Project #2: Newton-Krylov method for solving nonlinear systems

Advanced numerical analysis

Behzad Baghapour

1 Introduction

We examine the Newton-Krylov algoithm for solving the steady-state nonliner Burger's equation using scipy.optimize package. The effect of line search and preconditioning are studied in convergence rate. The simulations are performed in a one-dimensional domain with uniform mesh of different sizes. Please provide your answers in a PDF file format including the explanations, algorithm, graphs, and codes developed in your report. For your report, consider *problem statement*, *methodology*, *results and discussions*, and *conclusion* sections. Submit your answers to the Courses according to the prescribed due-date.

2 Problem statement

Burger's equation has a steady-state solution with the prescribed boundary conditions as follows:

$$\begin{cases} u\frac{du}{dx} - \nu\frac{d^2u}{dx^2} = 0, \\ u(0) = +1, \\ u(1) = -1 \end{cases} \tag{1}$$

In Eq. (1), u is the scalar property. The first term, udu/dx is the convection and $\nu d^2u/dx^2$ is the diffusion of the property in the one-dimensional domain and ν is the diffusivity coefficient. The problem can be seen as a nonlinear system which the steady-state solution $u(x)$ is seeking to find as a root of nonlinear equation $F(u) = 0$ in the conservative form:

$$F(u) = 0, \qquad F(u) := \frac{d(u^2/2)}{dx} - \nu\frac{d^2u}{dx^2} \tag{2}$$

The domain is discretized into uniform mesh $\Delta x = 1.0/(n-1)$ where n is the number of grid points. The nonlinear operator in Eq. (1) is also discretized by the second-order central finite difference method. Therefore, for each node of domain we have:

$$F(i) = \frac{1}{4\Delta x}(u_{i+1}^2 - u_{i-1}^2) - \frac{\nu}{\Delta x^2}(u_{i+1} - 2u_i + u_{i-1}), \qquad i = 0, 1, \cdots, n-1 \tag{3}$$

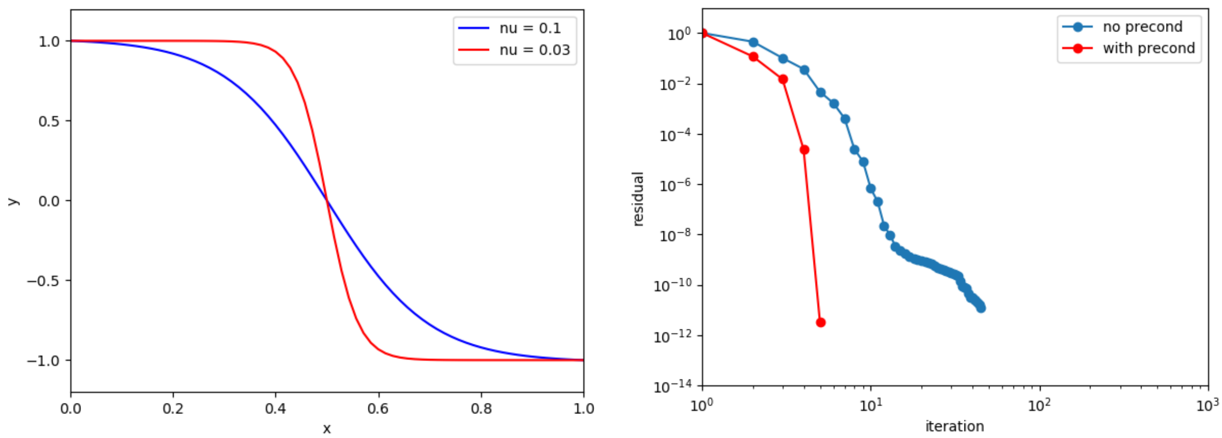
The norm $\|F\|$ is going to be minimized using the Newton-Krylov algorithm. Accordingly, the nonlinear problem is linearized as follows:

$$J(x)\delta x = R(x) \tag{4}$$

where $J(x) = F'(x)$ is the Jacobian of F and $R(x) = -F(x)$ is the nonlinear residual. For a discretized domain with n grid points, the Jacobian, $J_{n,n}$ has the dimension of $n \times n$ and the residual R_n has the dimension of n. The linearized system of equations (4) can be solved using fast Krylov methods (e.g. GMRES). The overall solution steps can be summarized as follows:

```
----------------------
Newton-Krylov algoritm
----------------------
1. Linearizing the nonlinear system F(x)=0
2. Solving the linearized system with Krylov methods:
   2.1. Preconditioning the linearized system
3. Global convergence check by stabilization techniques:
   3.1. Line search, Trust zone, ...
4. Update the solution for the next iteration
```

Figure 1(a) compares the field with different diffusivity coefficients and Fig. 1(b) shows the effect of preconditioning on the convergence with *Armijo* line search.



(a) Different diffusivity

(b) Preconditioning effect, $\nu = 0.1$

Figure 1: Solution of steady-state Burger's equation for $n = 71$.

3 Solution method

The system of nonlinear equations (3) is going to be solved using `newton_krylov` method of scipy.optimize module. The mesh dimensions considered for the domain $-1 \leq x \leq +1$ is $nx = \{51, 101, 301\}$ with the nonlinear residual norm of `f_tol=1e-10` and initial guess $\mathbf{u}_0 = 0$.

1. Compare the convergence rate (log scale), without line search and preconditioning, for the mesh size $n = 301$ with $\nu = 0.1, 0.01, 0.001$. Plot the scalar field for different diffusivity and analyze the effect of dissipation on the field and convergence.

2. For $\nu = 0.1$, and different mesh sizes with Armijo line search, study the effect of preconditioning M on the convergence rate of `newton_krylov`. Accordingly, the viscous (diffusion) part of Eq. (1) with sparse ILU `spilu` and `fill_factor=10` is considered:

$$F_v(u) = \nu \frac{\partial^2 u}{\partial x^2} \approx \frac{\nu}{\Delta x^2}(u_{i+1} - 2u_i + u_{i-1}) \ \rightarrow \ M_{ij} = \frac{\partial F_{v,i}}{\partial u_j}, \ M_{i,i\pm1} = \frac{\nu}{\Delta x^2}, M_{i,i} = \frac{-2\nu}{\Delta x^2}$$

   To set the preconditioner, use `LinearOperator` and introduce it to the method by `inner_M`.