code_2

October 6, 2020

1 Simplification for Finite Element Method (SFEM)-Coding tutorial (b)

- FEM solution for a concrete mechanic problem
- Convergence analysis

[59]:
```
import numpy as np
from numpy.polynomial.legendre import leggauss # Gauss quadrature
import matplotlib.pyplot as plt
from scipy import integrate

%matplotlib inline
```

Gauss numerical quadreture function

[60]:
```
def gauss_legendre_quad(f, n, a, b):
    x, w = leggauss(n)
    sum_ = 0
    for k in range(len(x)):
        sum_ += w[k] * f(0.5*(b-a)*x[k]+0.5*(b+a))
    return 0.5 * (b-a) * sum_
```

Linear shape function

[61]:
```
N1 = lambda x: -x/2+1/2
N2 = lambda x: x/2+1/2
dN1 = lambda x: -1/2  # B1
dN2 = lambda x: 1/2  # B2
```

1.1 Problem description

A bar of length $2l$, cross-sectional area A and Young's modulus E. The bar is fixed at $x = 0$, subjected to linear body force cx and applied traction $\bar{t} = -cl^2/A$ at $x = 2l$ as shown in Fig as follow:

<img src="model.png",width =600>

[62]:
```
E = 10e4  # Young modulus Nm-2
A = 1.  # Section area
c = 1.  # Nm-2
l = 1. # m
```

Strong form is given

$$\frac{d}{dx}\left(AE\frac{du}{dx}\right) + cx = 0,$$

$$u(0) = 0,$$

$$\bar{t} = E\frac{du}{dx}n\Big|_{x=2l} = -\frac{cl^2}{A}$$

exact (analytic) solution:

$$u(x) = \frac{c}{AE}\left(-\frac{x^3}{6} + l^2 x\right)$$

[63]:
```
f_ex = lambda x: c/(A*E)*(-x**3/6+l**2*x)
```

Derivation for the weak form (variational formulation)
Multiplication of test function $v \in U^0$ and integration by part from $(0, 2l)$

$$AEv\frac{du}{dx}\Big|_0^{2l} - \int_0^{2l} AE\frac{dv}{dx}\frac{du}{dx}dx + c\int_0^{2l} vxdx = 0 \tag{1}$$

$$\int_0^{2l} AE\frac{dv}{dx}\frac{du}{dx}dx = \int_0^{2l} vcxdx - vcl^2 \tag{2}$$

Thus, weak form for considered problem is given: find $u \in U$

$$\int_0^{2l} AE\frac{dv}{dx}\frac{du}{dx}dx = -cl^2\Big|_{x=2l} + \int_0^{2l} vcxdx, \ \ v \in U^0 \tag{3}$$

Matrix form

$$v\int_0^{2l} AEB^T Bdxu = -cl^2\Big|_{x=2l} + v\int_0^{2l} Ncxdx, \ \ v \in U^0 \tag{4}$$

Setup

[64]:
```
nb_e = 160  # number of element
h = 2*l / nb_e  # element size
nb_dof = nb_e + 1  # number of degree of freedom
x_nodes = np.linspace(0, 2*l, nb_dof)
```

Elementary stiffness matrix

[65]:
```
N = [N1, N2]
dN = [dN1, dN2]
K_e = np.zeros((2, 2))
M_e = np.zeros((2, 2))
for i in range(len(dN)):
    for j in range(len(dN)):
        f = lambda x: dN[i](x) * dN[j](x)
        K_e[i, j] = (2 / h) * gauss_legendre_quad(f, 5, -1, 1)
        g = lambda x: N[i](x) * N[j](x)
        M_e[i, j] = (h / 2) * gauss_legendre_quad(g, 5, -1, 1)
```

Assembly the global matrix system

[66]:
```
K = np.zeros((nb_dof, nb_dof))
for i in range(nb_e):
    K[i:i+2, i:i+2] += A*E*K_e
F = np.zeros((nb_dof))
F_e = np.zeros((2))
for i in range(nb_e):
    ff = lambda x: ((x_nodes[i+1]-x)/h)*c*x
    gg = lambda x: ((x-x_nodes[i])/h)*c*x
    F_e[0]= gauss_legendre_quad(ff, 5, x_nodes[i], x_nodes[i+1])
    F_e[1] = gauss_legendre_quad(gg, 5, x_nodes[i], x_nodes[i+1])
    F[i:i+2] += F_e
```

Constraint the boundary conditions

[67]:
```
F[-1] += -c*l**2  # Neumann BC

K[0] = 0  # Dirichlet BC
K[:,0] = 0
K[0,0] = 1
F[0] = 0
```

Solving the linear system

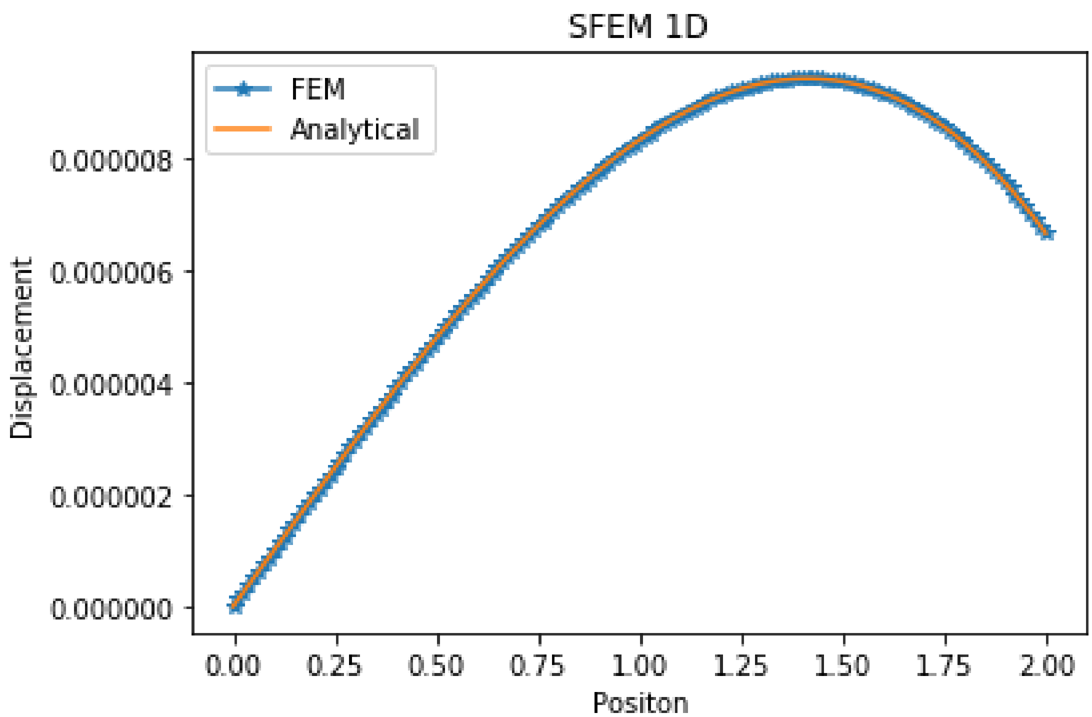
[68]:
```
U = np.zeros((nb_dof))
U[:] = np.linalg.solve(K, F)
```

1.2 Plot the solutions

[69]:
```
fig = plt.figure()
ax = fig.add_subplot(111)
ax.set_title('SFEM 1D')
ax.set_xlabel('Positon')
ax.set_ylabel('Displacement')
ax.plot(x_nodes, U, '-*', label='FEM')
```

```
U_ex =  [f_ex(x) for x in x_nodes]
ax.plot(x_nodes, U_ex, '-', label='Analytical')
ax.legend()
```

[69]: <matplotlib.legend.Legend at 0x7f88bd0db1d0>



1.3 Error estimation and convergence analysis

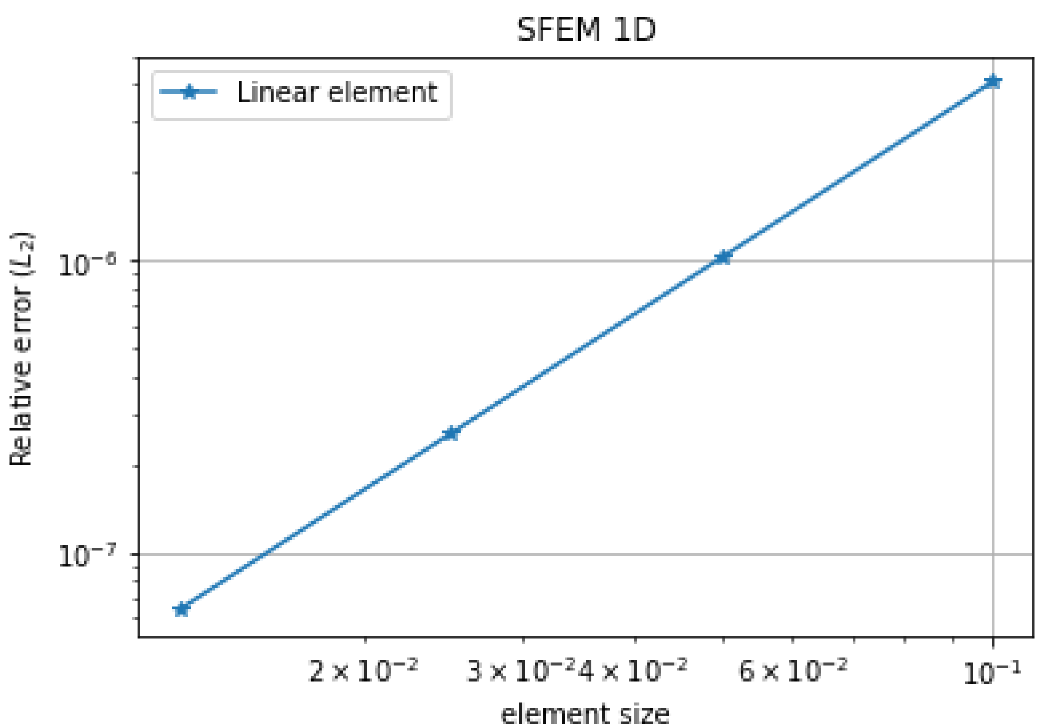
[70]:
```
e_l2 = 0
for i in range(nb_e):
        u_ex_s = integrate.quad(f_ex, 0, 2)[0]
        x_trans = lambda x: (2 * x) / h - (x_nodes[i] + x_nodes[i+1])/h
          #  use of analytical lobatto expressions
        u_FE = lambda x: sum(N[j](x_trans(x)) * U[i + j] for j in range(len(N)))
        f_error = lambda x:(f_ex(x)-u_FE(x))**2
        e_l2_element = gauss_legendre_quad(f_error, 20, x_nodes[i],␣
 ↪x_nodes[i+1])
#         print(e_l2_element)
        e_l2 += e_l2_element
e_norm = np.sqrt(e_l2 / u_ex_s)
```

[71]:
```
e_norm
```

[71]: 6.378849861675608e-08

[76]:
```
size_set = [2/20, 2/40, 2/80, 2/160]
error_set = [4.081267699102172e-06, 1.020544784284396e-06, 2.
↪551504351688869e-07, 6.378849861675608e-08]
fig = plt.figure()
ax = fig.add_subplot(111)
ax.set_title('SFEM 1D')
ax.set_xlabel('element size')
ax.set_ylabel('Relative error ($L_2$)')
ax.plot(size_set, error_set, '-*', label='Linear element')
ax.loglog()
ax.grid('True')
ax.legend()
```

[76]: <matplotlib.legend.Legend at 0x7f88b622f110>



[ ]: