Name : _____

Homework Project #6

Math 365, Fall 2015

Due Friday November 20th

You may work with a partner on this assignment and turn in a single assignment for the both of you.

1. (**Using `fzero`**) Consider the Colebrook equation for the friction factor in a fully developed pipe flow

$$\frac{1}{\sqrt{f}} = -2\log_{10}\left(\frac{\epsilon/D}{3.7} + \frac{2.51}{\mathrm{Re}_D\sqrt{f}}\right)$$

where f is the Darcy friction factor, ϵ/D is the relative roughness of the pipe material, and Re is the Reynolds number based on a pipe diameter D. Use `fzero` to find the friction factor f corresponding to parameter values $\epsilon/D = 0.0001$ and $\mathrm{Re}_D = 3 \times 10^5$. Use a tolerance of 10^{-8} and write the root you find to the file `frictionfactor.out`. **Hint :** Use the function `optimset` to set up the tolerance `TolX`.

2. (**Using `fzero`**) David Peters (SIAM Review, 1997) obtains the following equation for the optimum damping ratio of a spring-mass-damper system designed to minimize the transmitted force when an impact is applied to the mass:

$$\cos\left[4\zeta\sqrt{1-\zeta^2}\right] = -1 + 8\zeta^2 - 8\zeta^4$$

Use `fzero` to find the $\zeta \in [0, 0.5]$ that satisfies this equation. Use a tolerance of 10^{-12} and write the solution to the file `damping.out`.

3. (**Using `fminbnd`**) Find the two maximums and the two minimums of the function

$$h(x) = \frac{1}{(x-0.3)^2 + 0.01} + \frac{1}{(x-0.9)^2 + 0.04}$$

over the interval $[-1, 2]$. Plot the function over this interval, along with the four extrema. Write out the x locations and the four extreme values you find to the file `extremevalues.out`.

4. (**Quadrature rules**) Approximate the following definite integral using the Midpoint rule, the Trapezoidal Rule, Simpson's Rule and Matlab's adaptive routine `integral`.

$$I = \int_{-1}^{1} (x-1)^2 e^{-x^2}\, dx$$

For the Midpoint Rule, the Trapezoidal Rule and Simpson's Rule, subdivide the interval into 100 sub-intervals. For the `integral` routine, use a tolerance of 10^{-12}.

   (a) Use Wolfram Alpha (`www.wolframalpha.com`) to find the exact expression for definite integral above. The correct expression will involve e, π, and the Matlab function `erf(x)`. Call this expression I_{exact} and compute and its value.

   (b) Compute the approximate value of the integral above, using each of the four methods listed above. Compute the error in your approximations using the exact value you found in Problem 4a. Store your results in a 4×2 array where the first column contains the four approximate values I_i and the second column contains the errors in the approximations.

   Compute your errors as $e_i = |I_i - I_{exact}|$ and write your array to the file `quaderrors.out`.

(c) Of the three methods, Midpoint Rule, Trapezoidal Rule, and Simpson's Rule, which gives the most accurate results?

5. (**Steady state heat diffusion - again.**) The temperature problem from Homework 2 and the condition number problem from Homework 3 both involved solving the second order differential equation

$$u''(x) = f(x) \tag{1}$$

for a given function $f(x)$. For both problems, we converted this second order differential equation to a linear system of equations $A\mathbf{u} = \mathbf{f}$, which we solved using Gaussian elimination (i.e. using the backslash operator). But we saw on Homework 3, Problem 1 that accumulated round off error gave rise to a situation in which, for large N, the error didn't improve any more, and in fact started to increase with increasing N. This was due to the ill-conditioning of the matrix A. When faced with the same situation with the Vandermonde matrix system, we introduced the Lagrange interpolating formula, which we used to explicitly construct the interpolating polynomial. This way, we avoided the ill-conditioning associated with inverting the Vandermonde matrix.

In a similar fashion, we can avoid the ill-conditioning associated with solving the heat diffusion problem by using an explicit formula for the solution to (1). Over the interval $[0, 1]$, the exact solution to (1) can be written in terms of two integrals as

$$u(x) = \left(a - \int_0^x \xi f(\xi)\, d\xi\right)(1 - x) + \left(b + \int_x^1 (\xi - 1) f(\xi)\, d\xi\right) x$$

where $u(0) = a$ and $u(1) = b$ are the prescribed boundary conditions.

(a) Use the trapezoidal rule on nodes $\xi_j = jh$, $h = 1/N$, $j = 0, 1, \ldots, N$ to numerically evaluate the above integrals. Use this numerical evaluation to get the solution to

$$u''(x) = -(2\pi)^2 \sin(2\pi x)$$

over $[0, 1]$. Assume boundary conditions $a = b = 0$. Plot your solution along with the exact solution on the same graph. Add a title, axis labels and a legend.

(b) Show that for $N = 100$, you get the same answer for $u(x)$ using the above formula as you got from Homework 3, Problem 1. To show this, you can print out the value of `norm(u_integral-u_hmwk3,Inf)`, for example, or demonstrate some other way that your answers are essentially identical.

(c) (**Extra Credit - 10 points**) Use `logspace` to create a range of N values from $N = 100$ to $N = 2 \times 10^5$. Loop over these N values and evaluate the integral solution for each value of N. In a manner similar to what you did in Homework 3, compute an error produced by the integral solution for each N. Graph this error, and the error to that from inverting the matrix A.

You should see that the error for the integral solution decreases even for very large N, unlike the solution resulting from inverting the matrix A. **Note:** This problem could take an hour or more to run!

6. **Colormaps and 3D rendering :** For this problem, you are going to experiment with three dimensional rendering in Matlab. Below are two problems from different application areas. You only have to do one of the problems. You can choose to do the second one for extra credit.

(a) **Mt. St. Helens Volcano :** From the course website, download the data file `msh.region`, taken from a USGS digital elevation map (DEM) of the Mt. St. Helens volcano in Washington State. Also download the sample code `plot_msh.m`, which will get you started on visualizing the elevation map.

The sample code currently has a gray-scale colormap that is a somewhat boring. Your goal for this project is to design a more interesting colormap that produces an artistic rendering of the Mt. St.

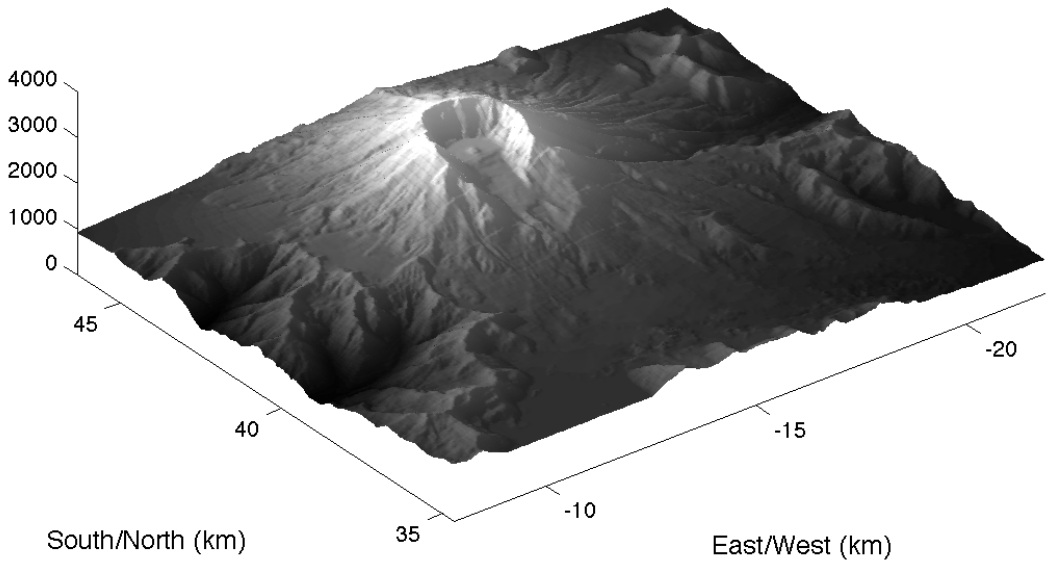

Figure 1: Grayscale rendering of Mt. St. Helens

Helen's volcano and surrounding region. You can choose to render the mountain topography so that it looks as realistic as possible (e.g. white for elevations above snow line, green for elevations below), or you can choose to render it in a way that shows some scientific quantity of interest (zones affected by the 1980 eruption, south facing slopes, and so on). To get started on designing your colormap, read the documentation on Matlab's `colormap` function. See Figure 2 for an example.

(b) (**Plotting three dimensional solids**) Often when visualizing the results of heat and mass transfer problems, you want to be able to plot a quantity such as temperature or pressure on the surface of a solid.

Download the sample file **plot_shapes.m** from the course website. This code shows you how you can plot simple solids (a cylinder and a sphere), and then color them using a scalar value that depends on the location on the surface of the shape. Your job is to modify this code to design your own arrangement cylinders, spheres or other shapes you construct and then color them with a background scalar field.

Example You might create a ring of cylinders, each with a sphere on top, and place a heat source at the center of the configuration. The three-dimensional temperature field could be then described as a function of the distance of a point on the surface of a solid from the heat source.

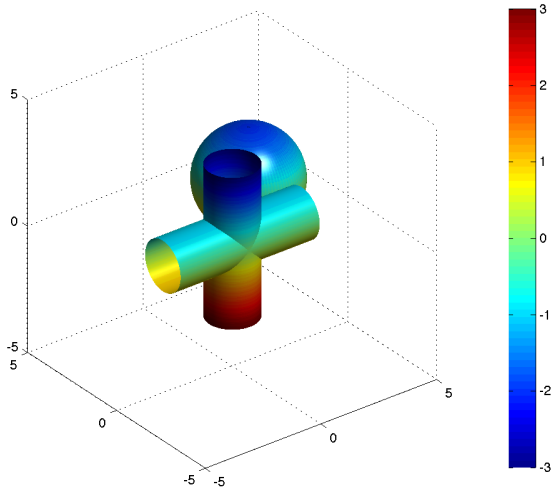

Figure 2: Temperature field with heating on the floor, imposed on solids.

You can either include your modified code in the same file with your other problems or publish it separately.

7. (**Differential Equations**) Use the Matlab function `ode45` to solve the following scalar ODEs.

   (a)
   $$\begin{aligned} y'(t) &= -10y, \qquad \text{over} \qquad t \in [0,1] \\ y(0) &= 1 \end{aligned} \tag{2}$$
   Plot the solution $y(t)$ verses t. On the same graph, plot the exact solution.

   (b)
   $$\mathbf{y}'(t) = \begin{bmatrix} 0 & 2 \\ -2 & 0 \end{bmatrix} \mathbf{y}, \qquad \text{over} \qquad t \in [0, \pi] \tag{3}$$
   where $\mathbf{y(t)} = (y_1(t), y_2(t))$, $y_1(0) = 1$, $y_2(0) = 0$. Plot the solution as a paramterized curve $(y_1(t), y_2(t))$. Plot the exact solution on the same graph.

   (c) (**Extra Credit (15pts):**) Write your own numerical method to solve the problem in (7b). Try each of the following methods :

   i. The *Forward Euler* method

   ii. The *Backward Euler* method

   iii. The *4th order Runge-Kutta* method

   Plot the solutions you get for the above on three separate graphs, along with the exact solution. Which methods works the best?

8. (**Pursuit curves**) A fox is chasing a rabbit, who is happily following a path traced out by the parametric curve $\mathbf{R}(t) = (x_R(t), y_R(t))$. If the fox moves at the same speed as the rabbit (let's say speed 1), and always heads towards the rabbit, we can write down a differential equation for the fox's trajectory $\mathbf{F}(t) = (x_F(t), y_F(t))$ as
   $$\mathbf{F}'(t) = \| \mathbf{R}'(t) \| \, \frac{\mathbf{R} - \mathbf{F}}{\| \mathbf{R} - \mathbf{F} \|}$$

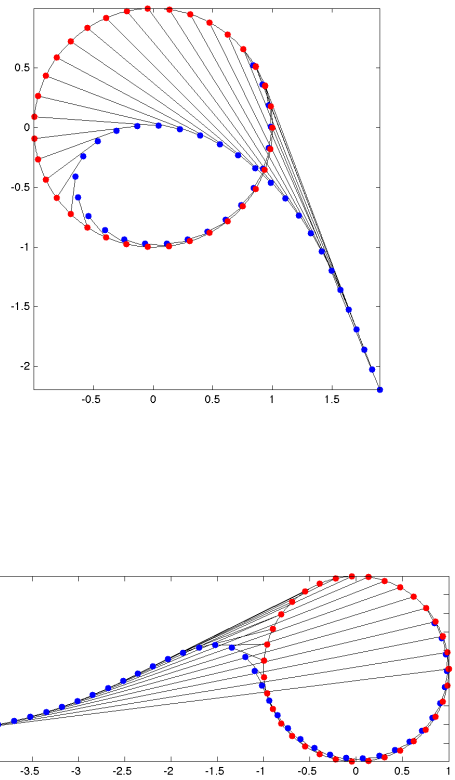

Figure 3: Pursuit curves from two different starting locations. The fox (blue) is chasing the rabbit (red). Both are moving at unit speed.

Suppose the rabbit runs in a circular path around a duck pond. This path is given by $\mathbf{R}(t) = (\cos(t), \sin(t))$. Solve for the pursuit path traced out by the fox, and plot your results. Use `ode45`. Try out different initial locations for the fox and the rabbit (assume the rabbit is always on the curve $\mathbf{R}(\mathbf{t})$). Turn in a plot of the pursuit curve for three different starting configurations.

To compute the norm of a vector, use `norm`. That is, $\|\mathbf{v}\| =$ `norm(v, 2)`. See Figure 3.

9. (**Plotting streamlines**) The streamlines for potential flow past a cylinder can be expressed as the curves of constant $\psi(x, y)$, given by

$$\psi(x, y) = U\left(r - \frac{R^2}{r}\right) y$$

where $r = \sqrt{x^2 + y^2}$, U is the mean velocity, and R is the radius of the cylinder.

Use the graphics functions `ndgrid` and `contour` to plot the streamlines of potential flow with a mean flow $U = 1$ past a cylinder of radius $R = 0.2$ in the domain $[-1, 1] \times [-1, 1]$. Use the `fill` function to plot a solid disk representing the cylinder. See Figure 4 to see what your final figure should look like.

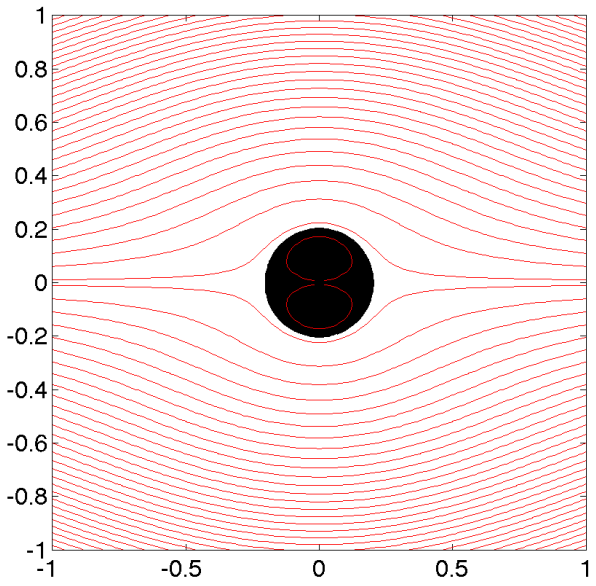

Figure 4: Streamlines of potential flow past a cylinder.