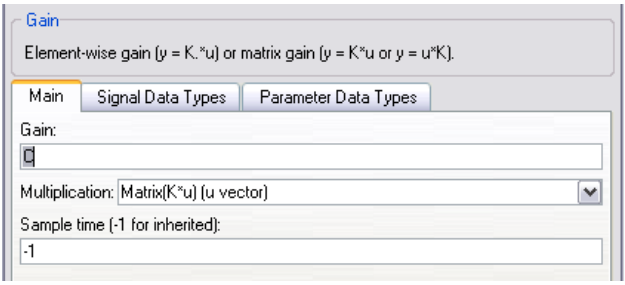
$$y=[0\quad 0\quad 1]\begin{bmatrix}\alpha\\ q\\ \theta\end{bmatrix}+[0]\delta_e$$

حال باید در محیط سیمولینک این مدل دینامیکی را بسازیم. پس مطابق شکل (۶-۲۷) بلوک‌های مورد نظر را از کتابخانه سیمولینک وارد پنجره مدل سازی کرده و با استفاده از سیگنال، آنها را به هم ارتباط دهید.

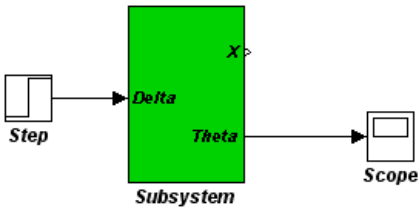


شکل (۶- ۲۷) مدل دینامیکی سیستم

روی هر یک از بهره‌ها دوبار کلیک کرده و ماتریس‌های A,B,C,D را مطابق شکل وارد نموده، تنظیمات زیر را انجام دهید.



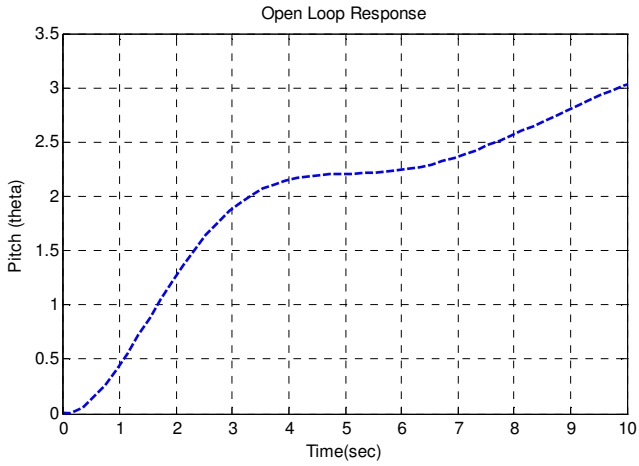
بعد از انجام مرحله مدل سازی سیستم باید پاسخ پله سیستم را به‌دست آوریم. برای انجام این کار تمام بلوک-های کشیده شده در شکل قبل را انتخاب کنید و با کلیک راست کردن بر روی صفحه و انتخاب subsystem بلوک‌ها را در داخل یک زیر سیستم قرار دهید.



سپس در پنجره دستورات MATLAB فرامین زیر را وارد کنید و مقادیر زیر را به ماتریس‌های A,B,C,D اختصاص دهید.

```
A=[-0.313 56.7 0; -0.0139 -0.426 0; 0 56.7 0];
B=[0.232; 0.0203; 0];
C=[0 0 1]; D=[0];
```

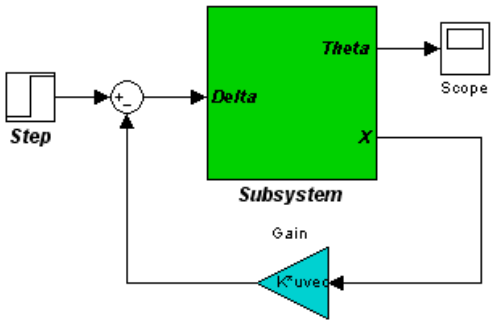
بعد از اجرای برنامه، پاسخ پله سیستم به صورت زیر به‌دست می‌آید.



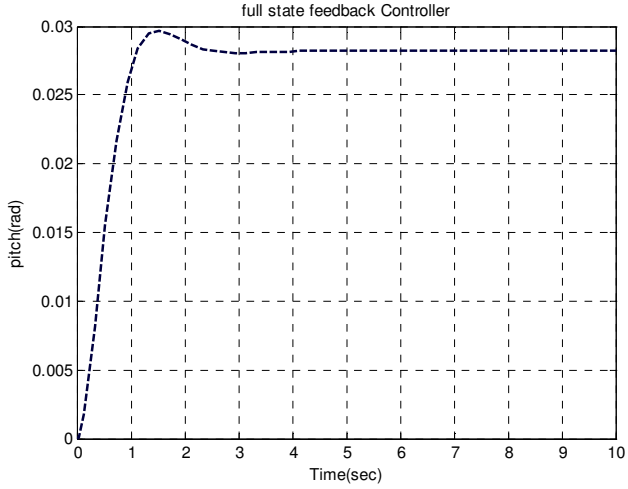
شکل (۶- ۲۸) پاسخ پله سیستم حلقه باز

همانطور که از روی شکل مشخص است سیستم ناپایدار می‌باشد. پس با استفاده از یکی از روشها نیاز داریم که یک کنترلر برای سیستم طراحی کنیم. روشی که در اینجا استفاده می‌کنیم روش فیدبک تمام متغیرهای حالت است. پس مطابق شکل زیرسیستم کنترلی حلقه بسته را تکمیل می‌کنیم. قبل از اجرای برنامه باید با استفاده از روش LQR مقدار ماتریس k را تعیین کنیم.

```
p=50;
Q=[0 0 0;0 0 0;0 0 p];
[K]= lqr (A,B,Q,1)
```

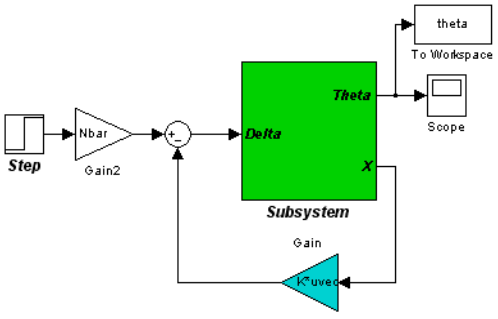


روی Step کلیک کنید و مقدار final value را برابر 0.2 وارد کنید. بعد از اجرای برنامه نمودار زیر حاصل می‌شود که نشان می‌دهد سیستم تحت کنترل است.

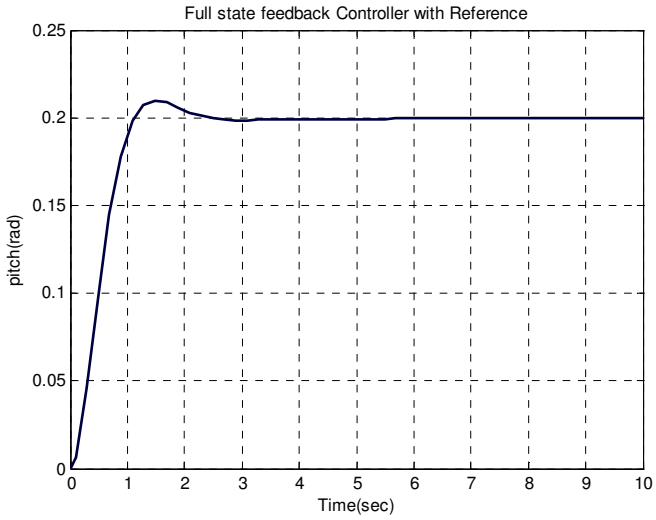


شکل (۶- ۲۹) پاسخ پله سیستم حلقه بسته به کمک روش فیدبک کلیه متغیرهای حالت

حال یک ورودی مرجع تعریف کرده و مقدار Nbar=7 را وارد می‌نمائیم.



بعد از اجرا، خروجی زیر حاصل می‌شود:



شکل (۶-۳۰) پاسخ پله سیستم حلقه بسته به کمک روش فیدبک کلیه متغیرهای حالت همراه با ورودی مرجع

پروژه هفتم

سیستم کنترل موقعیت گوی بر روی تیر

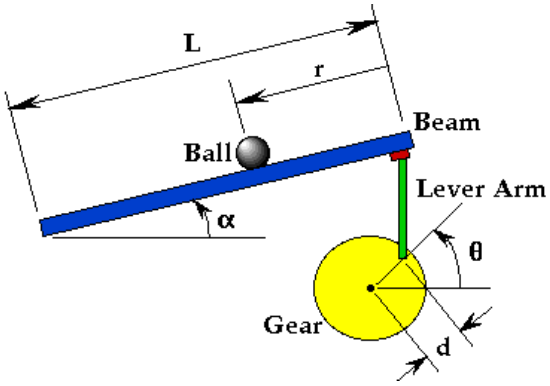
اهداف این پروژه

- مدل سازی
- طراحی کنترلر PID
- رسم مکان هندسی ریشه‌ها (Root Locus)
- پاسخ فرکانسی
- طراحی فضای حالت
- طراحی کنترلر دیجیتالی
- طراحی جبران ساز Lead در محیط سیمولینک

۷–۱ مدل سازی

۷-۱-۱ مدل سازی سیستم براساس اصول فیزیکی و استخراج معادلات سیستم

به شکل (۷-۱) توجه کنید. در این شکل یک توپ روی میله‌ای قرار گرفته است. توپ در راستای میله یک درجه آزادی دارد و در این راستا می‌تواند بغلتد. یک بازوی اهرمی[1] از یک انتها به میله متصل و از انتهای دیگر به یک چرخ دنده سرو متصل شده است. وقتی چرخ دنده سرو به اندازه θ می‌چرخد زاویه میله به اندازه α تغییر می‌کند. وقتی میله از حالت افقی خارج می‌شود گرانش باعث می‌شود توپ در راستای میله به سمت پائین بلغزد. هدف این پروژه این است که برای این سیستم، کنترلری طراحی کنیم که موقعیت توپ قابل کنترل باشد.



شکل (۷–۱) مدل فیزیکی توپ روی میله

[1] Lever Arm

در این مساله فرض بر این است که توپ در راستای میله می‌غلتد و هیچگونه لغزشی وجود ندارد و می‌توان از اصطکاک بین توپ و میله چشم‌پوشی کرد. ثابت‌ها و متغیر را برای این مساله به صورت زیر تعریف می‌کنیم:

جرم توپ : M=0.11Kg

شعاع توپ : R=0.015m

فاصله بین بازوی اهرمی تا مرکز چرخ دنده : d=0.03m

شتاب گرانشی : $g = 9.8\frac{m}{s^2}$

طول میله : L=1m

ممان اینرسی توپ : $J = 9.99\times10^{-6} kg.m^2$

موقعیت توپ : r

زاویه میله α و θ زاویه چرخدنده سرو

معادلات دینامیکی سیستم را می‌توان به صورت زیر به‌دست آورد. متغیر تعمیم یافته که حرکت سیستم را به طور کامل توصیف می‌کند r می‌باشد. موقعیت توپ را می‌توان به صورت زیر بیان کرد (دستگاه مختصاتی در بالای تیر قرار گرفته است).

$$\begin{cases} x = -rCos\alpha \\ y = -rSin\alpha \end{cases} \Rightarrow \begin{cases} \dot{x} = -\dot{r}Cos\alpha + r\dot{\alpha}.Sin\alpha \\ \dot{y} = -\dot{r}Sin\alpha - r\dot{\alpha}.Cos\alpha \end{cases} \quad (۱-۷)$$

سرعت توپ و انرژی جنبشی برابر است با:

$$V^2 = \dot{x}^2 + \dot{y}^2 = \dot{r}^2 + (r\dot{\alpha})^2$$
$$T = \frac{1}{2}mV^2 + \frac{1}{2}J\omega^2 = \frac{1}{2}m(\dot{r}^2 + (r\dot{\alpha})^2) + \frac{1}{2}J(\frac{\dot{r}}{R})^2 \quad (۲-۷)$$

انرژی پتانسیل سیستم برابر است با:

$$U = -mgrSin\alpha \quad (۳-۷)$$

از این‌رو تابع لاگرانژ سیستم عبارت‌است از:

$$Lagrange = L = T - U = \frac{1}{2}m(\dot{r}^2 + (r\dot{\alpha})^2) + \frac{1}{2}J(\frac{\dot{r}}{R})^2 + mgrSin\alpha \quad (۴-۷)$$

$$\frac{d}{dt}(\frac{\partial L}{\partial \dot{r}}) - \frac{\partial L}{\partial r} = 0 \quad (۵-۷)$$

$$\begin{cases} (m + \frac{J}{R^2})\ddot{r} - mr\dot{\alpha}^2 + mgSin\alpha = 0 \\ d.\theta = l.\alpha \end{cases} \quad (۶-۷)$$

$$(m + \frac{J}{R^2})\ddot{r} - mr(\frac{d}{l}\dot{\theta})^2 + mgSin(\frac{d}{l}\dot{\theta}) = 0 \quad (۷-۷)$$

با محاسبه ماتریس‌های ژاکوبین در نقطه کاری($\alpha = 0$)، معادلات خطی شده به صورت زیر نوشته خواهد شد:

$$(\frac{J}{R^2}+m)\ddot{r} = -mg\frac{d}{L}\theta \qquad (۸-۷)$$

۷-۱-۲ تابع تبدیل سیستم

برای به‌دست آوردن تابع تبدیل سیستم نیاز داریم از معادله مدل سازی (۷-۸) تبدیل لاپلاس بگیریم. توجه کنید زمانی که در حال پیدا کردن تابع تبدیل سیستم هستیم، شرایط اولیه را برابر صفر در نظر می‌گیریم.

$$(\frac{J}{R^2}+m)R(s).s^2 = -mg\frac{d}{L}\theta(s)$$
$$\frac{R(s)}{\theta(s)} = -\frac{mgd}{L(\frac{J}{R^2}+m)}\times\frac{1}{s^2} \qquad (۹-۷)$$

با توجه به معادله بالا متوجه می‌شویم که سیستم دو تا انتگرال‌گیر دارد.

۷-۱-۳ مدل فضای حالت سیستم

معادلات خطی شده را می‌توان به فرم فضای حالت نوشت. در این مساله $r, \dot{r}$ را به عنوان متغیرهای حالت و θ را به عنوان ورودی در نظر می‌گیریم، بنابراین داریم:

$$\begin{cases}\dot{x}(t) = Ax(t) + Bu(t)\\ y(t) = Cx(t) + Du(t)\end{cases} \Rightarrow \begin{cases}\begin{bmatrix}\dot{r}\\ \ddot{r}\end{bmatrix} = \begin{bmatrix}0 & 1\\ 0 & 0\end{bmatrix}\begin{bmatrix}r\\ \dot{r}\end{bmatrix} + \begin{bmatrix}0\\ \dfrac{mgd}{L(\frac{J}{R^2}+m)}\end{bmatrix}\theta\\ y = [1 \quad 0]\begin{bmatrix}r\\ \dot{r}\end{bmatrix}\end{cases} \qquad (۱۰-۷)$$

به هر حال برای این سیستم می‌توان مدل دیگری تعریف نموده، $r, \dot{r}, \alpha, \dot{\alpha}$ را به عنوان متغیرهای حالت معرفی کرد و برای کنترل این سیستم از گشتاور $\mathbf{u}$ استفاده نمود.

$$\begin{cases}\dot{x}(t) = Ax(t) + Bu(t)\\ y(t) = Cx(t) + Du(t)\end{cases}$$
$$\begin{cases}\begin{bmatrix}\dot{r}\\ \ddot{r}\\ \dot{\alpha}\\ \ddot{\alpha}\end{bmatrix} = \begin{bmatrix}0 & 1 & 0 & 0\\ 0 & 0 & \dfrac{-mg}{L(\frac{J}{R^2}+m)} & 0\\ 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0\end{bmatrix}\begin{bmatrix}r\\ \dot{r}\\ \alpha\\ \dot{\alpha}\end{bmatrix} + \begin{bmatrix}0\\ 0\\ 0\\ 1\end{bmatrix}u\\ y = [1 \quad 0 \quad 0 \quad 0]\begin{bmatrix}r\\ \dot{r}\\ \alpha\\ \dot{\alpha}\end{bmatrix}\end{cases} \qquad (۱۱-۷)$$

۷-۱-۴ قیدهای حاکم بر طراحی

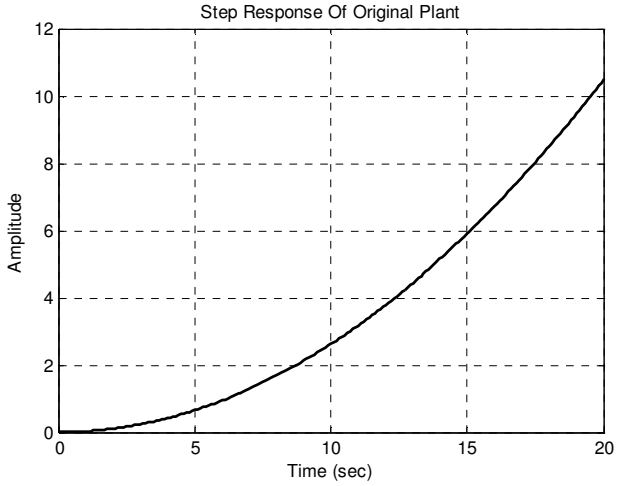
قدم بعدی در مدل‌سازی سیستم این است که بدانیم این سیستم قرار است چه شرایطی داشته باشد، یعنی قیدهای حاکم بر سیستم باید مشخص شود.

- زمان نشست کمتر از 3 ثانیه باشد.
- مقدار جهش کمتر از 5% باشد.

۷-۱-۵ پاسخ سیستم حلقه باز

دستورات زیر را در یک m-file جدید بنویسید:

```
          %~~~~~~~~~~open Loop ball & beam~~~~~~~~
m = 0.111;R = 0.015; g = -9.8;L = 1.0; d = 0.03;J = 9.99e-6;
K = (m*g*d)/(L*(J/R^2+m));   %simplifies input
num = [-K]; den = [1 0 0];
ball=tf(num,den)
step(0.25*ball)
%state space
H = -m*g/(J/(R^2)+m);
A=[0 1 0 0;0 0 H 0;0 0 0 1;0 0 0 0];
B=[0;0;0;1];
C=[1 0 0 0];
D=[0];
ball=ss(A,B,C,D);
step(0.25*ball)
```



شکل (۷-۲) پاسخ پله سیستم حلقه باز

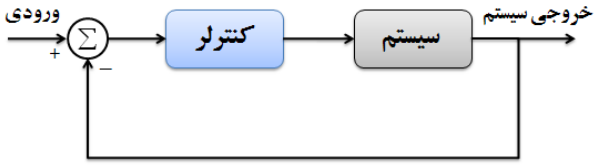
از روی شکل واضح است که سیستم کاملاً ناپایدار است، یعنی در اثر ورودی پله، توپ به انتهای سمت راست حرکت کرده و می‌افتد. پس برای موقعیت دهی توپ به کنترلر نیاز داریم.

۷-۲ طراحی کنترلر PID

همانطور که در بخش اول بررسی شد تابع تبدیل سیستم به صورت زیر به‌دست می‌آید:

$$\frac{R(s)}{\theta(s)} = -\frac{mgd}{L(\frac{J}{R^2}+m)} \times \frac{1}{s^2}$$

و همانطور که قبلاً گفته شده، هدف ما این است که زمان نشستی کمتر از 3 ثانیه و جهشی کمتر از 5% داشته باشیم. بلوک نمودار سیستم کنترلی حلقه بسته با فیدبک واحد برای کنترل موقعیت توپ به صورت زیر می‌باشد:



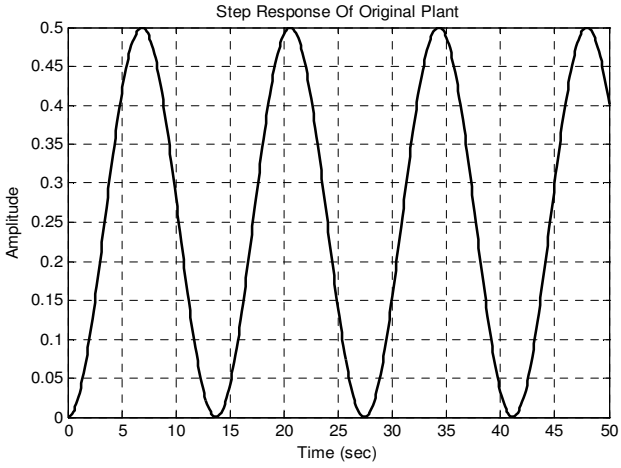
شکل (۷-۳) بلوک نمودار سیستم کنترلی با فیدبک واحد

در ابتدای کار پاسخ سیستم بالا را با کنترلر تناسبی به‌دست می‌آوریم. به طور مثال مقدار بهره را برابر $K_p = 1$ قرار می‌دهیم. تابع تبدیل سیستم با نوشتن دستورات زیر در یک m-file جدید حاصل می‌شود:

```
m = 0.111;R = 0.015;g = -9.8;L = 1.0;d = 0.03;J = 9.99e-6;
K = (m*g*d)/(L*(J/R^2+m));
num = [-K]; den = [1 0 0]; kp = 1;
numP = kp*num;
[numc, denc] = cloop(numP, den)
```

برای مدل سازی سیستم به ورودی پله 0.25m دستورات زیر را به ادامه m-file اضافه کنید.

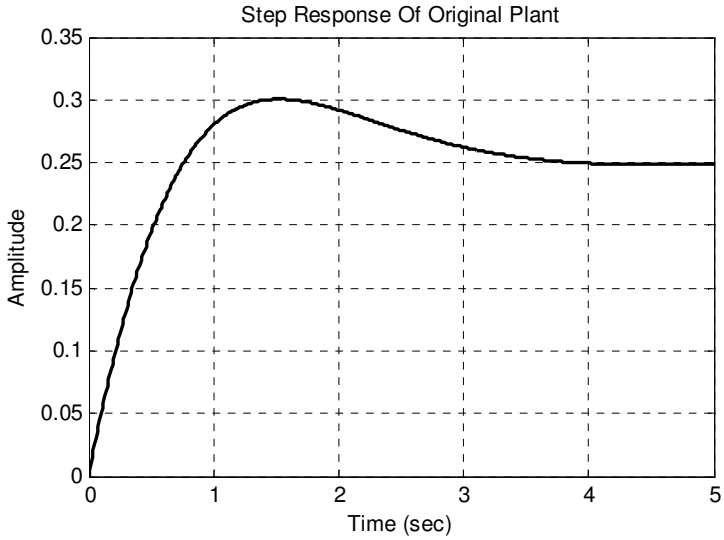
```
step(0.25*numc,denc);
axis([0 50 0 0.5]);
```



شکل (۷-۴) پاسخ پله سیستم همراه با کنترل تناسبی

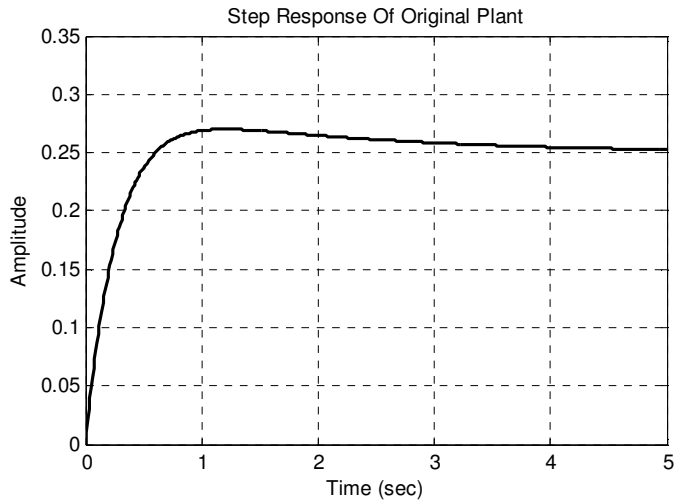
همانطور که از شکل (۷-۴) بر می‌آید می‌بینیم که بهره تناسبی به تنهایی قادر به پایدارسازی توپ روی میله نیست. مقادیر Kp را تغییر دهید تا ببینید که سیستم به هیچ وجه با کنترلر تناسبی پایدار نمی‌گردد. فقط با افزایش Kp فرکانس نوسانات بیشتر می‌شود. برای حل این مشکل از کنترلر PD استفاده می‌کنیم و پاسخ سیستم را با این کنترلر بررسی می‌کنیم. پس دستورات زیر را در یک m-file جدید وارد کنید و برنامه را اجرا نمایید.

```
m = 0.111;R = 0.015;g = -9.8;L = 1.0;d = 0.03;J = 9.99e-6;
K = (m*g*d)/(L*(J/R^2+m));   %simplifies input
num = [-K];den = [1 0 0];
kp = 1۰;kd = 10;
numPD = [kd kp];
numh = conv(num, numPD);
[numc, denc] = cloop(numh, den);
t=0:0.01:5;
step(0.25*numc,denc,t)
```



شکل (۷- ۵) پاسخ پله سیستم همراه با کنترلر PD

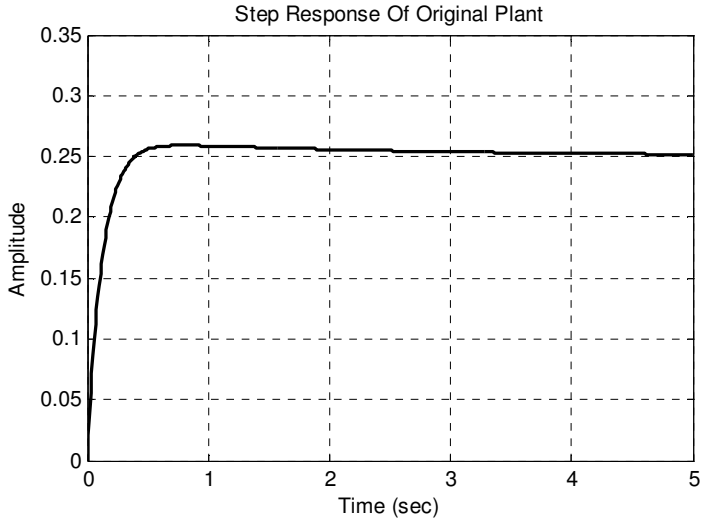
در این حالت سیستم پایدار است ولی مقدار جهش خیلی زیاد بوده و مقدار زمان نشست نیز باید تا حدودی کاهش پیدا کند. می‌توانیم با اضافه کردن بهره مشتق‌گیر هر دو مقدار را کاهش دهیم. پس مقدار آن را برابر $K_d = 20$ قرار دهید.



شکل (۷– ۶) پاسخ پله سیستم همراه با کنترلر PD با اضافه کردن بهره مشتق‌گیر

در این حالت مقدار جهش مناسب است ولی مقدار نشست باز هم باید کاهش پیدا کند. برای کاهش زمان نشست می‌توان مقدار بهره K_p را افزایش داد. پس مقادیر بهره‌ها را به صورت زیر تنظیم کنید:

$k_p = 15, k_d = 40$



شکل (۷–۷) پاسخ پله سیستم همراه با کنترلر PD با اضافه کردن بهره مشتق‌گیر و تناسبی

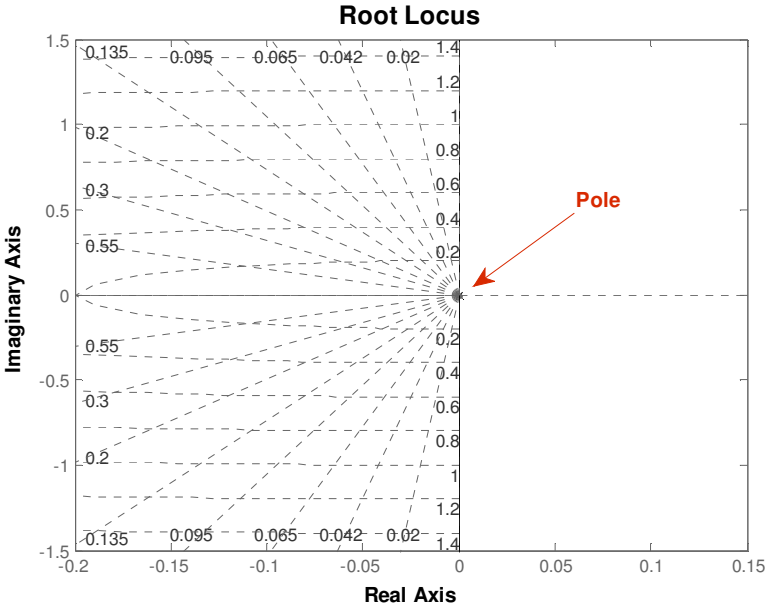
همانطور که در شکل دیده می‌شود، بدون کنترلر انتگرال‌گیر تمام معیارهای طراحی برآورده می‌شوند.

۷-۳ رسم مکان هندسی ریشه‌ها (Root Locus)

ایده اصلی در طراحی به کمک مکان هندسی ریشه‌ها، پیدا کردن پاسخ سیستم حلقه بسته از روی نمودار مکان هندسی ریشه‌های سیستم حلقه باز می‌باشد، به طوری که با اضافه کردن یک سری صفر و قطب به سیستم اصلی، پاسخ سیستم حلقه بسته تصحیح شود. پس فرامین زیر را در یک m-file جدید بنویسید:

```
m = 0.111;R = 0.015;g = -9.8;
L = 1.0;d = 0.03;J = 9.99e-6;
K = (m*g*d)/(L*(J/R^2+m));   %simplifies input
num = [-K];
den = [1 0 0];
rlocus(num,den)
```

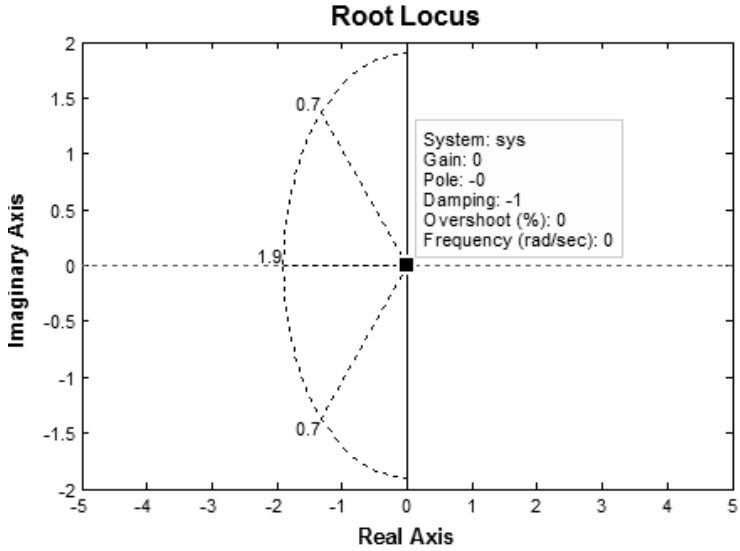
با اجرای دستورات بالا به نمودار زیر می‌رسید:



شکل (۷- ۸) مکان هندسی ریشه‌ها

همانطور که از شکل مشخص است سیستم دو قطب در مبدا مختصات دارد که در راستای محور موهومی به سمت بینهایت می‌روند. از آنجائی‌که همه قطب‌ها پذیرفتنی نیستند بنابراین از دستور sgrid برای پیدا کردن ناحیه مطلوب استفاده می‌نمائیم. با دستور sgrid می‌توانید خطوط نسبت میرایی ثابت و فرکانس طبیعی را رسم کنید. این دستور به دو آرگومان نیاز دارد: نسبت میرایی $\zeta = 0.7$ که متناسب با جهش 5% است و فرکانس طبیعی $\omega_n = 1.9$. پس دستورات زیر را در ادامه اضافه کنید:

```
sgrid(0.70, 1.9)
axis([-5 5 -2 2])
```



شکل (۷-۹) مکان هندسی ریشه‌ها

ناحیه بین دوخط چین، ناحیه‌ای است که مقدار جهش کمتر از 5% است و ناحیه خارج منحنی، ناحیه‌ای می-باشد که زمان نشست کمتر از 3 ثانیه است و همانطور که می‌بینیم هیچ قسمتی از مکان هندسی ریشه‌ها در این ناحیه قرار نگرفته است. پس برای جبران اثر آن از یک کنترلر Lead استفاده می‌کنیم تا مکان هندسی ریشه‌ها را به سمت چپ بکشاند و سیستم پایدار گردد.

۷-۳-۱ طراحی کنترلر lead

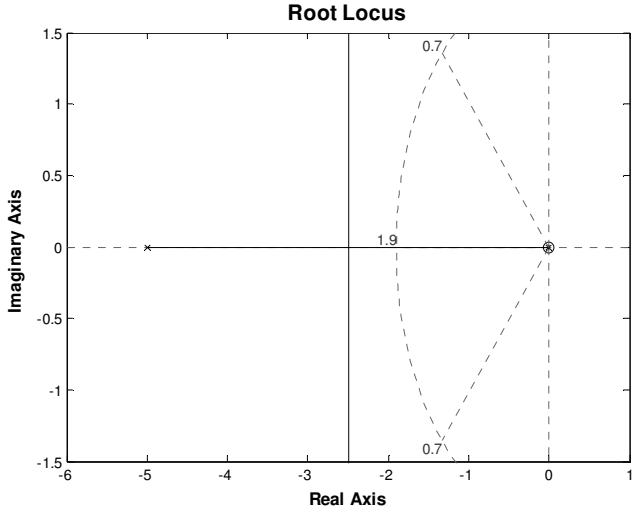
جبران کننده Lead مرتبه اول با استفاده از مکان هندسی ریشه‌ها نیز طراحی می‌شود. این جبران کننده به فرم مکان هندسی ریشه‌ها به صورت زیر می‌باشد:

$$G(s) = K_c \frac{(s - z_0)}{(s - p_0)} \qquad (۱۲-۷)$$

که بزرگی Z_0 کوچکتر از p_0 است. جبران کننده Lead باعث کشیده شدن مکان هندسی ریشه‌ها به سمت نیم صفحه چپ می‌گردد. این نتیجه باعث افزایش و بهبود پایداری سیستم و افزایش پاسخ سرعت سیستم می‌گردد. حال این کنترلر را به سیستم اضافه می‌کنیم تا تاثیر آن را در مکان هندسی ریشه‌ها ببینیم. صفر کنترلر را نزدیک یکی از قطب‌ها قرار می‌دهیم تا اثر آن را از بین ببرد و قطب این کنترلر را در سمت چپ مبدا قرار می‌دهیم تا مکان هندسی را به سمت چپ بکشد. بنابراین دستورات زیر را به ادامه m-file اضافه کنید:

```
zo = 0.01; po = 5;
numlead = [1 zo]; denlead = [1 po];
numl = conv(num,numlead);
denl = conv(den,denlead);
rlocus(numl,denl)
sgrid(0.70, 1.9)
```

با اجرای دستورات بالا به نمودار زیر می‌رسید:



شکل (۷-۱۰) مکان هندسی ریشه‌ها با جبران ساز پیش فاز

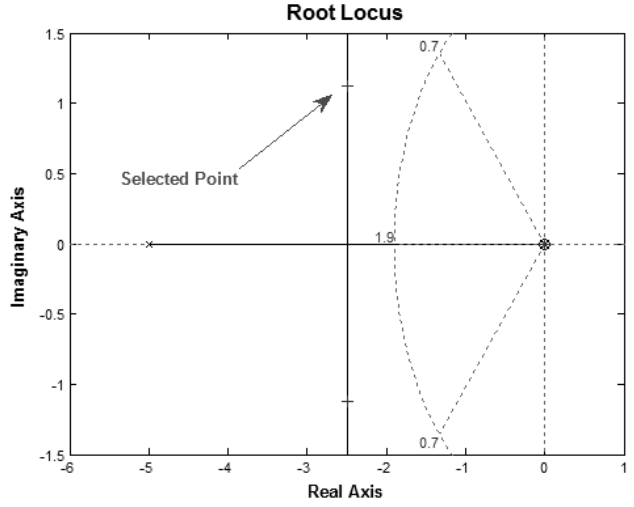
۷-۳-۲ انتخاب بهره

تاکنون مکان هندسی ریشه‌ها به سمت چپ انتقال پیدا کرده است، حال باید بهره‌ای را انتخاب کنیم که ملزومات را برآورده سازد. برای انجام این کار از دستور rlocfind استفاده می‌کنیم. پس دستور زیر را به ادامه m-file اضافه کنید:

```
[kc,poles]=rlocfind(numl,denl)
```

حال از روی نمودار نقطه زیر را انتخاب کنید:

```
selected_point =
  -2.4914 + 1.1649i
```



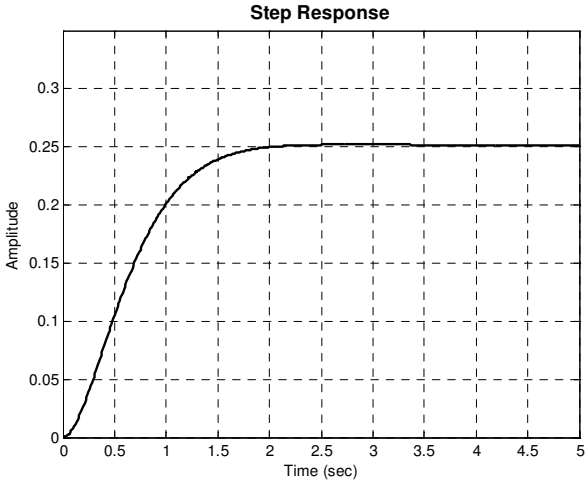
شکل (۷-۱۱) مکان هندسی ریشه‌ها با بهره انتخابی

۷-۳-۳ رسم پاسخ حلقه بسته

این مقدار kc را می‌توان در سیستم قرار داد و پاسخ سیستم را به ورودی پله 0.25m تعیین کرد. پس دستورات زیر را به ادامه m-file اضافه کنید:

```
numl2 = kc*numl;
[numcl,dencl] = cloop(numl2,denl);
t=0:0.01:5;
figure
step(0.25*numcl,dencl,t)
```

با اجرای دستورات بالا به نمودار زیر می‌رسید:



شکل (۷-۱۲) پاسخ پله سیستم حلقه بسته

از روی شکل مشخص است که تمام معیارها مثل زمان نشست و مقدار جهش برآورده شده اند.

۷-۴ پاسخ فرکانسی

تابع تبدیل حلقه باز سیستم به صورت زیر می‌باشد:

$$\frac{R(s)}{\theta(s)} = -\frac{mgd}{L(\frac{J}{R^2}+m)} \times \frac{1}{s^2}$$

همان طور که قبلاً گفته شد معیارهای طراحی ما رسیدن به زمان نشستی کمتر از 3 ثانیه و جهشی کمتر از 5% می‌باشد.

۷-۴-۱ نمودار بود سیستم حلقه باز

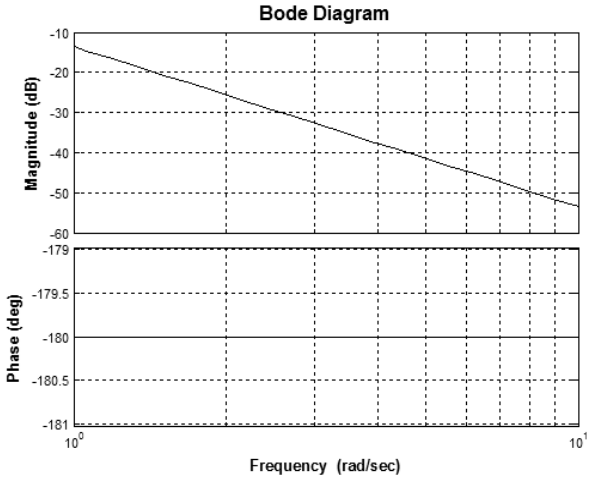
ایده اصلی طراحی برمبنای پاسخ فرکانسی بر این است که با رسم نمودار بود[2] تابع تبدیل حلقه باز، تخمینی از پاسخ سیستم حلقه بسته به‌دست آید. اضافه کردن کنترلر به سیستم، نمودار بود حلقه باز را تغییر می‌دهد.

[2] Bode diagram

بنابراین پاسخ سیستم حلقه بسته را نیز تغییر می‌دهد. اجازه دهید که در ابتدا نمودار بود تابع تبدیل حلقه باز را رسم کنیم. پس تابع تبدیل را در یک m-file وارد کنید:

```
m = 0.111;R = 0.015;g = -9.8;L = 1.0;d = 0.03;J = 9.99e-6;
K = (m*g*d)/(L*(J/R^2+m));
num = [-K]; den = [1 0 0];
bode(num,den)
```

در نتیجه چنین نموداری حاصل می‌شود:



شکل (۷-۱۳) نمودار بود سیستم حلقه باز

از روی نمودار مشخص است که حد فاز صفر است، پس سیستم ناپایدار می‌باشد. طبق تعریف، حد فاز میزان تغییراتی است که می‌توانیم در فاز سیستم حلقه باز انجام دهیم تا سیستم حلقه بسته همچنان پایدار باقی بماند. حال می‌خواهیم برای پایدار کردن سیستم، حد فاز را افزایش دهیم. بنابراین از یک جبران ساز Lead برای رسیدن به این هدف استفاده می‌کنیم.

۷-۴-۲ جبران کننده پیش فاز

جبران کننده پیش فاز مرتبه اول به فرم زیر می‌باشد:

$$G(s) = k\frac{1+Ts}{1+aTs} \qquad (۷-۱۳)$$

جبران کننده Lead یک فاز مثبت در بازه فرکانسی $\frac{1}{T}$ تا $\frac{1}{aT}$ به سیستم اضافه می‌کند که در آن فرکانس‌های $\frac{1}{T}$ و $\frac{1}{aT}$ را فرکانس‌های گوشه می‌گویند. حداکثر فازی که با استفاده از یک جبران کننده Lead می‌توانیم اضافه کنیم 90 درجه است. در طراحی کنترلر نیاز داریم به جهش کمتر از 5% برسیم که این جهش متناظر با $\zeta = 0.7$ است. به طورعمومی $100\times\zeta$ ، حداقل فازی را که نیاز داریم به جهش مورد نظر برسیم را به ما می‌دهد. پس ما به حد فاز بزرگتر از 70 درجه نیاز داریم.

برای به‌دست آوردن a,T نیاز است که کارهای زیر را انجام دهیم:

۱. از طراحی کنترلر نیاز به حداقل 70 درجه فاز داریم.

۲. تعیین فرکانس مرکز، یعنی فرکانسی که فاز باید به آن اضافه شود. در این نمونه چون نمودار فاز بر حسب فرکانس یک خط صاف است پیدا کردن این نقطه مشکل است. به هر حال ما رابطه بین پهنای فرکانسی و زمان نشست را داریم و این نمودار به ما می‌گوید که wbw=1.92rad/s. برای این کار نیاز به فرکانس مرکز داریم، که مقدار آن را 1rad/s در نظر می‌گیریم.

۳. پیدا کردن ثابت a از رابطه زیر:

$$a=\frac{1-\sin\varphi}{1+\sin\varphi}$$

این رابطه فضای لازم بین صفر و قطب را تعیین می‌کند تا اینکه حداکثر فاز اضافه شود. برای فاز مطلوب:

$$\varphi=70\Rightarrow a=0.0311$$

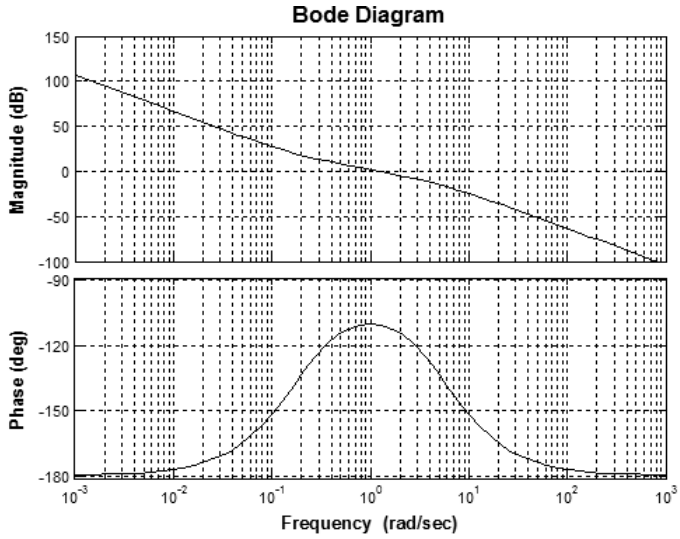
۴. تعیین T و At از معادله زیر:

$$\begin{cases}T=\dfrac{1}{w\sqrt{a}}\Rightarrow aT=\dfrac{\sqrt{a}}{w}\\ w=1, a=0.0311\end{cases}\quad\Rightarrow T=5.67\ ,\ aT=0.176$$

حالا می‌توانیم جبران کننده Lead را به سیستم اضافه نموده و نمودار بود حاصل را مشاهده کنیم. پس دستورات زیر را به ادامه m-file اضافه کنید:

```
k=1;
numlead = k*[5.67 1]; denlead = [0.176 1];
numl = conv(num,numlead);denl = conv(den,denlead);
bode(numl,denl)
```

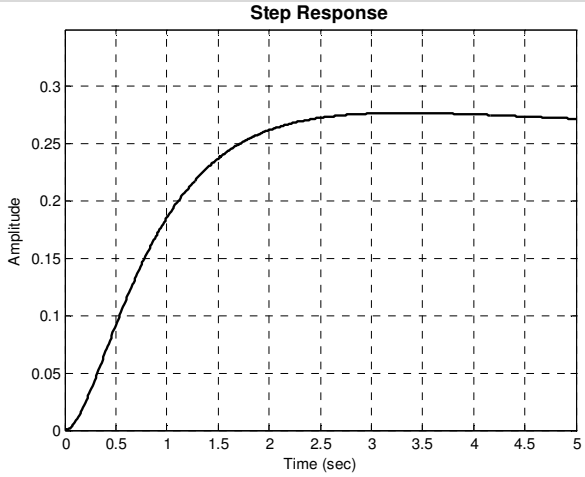
نمودار بود به صورت زیر خواهد بود:



شکل (۷-۱۴) نمودار بود سیستم با جبران ساز Lead

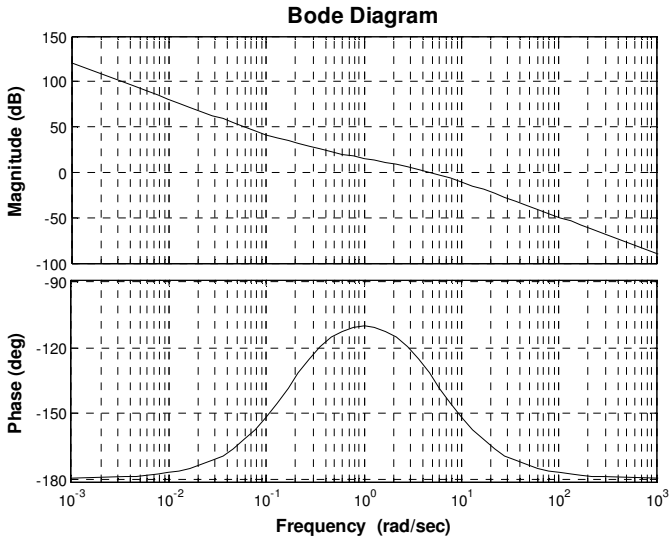
از روی شکل مشخص است که حاشیه فاز حدود 70 درجه است. حال پاسخ سیستم حلقه بسته را به ورودی پله بررسی می‌کنیم. پس دستورات زیر را به ادامه m-file اضافه کنید:

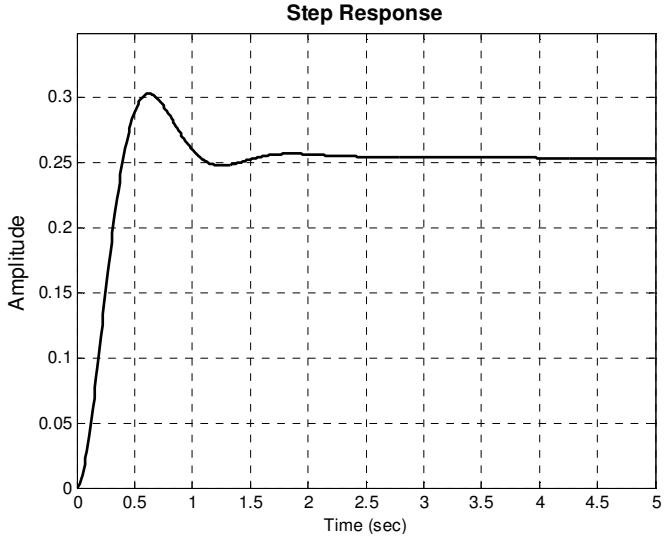
```
[numcl,dencl] = cloop(numl,denl);
t=0:0.01:5;
step(0.25*numcl,dencl,t)
```



شکل (۷-۱۵) پاسخ پله سیستم حلقه بسته

هرچند در این حالت سیستم پایدار شده و جهشی کمتر از 5% داریم ولی زمان نشست رضایت بخش نیست. اضافه کردن بهره باعث زیاد شدن فرکانس w_g می‌شود و پاسخ را سریعتر می‌کند. حال k را برابر 5 قرار می‌دهیم. پاسخ به صورت زیر خواهد بود:





شکل (۷-۱۶) نمودار بود سیستم با جبران ساز Lead و پاسخ پله سیستم حلقه بسته با k=5

در این حالت نیز جهش خیلی زیاد می‌شود، زیاد کردن بهره باعث زیاد شدن جهش می‌گردد.

۷-۴-۳ اضافه کردن فازهای بیشتر

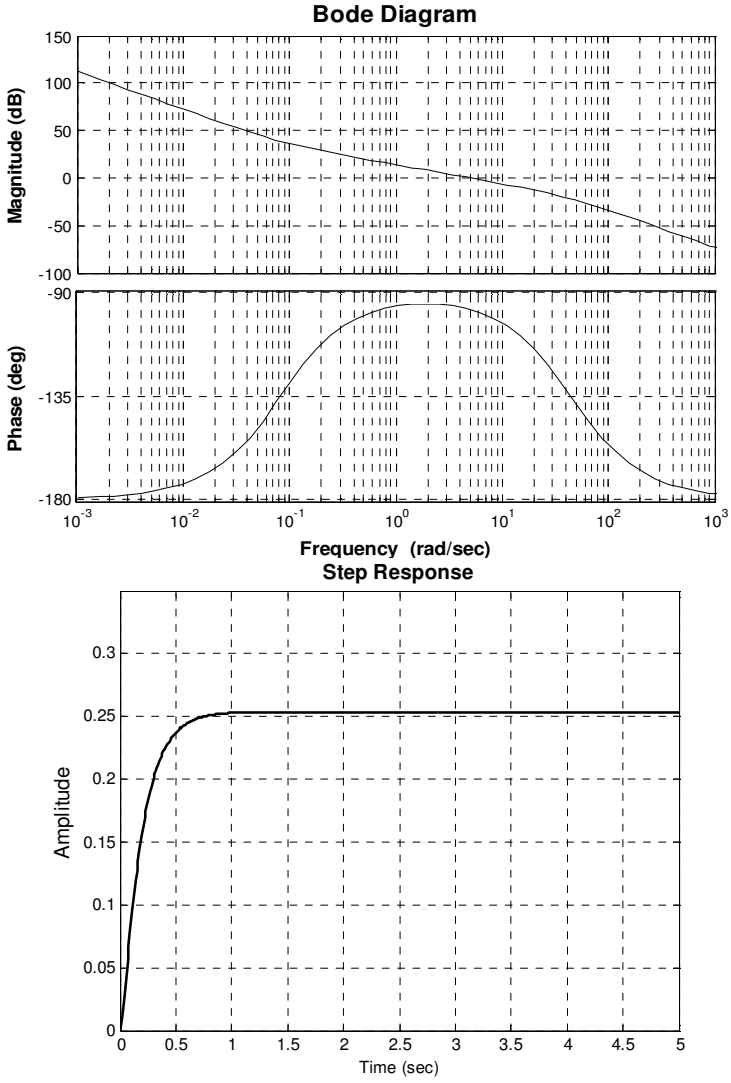
برای کاهش جهش می‌توانیم فاز جبران کننده Lead را افزایش دهیم تا مقدار جهش کاهش پیدا کند. برای راحتی کار از برنامه زیر استفاده کنید:

```
m = 0.111;R = 0.015;g = -9.8;L = 1.0;d = 0.03;J = 9.99e-6;
K = (m*g*d)/(L*(J/R^2+m));
num = [-K]; den = [1 0 0];

%ask user for controller information
pm = input('Phase Margin?.......');
w  = input('Center Frequency?...');
k  = input('Gain?..............');

%view compensated system bode plot
pmr = pm*pi/180;
a = (1 - sin(pmr))/(1+sin(pmr)); T = sqrt(a)/w;
At = 1/(w*sqrt(a));
numlead = k*[At 1]; denlead = [T 1];
numl=conv(num,numlead);
denl=conv(den,denlead);
figure
bode(numl,denl)
%view step response
[numcl,dencl]=cloop(numl,denl);
```

```
t=0:0.01:5;
figure
step(0.25*numcl,dencl,t)
```

با استفاده از این برنامه شما می‌توانید حد فاز و فرکانس مرکز و بهره را انتخاب کنید. به طور مثال مقادیر زیر را وارد کنید:

```
Phase Margin?.......85
Center Frequency?...1.9
Gain?..............2
```



شکل (۷– ۱۷) نمودار بود سیستم و پاسخ پله سیستم حلقه بسته با اضافه کردن فاز به جبران ساز Lead

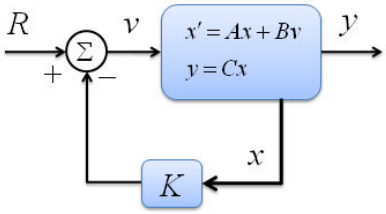
۷-۵ طراحی فضای حالت

معادله فضای حالت سیستم از روی معادلات دینامیکی به صورت زیر به‌دست می‌آید:

$$\begin{bmatrix} \dot{r} \\ \ddot{r} \\ \dot{\alpha} \\ \ddot{\alpha} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & \dfrac{-m\,g}{L(\frac{J}{R^2}+m)} & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} r \\ \dot{r} \\ \alpha \\ \dot{\alpha} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} u$$

$$y = [1 \quad 0 \quad 0 \quad 0] \begin{bmatrix} r \\ \dot{r} \\ \alpha \\ \dot{\alpha} \end{bmatrix}$$

با اعمال گشتاور به وسط تیر می‌توانیم موقعیت توپ را روی میله کنترل کنیم و دیگر نیازی به چرخ دنده و اهرم نداریم. کنترلر با فیدبک تمام متغیرهای حالت در شکل(۷-۱۸) نمایش داده شده است.



شکل (۷- ۱۸) کنترل به روش فیدبک کلیه متغیرهای حالت

معادله مشخصه برای سیستم حلقه بسته از رابطه $\det(sI-(A-BK))=0$ به‌دست می‌آید، که در آن s متغیر حوزه لاپلاس است. ماتریس‌های A,BK هر دو 4×4 می‌باشند. پس این سیستم 4 قطب دارد و می‌توانیم قطب‌ها را به هر نقطه دلخواهی جابه‌جا کنیم. برای طراحی این کنترلر نیاز داریم جهش کمتر از 5% شود که معادل $\zeta=0.7$ است یا همان خط با زاویه 45° در مکان هندسی ریشه‌ها که از نیم صفحه چپ می‌گذرد. می‌خواهیم قطب‌ها روی این خط یا نزدیک این خط باشند، از طرفی زمان نشست باید کمتر از سه ثانیه باشد که آن نیز متناظر است با $\zeta w_n = \frac{4.6}{T_s} = \frac{4.6}{3} = 1.53$ که با خط عمودی 1.53- در مکان هندسی ریشه‌ها قابل نمایش است. قطب‌ها باید در سمت چپ این محور قرار بگیرند، پس قطب‌ها را در نقاط زیر قرار دهید:

$$P = -2 \pm 2i, -20, -80$$

دو قطب را در 80- ,20- قرار می‌دهیم تا اثر زیادی روی جواب نداشته باشد. پس دستورات زیر را در یک m-file وارد کنید:

```
m = 0.111;R = 0.015;g = -9.8;J = 9.99e-6;
H = -m*g/(J/(R^2)+m);
A=[0 1 0 0
```

```
   0 0 H 0
   0 0 0 1
  0 0 0 0];
B=[0;0;0;1];C=[1 0 0 0];D=[0];
p1=-2+2i;p2=-2-2i;
p3=-20;p4=-80;
K=place(A,B,[p1,p2,p3,p4])
```

که مقدار بهره K برابر است با:

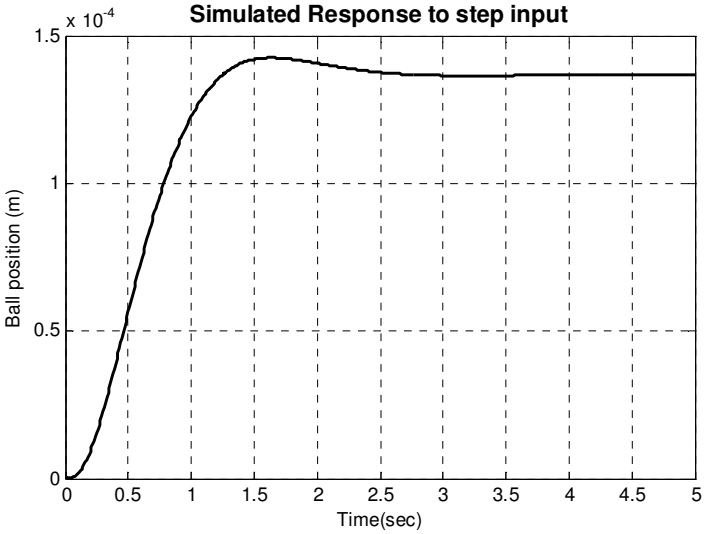
```
K =
  1.0e+003 *
    1.8286   1.0286   2.0080   0.1040
```

پس معادله فضای حالت به صورت زیر به‌دست می‌آید:

$$\begin{cases} \dot{x} = (A - BK)x + Bu \\ y = Cx \end{cases}$$

حال پاسخ سیستم حلقه بسته را به ورودی پله 0.25m بررسی کنید:

```
T = 0:0.01:5;
U = 0.25*ones(size(T));
[Y,X]=lsim(A-B*K,B,C,D,U,T);
plot(T,Y)
```

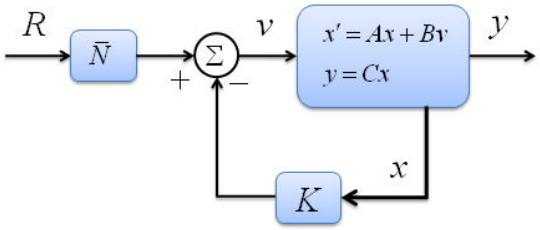


شکل (۷-۱۹) پاسخ پله سیستم با فیدبک کلیه متغیرهای حالت

از روی شکل (۷-۱۹) واضح است که خطای حالت ماندگار بسیار بالا می‌باشد. ما نیاز داریم یک ورودی مرجع به آن اضافه کنیم. در هر صورت مقدار جهش و زمان نشست راضی کننده است. اگر بخواهیم مقدار جهش را باز هم کاهش دهیم باید قسمت موهومی قطب‌های دلخواه را کوچکتر از قسمت حقیقی در نظر بگیریم و اگر می خواهیم زمان نشست کمتر شود باید قطب‌ها را بیشتر به سمت چپ بکشیم.

۷-۵-۱ اضافه کردن ورودی مرجع

برای رها شدن از خطای حالت ماندگار، باید خروجی‌های سیستم را بعد از فیدبک کردن با یک ورودی مرجع مقایسه کنیم تا مقدار خطا محاسبه شود. این کار با اضافه کردن یک بهره بعد از ورودی به‌دست می‌آید:

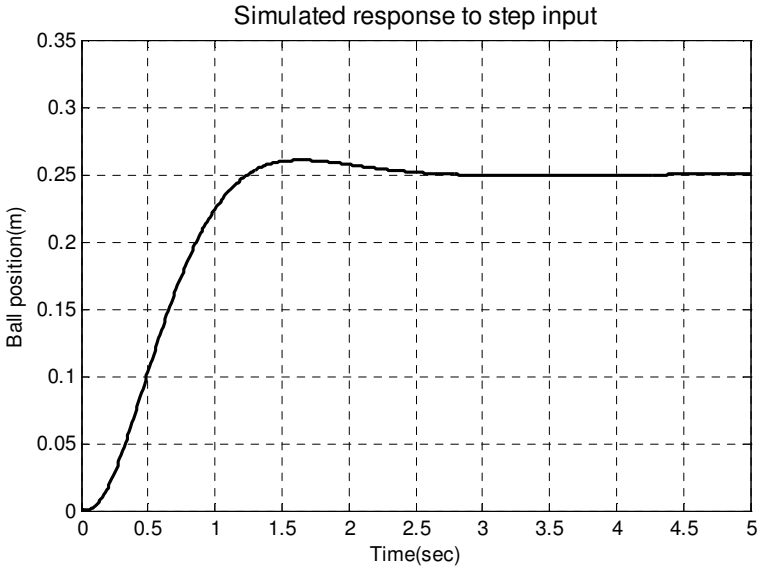


شکل (۷-۲۰) کنترل به روش فیدبک کلیه متغیرهای حالت همراه با ورودی مرجع

$$u = -kx + R\bar{N}$$
$$\dot{x} = Ax + Bu \Rightarrow \dot{x} = (A - Bk)x + BR\bar{N} \qquad (۱۴-۷)$$
$$\bar{N} = 1828$$

دستورات زیر را در m-file وارد کنید:

```
Nbar=rscale(A,B,C,D,K)
T = 0:0.01:5;
U = 0.25*ones(size(T));
[Y,X]=lsim(A-B*K,B*Nbar,C,D,U,T);
plot(T,Y)
```



شکل (۷-۲۱) پاسخ پله سیستم با فیدبک کلیه متغیرهای حالت همراه با ورودی مرجع

همان طور که از شکل(۷-۲۱) نتیجه می‌شود، تمام معیارهای طراحی ما برآورده شده است.

۷-۶ طراحی کنترلر دیجیتالی

در این قسمت هدف این است که با استفاده از روش کنترلر PID یک کنترلر دیجیتالی طراحی کنیم، به طوری که به جهش کمتر از 5% و زمان نشست کمتر از 3 ثانیه برسیم.

۷-۶-۱ کنترلر PID دیجیتالی

تابع تبدیل کنترلر PID در فضای پیوسته به صورت زیر است:

$$\text{PID Continuous}: k_p + k_D s + \frac{k_I}{s} = \frac{k_D s^2 + k_p s + k_I}{s} \qquad (۱۵-۷)$$

و در فضای گسسته به صورت زیر نوشته می‌شود:

$$\text{PID Digital}: \frac{(K_p + \frac{K_I.T_s}{2} + \frac{2K_D}{T_s}).z^2 + (K_I.T_s - \frac{4K_D}{T_s}).z + (-K_p + \frac{K_I T_s}{2} + \frac{2K_D}{T_s})}{z^2 - 1} \qquad (۱۶-۹)$$

۷-۶-۲ تابع تبدیل گسسته

اولین کاری که باید انجام دهیم این است که تابع تبدیل پیوسته را به گسسته تبدیل کنیم. بنابراین در نرم افزار MATLAB از دستور c2dm استفاده می‌کنیم. برای استفاده از این دستور به آرگومان‌های زیر نیاز داریم: صورت کسر (num)، مخرج کسر (den)، زمان نمونه برداری، و روشی که انتخاب می‌کنیم.

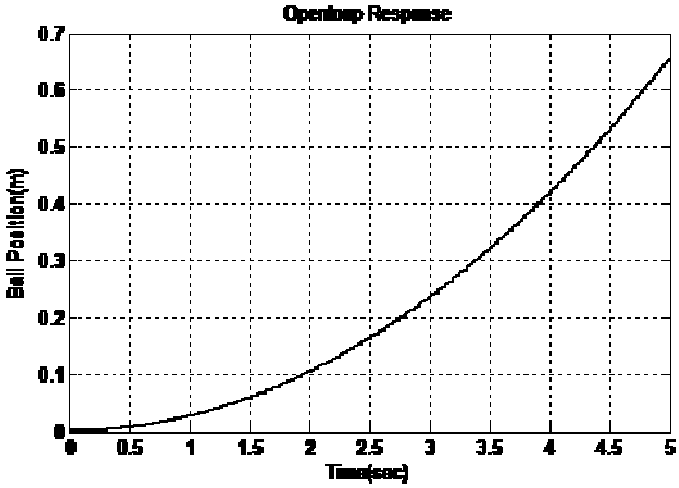
زمان نمونه‌برداری باید کوچکتر از $1/(30\times BW)$ باشد که BW پهنای باند فرکانسی سیستم حلقه بسته می‌باشد. روشی که انتخاب می‌کنیم zoh است و فرض می‌کنیم که فرکانس پهنای باند برابر 1rad/s است و زمان نمونه برداری را برابر 1/50 ثانیه در نظر می‌گیریم. حال دستورات زیر را وارد کنید:

```
m = 0.111;R = 0.015; g = -9.8;L = 1.0;d = 0.03;
J = 9.99e-6;
K = (m*g*d)/(L*(J/R^2+m));   %simplifies input
num = [-K];
den = [1 0 0];
Ts = 1/50;
[numDz,denDz]= c2dm (num,den,Ts,'zoh')
```

۷-۶-۳ پاسخ حلقه باز سیستم

حال پاسخ توپ را به ورودی پله 0.25m بررسی می‌کنیم. پس دستورات زیر را وارد کنید:

```
numDz = 0.0001*[0.42 0.42];
denDz = [1 -2 1];
[x] = dstep (0.25*numDz,denDz,251);
t=0:0.02:5;
stairs(t,x)
```



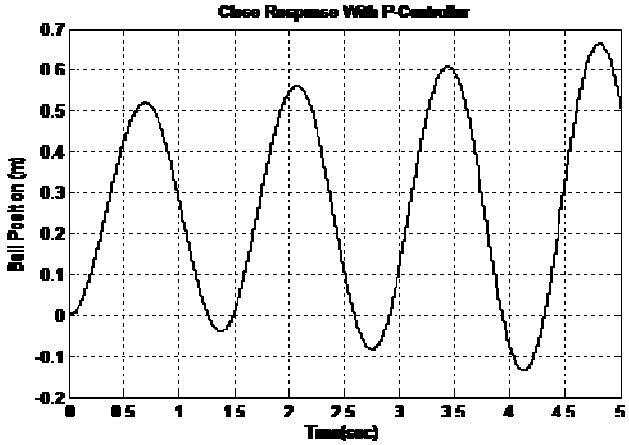
شکل (۷-۲۲) پاسخ پله سیستم حلقه باز

از نمودار می‌بینیم که سیستم حلقه باز ناپایدار است و توپ تا انتهای تیر می‌غلتد.

۷-۶-۴ کنترلر تناسبی

حال یک کنترلر تناسبی k_p به سیستم اضافه می‌کنیم و پاسخ سیستم حلقه بسته را بررسی می‌کنیم. مقدار k_p را برابر 100 قرار داده و پاسخ را بررسی می‌کنیم. دستورات زیر را به m-file اضافه می‌کنیم:

```
numDz = 0.0001*[0.42 0.42];denDz = [1 -2 1];
Kp=100;
[numDzC,denDzC]=cloop (Kp*numDz,denDz);
[x] = dstep (0.25*numDzC,denDzC,251);
t=0:0.02:5;
stairs(t,x)
```



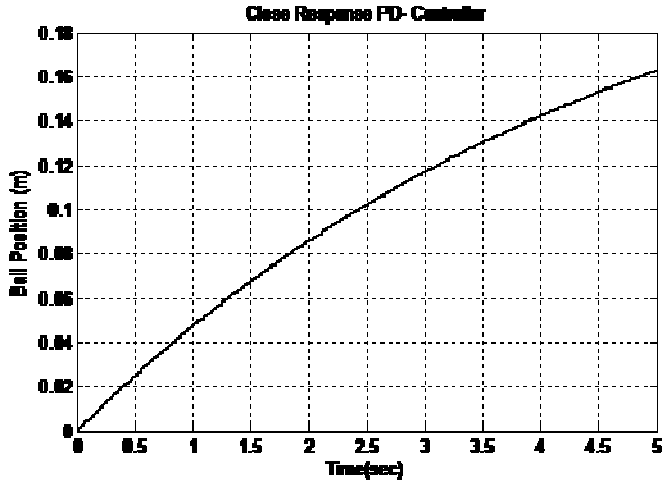
شکل (۷-۲۳) پاسخ پله سیستم با کنترلر تناسبی دیجیتالی

همانطور که از نمودار بر می‌آید کنترلر تناسبی سیستم را پایدار نمی‌سازد. مقدار k_p را افزایش دهید، ولی سیستم همچنان ناپایدار باقی می‌ماند.

۷-۶-۵ کنترلر مشتق گیر و تناسبی

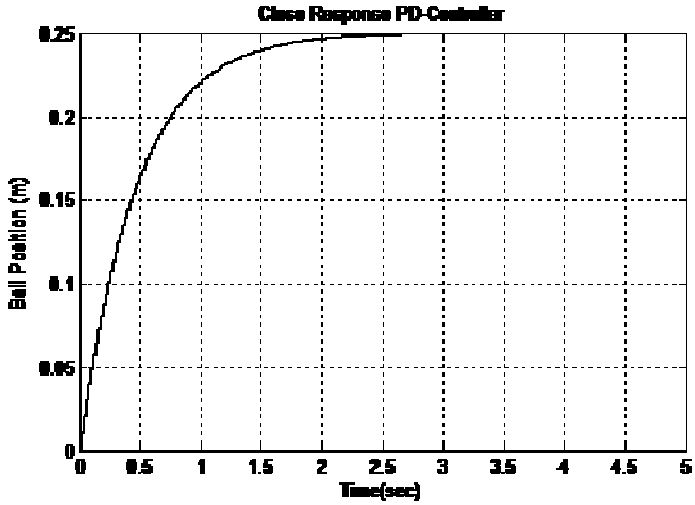
حال یک مشتق گیر به کنترلر قبلی اضافه می‌کنیم. مقدار $k_p=100,k_d=10$ را در نظر بگیرید و برنامه زیر را اجرا کنید:

```
numDz = 0.0001*[0.42 0.42];
denDz = [1 -2 1];
Kp=100;Kd=10;

numpd = [Kp+Kd -(Kp+2*Kd) Kd];
denpd = [1 1 0];

numDnew = conv(numDz,numpd);
denDnew = conv(denDz,denpd);
[numDnewC,denDnewC] = cloop(numDnew,denDnew);
[x] = dstep (0.25*numDnewC,denDnewC,251);
t=0:0.02:5;
stairs(t,x)
```



شکل (۷-۲۴) پاسخ پله سیستم با کنترلر PD دیجیتالی

در این حالت سیستم پایدار می‌گردد ولی زمان خیز طولانی می‌باشد. از کنترلرهای PID به یاد داریم که افزایش k_p باعث کاهش زمان خیز می‌گردد و بنابراین، مقدار k_p را برابر 1000 قرار می‌دهیم تا ببینیم چه اتفاقی می‌افتد.



شکل (۷-۲۵) پاسخ پله سیستم با کنترلر PD دیجیتالی با افزایش بهره K_P

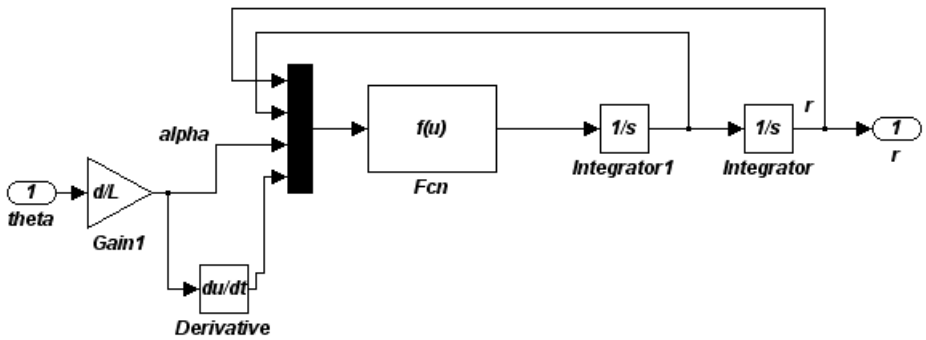
همانطور که می‌بینیم تمام ملزومات طراحی برآورده می‌شود.

۷-۷ طراحی جبران ساز پیش فاز در محیط سیمولینک

در این قسمت باید مدل ریاضی به‌دست آمده را در محیط شبیه سازی به یکسری بلوک نمودار تبدل کنیم. در ابتدا مدل دینامیکی سیستم را در محیط سیمولینک می‌سازیم و پاسخ سیستم حلقه باز را بررسی می‌کنیم. پس مطابق شکل(۷-۲۶)، از کتابخانه سیمولینک بلوک‌ها را وارد پنجره مدل کنید و مدل دینامیکی سیستم را بسازید.

$$\text{(I)}\ \ (m+\frac{J}{R^2})\ddot{r}-mr\dot{\alpha}^2+mgSin\alpha=0\Rightarrow\frac{d^2r}{dt^2}=\frac{-1}{(m+\frac{J}{R^2})}\times(-mr\dot{\alpha}^2+mgSin\alpha)$$

$$\text{(II)}\ \alpha=\frac{d}{l}.\theta$$



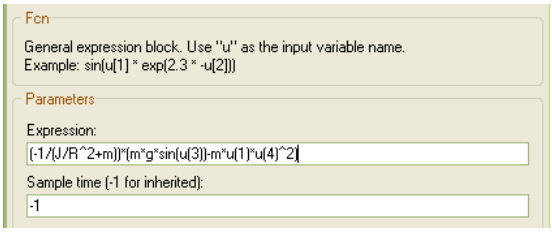
شکل (۷-۲۶) مدل دینامیکی سیستم

روی **Fcn** دوبار کلیک کنید و عبارت زیررا در قسمت Expression وارد کنید:

```
(-1/(J/(R^2)+m))*(m*g*sin(u[3])-m*u[1]*(u[4])^2)
```

در عبارت بالا هر یک پارامترهای u(i) به صورت زیر تعریف شده اند:

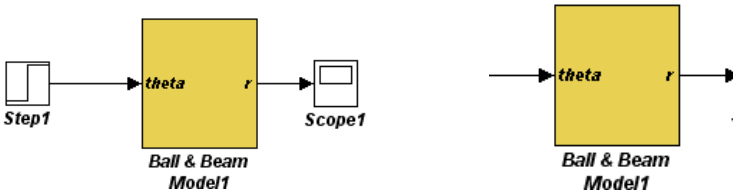
```
u[1]=r, u[2]=d/dt(r), u[3]=alpha, and u[4]=d/dt(alpha)
```



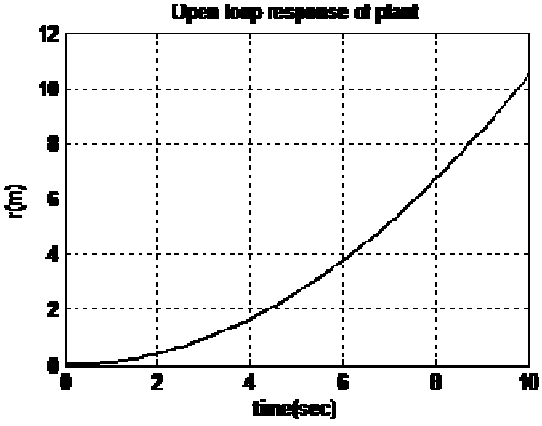
حال مقادیر زیر را در پنجره دستورات MATLAB وارد کنید:

```
m = 0.111;R = 0.015;g = -9.8;L = 1.0;d = 0.03;J = 9.99e-6;
```

سپس تمامی المان‌های کشیده شده را انتخاب کرده و روی صفحه، کلیک راست کرده و Create Subsystem را انتخاب کنید تا تمامی المان‌ها در داخل یک بلوک قرار بگیرند. از این به بعد، این بلوک نشان دهنده سیستم است. بعد از اینکه مدل سیستم را شبیه سازی کردیم باید پاسخ سیستم حلقه باز را بررسی کنیم. بنابراین در ورودی سیستم از ورودی پله و در خروجی یک Scope قرار می‌دهیم.



خروجی سیستم حلقه باز به صورت زیر خواهد بود. این پاسخ کاملاً ناپایدار است.

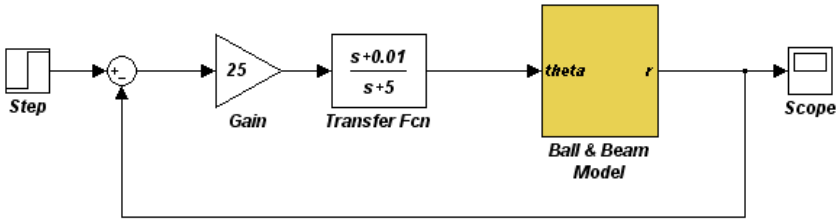


شکل (۷-۲۷) پاسخ پله سیستم حلقه باز

در این قسمت می‌خواهیم یک کنترل جبران کننده Lead برای این مدل طراحی کنیم. بنابراین مطابق شکل(۷-۲۸) سیستم کنترل حلقه بسته را طراحی می‌کنیم. روی تابع تبدیل کنترلر دوبار کلیک نموده و مقادیر زیر را وارد کنید:

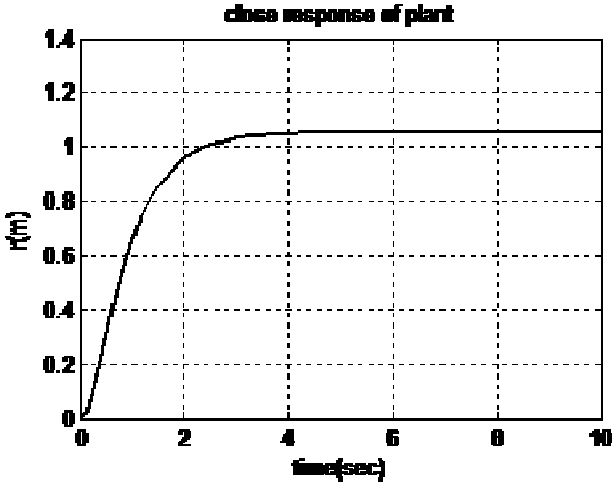
```
Num= [1 0.01];
Den=[1 5];
```

روی بلوک بهره نیز دو بار کلیک کنید و مقدار بهره را برابر 25 قرار دهید.



شکل (۷-۲۸) مدل سیستم به همراه کنترلر lead

خروجی سیستم تحت کنترل به صورت زیر به‌دست می‌آید:



شکل (۷-۲۹) پاسخ پله سیستم حلقه بسته با کنترلر Lead دیجیتالی

مراجع

1. Robert Bishop, 'Modern Control Systems Analysis and Design Using MATLAB', Addison Wesley 1997.
2. Ashish tewari, 'Modern Control Systems Design Using MATLAB and SIMULINK', Willey 2002.
3. Dawn Tilbury and Bill Messner, 'Control Tutorials for MATLAB and Simulink', University of Michigan.
4. Brian D. Hahn and Daniel T. Valentine , 'Essential MATLAB for Engineers and Scientists' , Elsevier, 2007.
5. Bernard Friedland, 'Control System Design: An Introduction to State-Space Methods', McGraw-Hill, 1987.
6. Introduction to the state-space approach of analysis and control synthesis, MITOpenCourseWare: http://ocw.mit.edu/OcwWeb.
7. J ohn J. Dazzo, Constantine H. Houpis, 'Linear Control System Analysis and Design :Conventional and Modern',McGraw Hill ,1988.
8. Howard B.Wilson, Louise H.Turcotte, 'Advanced Mathematis and Mechanics Applications with MATLAB', Chapman and Hall/CRC,2003
9. Jaan Kiusalaas, 'Numerical Methods in Engineering with MATLAB', Cambridge Universty press, 2005
10. Pierre Belanger, 'Control Engineering: A Modern Approach', Saunders College Publishing, 1995.
11. Won Young Yang, Wenwu Cao, John Morris, 'Applied Numerical Methods Using MATLAB', Wiley Interscience, 2005
12. Misza Kalechamn, 'Practical MATLAB Applications for Engineers',CRC press, 2009.

۱۳. علی خاکی صدیق ، "سیستم های کنترل خطی"، انتشارات پیام نور

۱۴. علی خاکی صدیق ، "اصول کنترل مدرن"، انتشارات دانشگاه تهران