MATH 575 ASSIGNMENT 4

In this assignment, you will use MATLAB's PDE Toolbox to numerically solve several partial differential equations. To do this assignment, it will be necessary to first read the document MATLAB's PDE Toolbox Commands, which is found at

`http://math.rutgers.edu/~falk/math575/matlab-pdeinfo.html`

1. Use MATLAB's GUI to numerically solve the boundary value problem

$$-\Delta u = 1 \quad \text{in} \quad \Omega, \qquad u = 0 \quad \text{on} \quad \partial\Omega,$$

where Ω is the unit circle. Print out and hand in a 3-D plot of the approximate solution obtained by MATLAB. To get this plot, choose *Parameters* from the Plot menu, delete the *Color* box, and add the *Height (3-D plot)* box.

2. First download the file *circp2.m* from

`http://math.rutgers.edu/~falk/math575/assignments.html`

to your directory, keeping this name. This file contains commands which solve the boundary value problem of Problem 1 without using the GUI, provided the files *circleg.m* containing information about the domain geometry and *circleb1.m* containing information about the boundary conditions already exist. These two files are part of the PDE Toolbox and you do NOT need them in your directory. The program *circp2* calculates various norms of the error between u_I, the piecewise linear interpolant of the true solution u, and the approximate solution u_h.

Now start MATLAB and type `circp2` (which executes the commands in the file *circp2.m*). When the MATLAB prompt "`>>`" returns, you can obtain the error quantities calculated by the program by typing: `fcnerror, maxerror, graderror, maxexerror`, and `maxeyerror`. These commands will each produce a vector whose first component gives the error for an initial mesh size and successive components give the error when the mesh size is successively cut in two.

fcnerror : measures the L^2 error between u_h and u_I,
maxerror : measures the maximum absolute error $|u_I - u_h|$ at the mesh points,
graderror : measures the L^2 error between $\operatorname{grad} u_h$ and $\operatorname{grad} u_I$,
maxexerror: measures the maximum of $|(u_h)_x - (u_I)_x|$,
maxeyerror: measures the maximum of $|(u_h)_y - (u_I)_y|$.

Hand in a table of the errors calculated by *circp2* for different mesh sizes and your calculation of the orders of convergence for each type of error. Do the orders correspond to the theory developed in class?

3. In this problem, we combine use of the GUI and the main MATLAB workspace to obtain the approximate solution and analyze its accuracy. We use the GUI to input the geometry

and boundary data, and and export this information to the main MATLAB workspace to perform additional analysis. Consider the boundary value problem

$$-\Delta u = 2[x(1-x)+y(1-y)] \quad \text{in } \Omega, \qquad u = 0 \quad \text{on } \partial\Omega,$$

where Ω is the unit square $[0,1]\times[0,1]$. The exact solution of the problem is given by $u = x(1-x)y(1-y)$. Start up MATLAB and type `pdetool`. Change the axes limits to $x = -1, 2$ and $y = -0.5, 1.5$. Then draw the domain Ω. Next input the boundary conditions of the problem and export the decomposed geometry and boundary conditions. Exit the PDE Toolbox and use the commands *wgeom* and *wbound* to save the decomposed geometry and boundary conditions to the files *square01g.m* and *square01b.m*, respectively. Then use ideas from the code in the file *circp2.m* to compute the same error quantities for this problem. In this case, quit the loop when maxerror $< .0001$. Use the results you obtained to compute the order of convergence of each of the error quantities obtained. Do they correspond to the theory developed in class? Also hand in a printout of the final mesh used. Note that a simple way to input the right hand side of the above equation into the function assempde is to define `f = '2.*(x.*(1-x) + y.*(1-y))';` and then use `f` as the last argument of assempde.

4. First download the files *elasinit.m* and *elassoln.m* from

`http://math.rutgers.edu/~falk/math575/assignments.html`

to your directory, keeping these names. Consider the equations of plane strain elasticity given by:

$$-\operatorname{div} C\boldsymbol{\epsilon}(\boldsymbol{u}) = \boldsymbol{f}, \quad \text{in } \Omega, \qquad \boldsymbol{u} = 0 \quad \text{on } \partial\Omega,$$

where Ω is the unit square $[0,1]\times[0,1]$,

$$\boldsymbol{\epsilon}_{ij}(\boldsymbol{u}) = \begin{pmatrix} \partial u_1/\partial x & (\partial u_2/\partial x + \partial u_1/\partial y)/2 \\ (\partial u_2/\partial x + \partial u_1/\partial y)/2 & \partial u_2/\partial y \end{pmatrix},$$

and for a 2×2 matrix $\boldsymbol{\tau}$, $\operatorname{tr}(\boldsymbol{\tau}) = \tau_{11} + \tau_{22}$,

$$C\boldsymbol{\tau} = \frac{E}{1+\nu}\left[\boldsymbol{\tau} + \frac{\nu}{1-2\nu}\operatorname{tr}(\boldsymbol{\tau})\boldsymbol{I}\right], \qquad \boldsymbol{I} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \qquad \operatorname{div}\boldsymbol{\tau} = \begin{pmatrix} \partial\tau_{11}/\partial x + \partial\tau_{12}/\partial y \\ \partial\tau_{21}/\partial x + \partial\tau_{22}/\partial y \end{pmatrix}.$$

Also,

$$\begin{aligned} f_1 = 40000(x-y)(&6x^3y - 3x^3 + 4x^2 - 6x^2y^2 - 3x^2y + 6xy^3 - 2xy \\ &-3xy^2 - 1 + 4y^2 - 3y^3), \\ f_2 = -40000(x-1+y)(&6x^3y - 3x^3 - 15x^2y + 5x^2 + 6x^2y^2 - x + 10xy - 15xy^2 \\ &+6xy^3 + 5y^2 - y - 3y^3), \end{aligned}$$

The exact solution of this problem when $G \equiv E/(2[1+\nu]) = 100$ is given by

$$\boldsymbol{u} = [-v_y + \frac{1-2\nu}{2(1-\nu)}v_x, v_x + \frac{1-2\nu}{2(1-\nu)}v_y], \qquad \text{where} \qquad v = 100\,x^2(1-x^2)y^2(1-y^2).$$

a) Using *pdetool*, solve this boundary value problem when the Young' modulus $E = 260$ and Poisson ratio $\nu = 0.3$. To save time, after starting up the PDE Toolbox, select *Open* from

the *File* menu, click on *elasinit.m* and then click on *Done*. This will define the domain to be the unit square, enter the boundary conditions and the values of E, ν, and the right hand side vector f, select a mesh, and solve the boundary value problem. To continue to part (b), you must first export the solution (naming it Uh) and the mesh.

b) Exit *pdetool* and type `elassoln`. This will compute the exact solution (u_1, u_2) and store it as a column vector U containing the values of u_1 at the vertices of the triangulation followed by the values of u_2 at the vertices of the triangulation. Note that when you export the approximate solution $Uh = (uh_1, uh_2)$, it is returned in the same form. The program *elassoln* also computes the quantities *norm(U,inf)*, *norm(Uh,inf)*, and *norm(U-Uh,inf)*, which give the maximum of $|U|$, $|Uh|$, and $|U - Uh|$, respectively, at the triangle vertices.

c) Repeat parts (a) and (b) for $E = 299.998$ and $\nu = 0.49999$. Note that this can be accomplished by starting up *pdetool* and simply changing the PDE coefficients, then exporting the solution and mesh, and finally changing $\nu = 0.3$ to $\nu = 0.49999$ in the file *elassoln.m*. Hand in the results you obtained for both values of ν. Do you notice any difference in the accuracy of the approximate solution?

5. In this problem, we consider the importance of adaptive mesh refinement for problems whose solutions are rapidly changing in some part of the domain. Let Ω be the domain consisting of the circular sector $0 \le r \le 1$, $-3\pi/4 \le \theta \le 3\pi/4$. Consider the boundary value problem

$$-\Delta u = 0 \quad \text{in } \Omega, \qquad u = 0 \quad \text{on the lines } \theta = \pm 3\pi/4, \quad 0 \le r \le 1, \tag{1}$$

$$u = \cos([2/3]\arctan(y/x)), \quad \text{on } r = 1, \quad -3\pi/4 \le \theta \le 3\pi/4. \tag{2}$$

The exact solution of this problem is given by

$$u = (x^2 + y^2)^{1/3}\cos([2/3]\arctan(y/x)).$$

The geometry and boundary conditions for this problem are contained in the files *cirsg.m* and *cirsb.m*, respectively. These are part of the PDE Toolbox, so you do not have to copy them to your home directory. Download the files *singa.m* and *singr.m* from the directory

`http://math.rutgers.edu/~falk/math575/assignments.html`

to your directory and type `singa` at the MATLAB prompt. Record the number of triangles at each iteration of refinement and the final error produced. Also print out a plot of the final mesh configuration. Now type `singr` at the MATLAB prompt. Record the number of triangles and the error achieved and print out a plot of the final mesh. Hand in the plots and the information you recorded.