```
%%%% MyGenetic.m %%%%  نام برنامه اصلی
clc; %  صفحه نمایش را پاک کن
clear all; %  کلیه متغیرهای موجود در حافظه را پاک کن
%%%% Initialization %%%% در این قسمت مقدار دهی اولیه پارامترهای الگوریتم انجام می‌شود
N=50; %  تعداد کروموزوم‌ها
Pc=0.9;%  نرخ همبری
Pm=0.005;%  نرخ جهش
ITER=100;%  تعداد دفعات تکرار الگوریتم
m=2;%  تعداد متغیرها
BS=[10 10];%  برداری با طول تعداد متغیرها که در آن تعداد بیت مورد نیاز هر متغیر لحاظ شده است
L=sum(BS);% تعداد کل بیت‌های در نظر گرفته شده برای یک کروموزوم که برابر مجموع بیت‌های متغیرهای مسأله است
%
Lo=[-4 -1.5];%  برداری با طول تعداد متغیرها که در آن حد پایین هر یک از متغیرها لحاظ شده است
Hi=[ 2  1];%  برداری با طول تعداد متغیرها که در آن حد بالای هر یک از متغیرها لحاظ شده است
Population=round(rand(N,L));% تولید جمعیت اولیه به صورت اتفاقی با تابع توزیع احتمال یکنواخت
%%%%%%%%%%%%%%%%%%%%
% برنامه حاضر قابلیت حل تمام مسایل مهندسی با متغیرهای باینری و پیوسته را دارد. برای حل مسایل مختلف،
% صرفاً باید پارامترهای الگوریتم که در قسمت بالا آمده‌اند، تنظیم شوند. علاوه بر آن، تابع شایستگی مناسب
% برای هر یک از مسایل باید متناسب و سازگار با آن تعریف شود
%
%%%%%%%%%%%%%%%%%%%%
best_so_far=[];
Average_fitness=[];
for it=1: ITER  %الگوریتم تا تعداد دفعات از پیش تعیین شده، تکرار می‌شود
[real_val]=chrom_decode(Population,N,L,BS,m,Lo,Hi);
```

```
این تابع مقادیر رمز شده باینری کروموزوم‌ها را گرفته و مقادیر واقعی رمزگشایی شده آن‌ها را برمی‌گرداند %
[selection_probability,fit,ave_fit,max_fit, opt_sol]=fit_eval(real_val,N,m);
این تابع مقادیر واقعی متغیرها را گرفته، هر راه حل را ارزیابی کرده و مقادیر شایستگی آن را برمی‌گرداند%
%%%% محاسبه مقادیر میانگین شایستگی جمعیت و بهترین جواب دیده شده
if it==1
   best_so_far(it)=max_fit;
   final_sol=opt_sol;
elseif max_fit>best_so_far(it-1)
   best_so_far(it)=max_fit;
   final_sol=opt_sol;
else
   best_so_far(it)= best_so_far(it-1);
end
Average_fitness(it)=ave_fit;
%%%%
[mating_pool]=g_roulette_wheel(Population,N,selection_probability);
این زیر برنامه والدین را برای زاد و ولد و تولید مثل بر مبنای شایستگی و روش چرخ‌گردان انتخاب می‌کند%
[new_pop]=g_crossover(mating_pool,Pc,N,L);
این تابع عملگر همبری یک نقطه‌ای را روی والدین انتخاب شده اعمال می‌کند%
[Population]=g_mutation(new_pop,Pm,N,L);
end
display('final Solution       optimum fitness');
result=[final_sol,best_so_far(end)]
x=1:ITER;
figure,plot(x,best_so_far,'k',x,Average_fitness,'.-k');
xlabel('Generation');
ylabel('Fitness Function')
legend('Best-so-far','Average fitness')
%%%%%%%%%%%%%%%%%%%%%%%%%%


                    %%%% chrom_decode.m  %%%
این تابع مقادیر رمز شده باینری کروموزوم‌ها را گرفته و مقادیر واقعی رمزگشایی شده آن‌ها را بر می‌گرداند %
function [real_val]=chrom_decode(Population,N,L,BS,m,Lo,Hi)
ابتدا باید برنامه‌ای نوشته شود که قسمت ابتدایی و انتهایی هر متغیر(ژن) در کروموزوم مشخص شود. در ادامه %
ابتدا این قسمت از برنامه نوشته می‌شود
%
real_val=[];
STED(1)=1;
for i=2:m+1
   STED(i)=STED(i-1)+BS(i-1);
end
%%%%%%%%%%%%%%%%
for j=1:m
  x=BS(j)-1:-1:0;
```

```
  Pow2x=2.^x; % ساختن یک رشته عددی از توان‌های ۲ برای دهدهی کردن عدد باینری
   for i=1:N
      gene=Population(i,STED(j): STED(j+1)-1);% جدا کردن یک متغیر از یک کروموزوم
      Var_norm=sum(Pow2x.*gene)/(2^BS(j)-1);% مقدار نرمال متغیر
      real_val(i,j)=Lo(j)+(Hi(j)-Lo(j))*Var_norm;% مقدار حقیقی متغیر
    end
end
return;
%%%%%%%%%%%%%%%%
```

%%%% fit_eval.m %%%%

```
% این تابع مقادیر واقعی متغیرها را گرفته، هر راه حل را ارزیابی کرده و مقادیر شایستگی آن را بر می‌گرداند
function[selection_probability,fit,ave_fit,max_fit, opt_sol]=fit_eval(real_val,N,m)
for i=1:N
   x=real_val(i,:);
   fit(i)=(1+cos(2*pi*x(1)*x(2)))*exp(-(abs(x(1))+abs(x(2)))/2);% محاسبه مقدار شایستگی
end
selection_probability=fit/sum(fit);%  محاسبه احتمال انتخاب
ave_fit=mean(fit);%                   محاسبه متوسط شایستگی جمعیت
[max_fit,max_loc]=max(fit);%          محاسبه مقدار بیشینه شایستگی
opt_sol=real_val(max_loc,:);
return;
%%%%%%%%%%%%%%%%
```

%%%% g_roulette_wheel.m %%%%

```
% این زیر برنامه والدین را برای زاد و ولد و تولید مثل بر مبنای شایستگی و روش چرخ‌گردان انتخاب می‌کند
function [mating_pool]=g_roulette_wheel(Population,N,selection_probability)
cdf(1)=selection_probability(1); % محاسبه احتمال تجمعی جمعیت
for i=2:N
  cdf(i)=cdf(i-1)+selection_probability(i);
end
 %%%%% پیاده سازی چرخ‌گردان
for i=1:N
  q=rand;
  for j=1:N
    if q<=cdf(j)
      mating_pool(i, :)=Population(j, :);
      break;
    end
  end
```

```
end
return;
%%%%%%%%%%%%%%%%%%
```

%%%% g_crossover.m %%%%

```
%این تابع عملگر همبری یک نقطه‌ای را روی والدین انتخاب شده اعمال می‌کند
function [new_pop]=g_crossover(mating_pool,Pc,N,L);
parent_num=randperm(N);
for j=1:2:N
    pointer1= parent_num (j);%        شمارنده مربوط به والد۱
    pointer2= parent_num (j+1);%      شمارنده مربوط به والد۲
    cut_point= randint(1,1,L);%       انتخاب محل برش به صورت اتفاقی
    off1=mating_pool(pointer1,:);%    کپی والد اول را در فرزند اول قرار بده
    off2=mating_pool(pointer2,:); %   کپی والد دوم را در فرزند دوم قرار بده
  if rand <Pc    %                    با نرخ احتمال Pc عمل همبری را انجام بده
        temp= off2;
        off2(cut_point+1:end)=off1(cut_point+1:end);
        off1(cut_point+1:end)=temp(cut_point+1:end);
   end
new_pop(j,:)=off1;%                   فرزندان تولید شده را به جمعیت جدید منتقل کن
new_pop(j+1,:)=off2;
end
return;
%%%%%%%%%%%%%%%%%%
```

%%%% g_mutaion.m %%%%

```
%این تابع، عملگر جهش را روی جمعیت تولیدشده، اعمال می‌کند
function [Population]=g_mutation(new_pop,Pm,N,L);
 mask=rand(N,L)<=Pm; %    با نرخ احتمال Pm عمل جهش را انجام بده
Population=xor(new_pop,mask);
return;
%%%%%%%%%%%%%%%%%%
```