Finite Element Methods (AERO97053)
Individual Assignment
Due: 01/05/2020

Setup and solve the following problem. Document each step of the implementation in your report and attach a copy of the coding to your submission. Your code should also be commented.

Consider the Helmholtz problem

$$\frac{\partial}{\partial x}\left(\sigma\frac{\partial u}{\partial x}\right) - \lambda u = f(x). \tag{1}$$

1. Analytically construct the Galerkin weak form of equation (1) in the region $a \leq x \leq b$ describing how to impose the Neumann and Dirichlet boundary conditions

$$u(a) = \alpha, \ \frac{du}{dx}(b) = \beta, \tag{2}$$

where α and β are known constants. [15%]

2. Discuss how to enforce the alternative mixed boundary condition

$$\sigma\frac{du}{dx}(a) + u(a) = \gamma. \tag{3}$$

where γ is a constant. [10%]

3. For the region $0 \leq x \leq 1$, show that the solution $u^{ex}(x) = \cos(2\pi x)$ satisfies equation (1) when $f(x) = -(4\sigma\pi^2 + \lambda)\cos(2\pi x)$ and determine the values of α, β and γ in the boundary conditions in equations (2) and (3). [10%]

4. Set up the linear and quadratic finite element approximation of the Helmholtz problem (1) in the interval $0 \leq x \leq 1$ with the boundary conditions given by equation (2) and $\lambda = 1$. For the linear expansion use the standard elemental expansion basis

$$\begin{aligned} \phi_0(\xi) &= \frac{1-\xi}{2} \\ \phi_1(\xi) &= \frac{1+\xi}{2} \end{aligned}$$

where $-1 \leq \xi \leq 1$. For the quadratic finite element approximation use the following shape functions

$$\begin{aligned} \phi_0(\xi) &= \frac{\xi(\xi-1)}{2} \\ \phi_1(\xi) &= (1-\xi)(1+\xi) \\ \phi_2(\xi) &= \frac{\xi(\xi+1)}{2} \end{aligned}$$

You will require a matrix inversion technique to complete this exercise. This can be obtained by either using Matlab to write your code or a matrix inversion package such as LAPACK or the Nag libraries. In the LAPACK library the function calls for LU factorisation of a general matrix are $dgetrs$ and $dgetrf$.

A short code performing matrix inverse and a solution of a linear system of equations in Python 3 is presented in the appendix.

a) Discuss how integration, differentiation and global matrix assembly are performed in your implementation explaining alternative methods if appropriate[15%]

b) Using a mesh of $N_{el} = 5, 10, 20, 50, 100$ equispaced elements determine the L_2 norm of the error $\epsilon(x_i) = u^{ex}(x_i) - u^{\delta}(x_i)$ between the exact $u^{ex}(x)$ and the numerical solution $u^{\delta}(x)$ defined as

$$L_2 = \sqrt{\sum_{i=0}^{N} \frac{[u^{ex}(x_i) - u^{\delta}(x_i)]^2}{(N+1)}},$$

where

$$x_i = i/N, \quad 0 \leq i \leq N,$$

and $N := N_{el}$ for linear finite element expansion and $N := 2N_{el}$ for quadratic finite element expansion.

Plot the error, for both your linear and quadratic finite element expansions, versus the size of the element size $1/N_{el}$ on a log-log graph [35%]

c) Comment on the slopes of error curves on the plot and whether there is an alternative way to express the solution error. [15%]

Some questions and answers regarding the assignment.

Below is a list of questions and answers regarding the assignment. These are questions that seem to come up repeatedly.

1. Q: Can we assume that σ in equation (1) is constant?
   A: Yes

2. Q: Can we compute entries in the elemental (mass and Laplace) matrices analytically, i.e. can we evaluate L_2 products of basis functions or their derivatives in reference element by integrating 'on paper' instead of using numerical quadrature?
   A: No. One of the points of the exercise is to demonstrate that you know how to select quadrature of sufficient accuracy and use it.

3. Do we need to construct differentiation matrices in order to compute basis function derivatives?
   A: No. If you need to compute discrete Laplacian, it is OK to express basis function derivatives analytically (on paper) and then transform them from physical to

reference space as needed in your code.

4. Do we need to respect the local mode numbering of quadratic modes in question 4? We used a bit different numbering during the tutorial session.
   A: Yes, please stick to the numbering used in the assignment sheet.

5. Q: Are we supposed to construct and use $\mathcal{A}$ and its transpose for global assembly?
   A: It is preferred not to. Using a map of DOF (degree of freedom) numbers between element-local and global numbering should suffice.

6. Q: Do we need to use sparse matrices or programming language X instead of Y in order to make the code more efficient?
   A: Dense global matrix is fine. Choice of language (Matlab, C, C++, Python) is up to the student as long as I am able to decipher the code later . Prefer ease of implementation, code clarity and correctness over efficiency.

Appendix

The following Python code constructs and solves a small linear system of equations:

```
1  import numpy as np
2  from numpy.linalg import inv
3
4  rows = 3;
5  cols = 3;
6
7  # A_data is a 2-dimensional array holding
8  # all matrix values
9  A_data = np.zeros([rows,cols])
10
11 # Here we fill A_data with some values ...
12 A_data[0][0] = 1.0
13 A_data[0][1] = 2.0
14 A_data[0][2] = 3.0
15
16 A_data[1][0] = 4.0
17 A_data[1][1] = 5.0
18 A_data[1][2] = 6.0
19
20 A_data[2][0] = 7.0
21 A_data[2][1] = 8.0
22 A_data[2][2] = 1.0
23
24 # For illustration, we generate the right-hand side vector b
25 # as a sum of all matrix columns:
```

```
26 b = np.sum(A_data, axis=1)
27
28 # In order to be able to compute the inverse,
29 # we will construct a matrix object from given data
30 A = np.matrix(A_data)
31 A_inv = inv(A)
32
33 # Finally, solve the linear system
34 # A * x = b by multiplying
35 # x = A^(-1) * b
36 #
37 # Note that the operation A_inv.dot(b) returns
38 # an object of type matrix (with 1 column) - the methods
39 # 'squeeze' and 'asarray' convert it to standard numpy array
40 #
41 # The expected result is a vector of ones
42
43 x = np.squeeze(np.asarray(A_inv.dot(b)))
44
45 print(x)
```