MANE-6760
Vignesh Ramakrishnan
RIN: 662028006
Due: 11pm November 15, 2022

MANE 6760 (FEM for Fluid Dyn.) Fall 2022: HW3

1. (10 points) Consider $\kappa = \kappa_0 \left(1 + \frac{1}{1+\phi_{,x}^2}\right)$. The nonlinear weak residual is given as:

$$G_A = \int_0^L \left(\ldots + \ldots + N_{A,x}\kappa\bar{\phi}_{,x} + \ldots + \ldots\right)\, dx$$

Find the contribution (only) of the term shown above to the tangent equation/LHS matrix $\frac{\partial G_A}{\partial \hat{\phi}_B}$. Hint: To think about $\kappa = \kappa(\phi_{,x})$.

The tangent equation is written as:

$$G_A + \frac{\partial G_A}{\partial \hat{\phi}_B}\Delta\phi_B = 0$$

The term under consideration in G_A will be differentiated as:

$$\begin{aligned}
&= \frac{\partial}{\partial \hat{\phi}_B}\left(N_{A,x}\kappa\bar{\phi}_{,x}\right) \\
&= N_{A,x}\left(\kappa\frac{\partial \sum N_{i,x}\hat{\phi}_i}{\partial \hat{\phi}_B} + \frac{\partial \kappa}{\partial \hat{\phi}_B}\bar{\phi}_{,x}\right) \\
&= N_{A,x}\left(\kappa N_{B,x} + \frac{\partial \kappa}{\partial \bar{\phi}_{,x}}\frac{\partial \bar{\phi}_{,x}}{\partial \hat{\phi}_B}\bar{\phi}_{,x}\right), \text{ where, } \kappa = \kappa(\bar{\phi}_{,x}) = \kappa_0\left(1 + \frac{1}{1+\bar{\phi}_{,x}^2}\right) \\
&= N_{A,x}\left(\kappa N_{B,x} + \frac{-2\kappa_0\bar{\phi}_{,x}}{(1+\bar{\phi}_{,x}^2)^2}N_{B,x}\bar{\phi}_{,x}\right) \\
&= N_{A,x}\kappa N_{B,x} - N_{A,x}\frac{2\kappa_0\bar{\phi}_{,x}^2}{(1+\bar{\phi}_{,x}^2)^2}N_{B,x}
\end{aligned}$$

2. (20 points) Update the code for the above equation (i.e., $\kappa = \kappa_0\left(1 + \frac{1}{1+\phi_{,x}^2}\right)$). Keep all the other settings the same (e.g., a_x, κ_0, s, N_e, etc.). Provide the updated solution plot and the updated Python code.

The updated code is in Listing 1, 2, 3, 4 and the solution to this problem is in Fig 1.

```
    def get_kappa2(prob_case, dphidx):
        kappa0 = get_kappa0()
        kappa = 0.0
        if prob_case<=2:
            kappa = kappa0*(1.0 + (1.0/(1.0 + dphidx**2)))
        else:
            print('Invalid choice of problem: ', prob_case)
            exit()
```

```
    return kappa

```

Listing 1: Update to computing kappa

```
    def get_kappa_dphix(prob_case, dphidx):
        kappa0 = get_kappa0()
        kappa_dphix = 0.0
        if prob_case<=2:
            kappa_dphix = -2.0*kappa0*dphidx/((1.0+dphidx**2.0)**2.0)
        else:
            print('Invalid choice of problem: ', prob_case)
            exit()
    return kappa_dphix

```

Listing 2: Update to compute $\frac{\partial \kappa}{\partial \bar{\phi}_{,x}}$

```
        kappaq = get_kappa2(prob_case,phidgblq)
        kappa_dphixq = get_kappa_dphix(prob_case,phidgblq)
        tauq = get_tau(kappaq)
        kappa_numq = tauq*ax*ax
        for idx_a in range(nes): # loop index in [0,nes-1]
            be[idx_a] = be[idx_a] \
                        - (shpdgbl[idx_a,q])*ax*phiq*wdetj \
                        + (shpdgbl[idx_a,q])*(kappaq+kappa_numq)*(phidgblq)*
    wdetj \
                        - shp[idx_a,q]*sq*wdetj \
                        - (shpdgbl[idx_a,q])*tauq*ax*sq*wdetj
            for idx_b in range(nes): # loop index in [0,nes-1]
                Ae[idx_a,idx_b] = Ae[idx_a,idx_b] \
                                  - (shpdgbl[idx_a,q])*ax*shp[idx_b,q]*wdetj \
                                  + (shpdgbl[idx_a,q])*(kappaq+kappa_numq) \
                                  *(shpdgbl[idx_b,q])*wdetj \
                                  +(shpdgbl[idx_a,q])*(kappa_dphixq)*\
                                  (phidgblq)*shpdgbl[idx_b,q]*wdetj

```

Listing 3: Update to compute $\frac{\partial G_A}{\partial \hat{\phi}_B}$

```
        def get_s(prob_case, x):
            # return s value
            s = 0.0

            if prob_case == -1:
                s = 1.0
            elif prob_case==0  or prob_case==1 or prob_case==2:
                # phi_exact = 1+x-(g1x+g2c)
                #           = 1-g2c + x - g1x = k1 + x - g1x
                # where g1x+g1c = (exp(-gamma(L-x)) \
                #               - exp(-gamma L))/(1 - exp(-gamma L))

                ax = get_ax()
                kappa0 = get_kappa0()

                axref = ax # assumed constant over the mesh
                kapparef = kappa0 # reference kappa
                L = get_L()
                gamma = axref/kapparef
```

```
                # involves constant part: 1-g2c
                k1 = 1.0/(1.0-np.exp(-gamma*L))
                # involves exponential function of x
                g1x = k1*np.exp(-gamma*(L-x))

                # first derivative of g1x
                g1xdx = gamma*g1x
                # first derivative of g1xdx or second derivative of g1x
                g1xd2x = gamma*g1xdx

                phi_exact = (k1 + x - g1x)
                dphi_ex_dx = (1.0 - g1xdx)
                kappa = get_kappa2(prob_case,dphi_ex_dx)
                # a form is easy to derive by hand
                s = ax*(1.0-g1xdx) - kappa*(-g1xd2x)
            else:
                print('Invalid prob_case of (for s):',prob_case)
                exit()
    return s

```

Listing 4: Update to computing s=s(x)

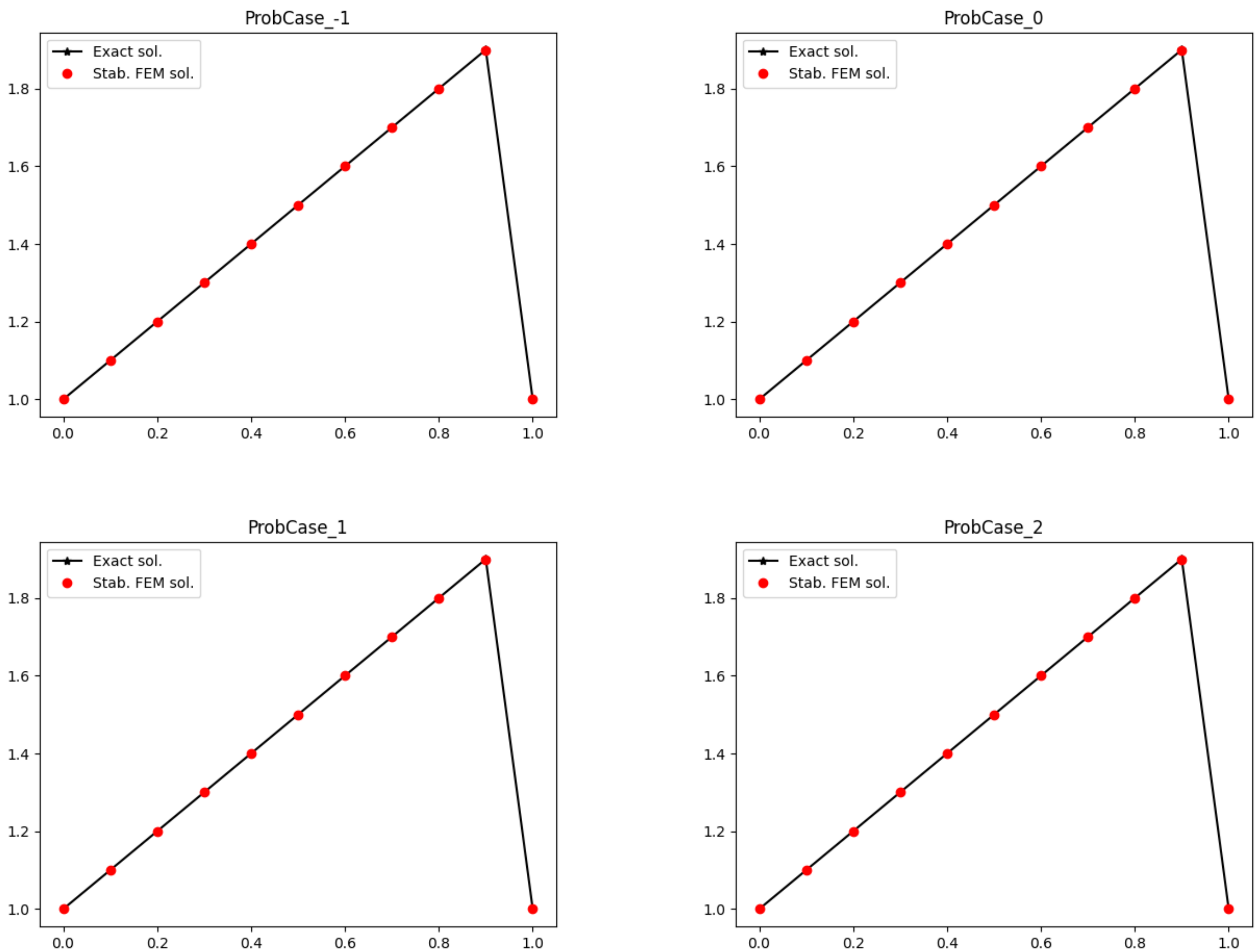

Figure 1: Solution to nonlinear $\kappa(\phi_{,x})$