Name : _____

Homework #2

Math 566, Spring 2014

Due Friday April 4nd

Both problems 1 and 2 require substantial Matlab programming. You will find the following Matlab functions useful

- `ndgrid` - Constructs a two dimensional grid of x and y values. For example,
- `surf` - For plotting two 2d functions.
- `contour` - For creating contour plots
- `diff` - For computing finite difference stencils

Here is some sample code that uses the first two functions.

```
N = 160;
x = linspace(0,1,N+1);
y = x;
[xm,ym] = ndgrid(x,y);    % Create grids of x and y values.
f = -8*pi^2*sin(2*pi*xm).*cos(2*pi*ym);
hd = surf(xm,ym,f)
set(hd,'edgecolor','none');
% contour(xm,ym,f,30);     % Plot 30 contour lines.
print -dpng f_surf.png
```

And here is how you might use the **diff** function to compute a 2d finite difference stencil for the Laplacian of the $(N+1) \times (N+1)$ grid function u_{ij}. Here, `Au` is the value of the discrete Laplacian (using the 5 point stencil) applied to u.

```
ux_diff = diff(u(:,2:N),1);
uy_diff = diff(u(2:N,:),1,2);
Au(2:N,2:N) = (ux_diff(2:end,:) - ux_diff(1:end-1,:) + ...
        uy_diff(:,2:end) - uy_diff(:,1:end-1))/h^2;
```

1. Solve the two-dimensional Poisson problem

$$\nabla^2 u(x,y) = 8\pi^2 \sin(2\pi x)\cos(2\pi y), \quad (x,y) \in [0,1] \times [0,1]$$

subject to the exact solution

$$u(x,y) = u_{exact}(x,y) = -\sin(2\pi x)\cos(2\pi y)$$

on the boundary of the domain. Use the conjugate gradient method to solve the resulting linear system. Do not set up a matrix for the linear system, but rather write a *matrix-vector* multiply that can be called instead.

   (a) Show that you are getting second order accuracy by plotting your errors vs N on a log-log plot. You can use the inf-norm error, computed as

$$\| u_{exact} - u_{est} \|_\infty = \max_{1 \le i,j \le N+1} |u_{exact}(x_i, y_j) - u_{ij}|$$

   for a grid function u_{ij} on a range of grids. For example, you might use $N = 20, 40, 80, 160, 320, 640$.

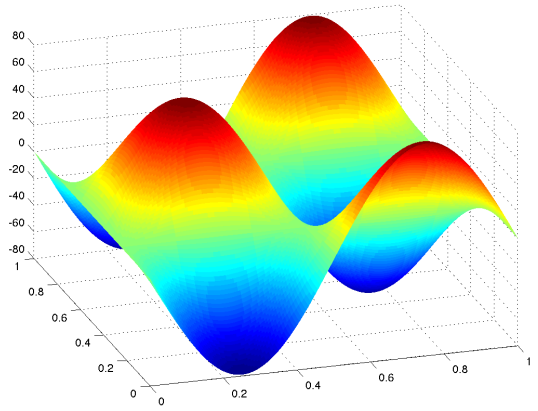

Figure 1: Right hand side function for Problem 1.

(b) On a semilogy plot, plot the number of iterations vs. N required by the conjugate gradient algorithm for this problem. What can you conclude about the efficiency of the conjugate gradient algorithm for this problem. For example, what is the approximate order of operations required to solve an $N \times N$ system? How does this compare to a direct solve using Gaussian Elimination for this system?

2. A common problem in engineering is to solve for a steady state temperature field in a medium with variable conductivity. We can model a two dimensional version of this problem with the 2d *variable coefficient* Poisson problem

$$\nabla\beta(x,y)\nabla u(x,y) = f(x,y)$$

where $\beta(x,y)$ represents the conductivity of the material. Heat will flow faster in regions of the domain where β is relatively high, and slower in regions with low β. The right hand side represents a heat source, such as a steady flame used to heat the material.

A particularly simple choice for β is a piecewise constant function. For example, we might divide the domain in half, and assign the left half a β^- value and the right half a β^+ value. On the domain $[0,1] \times [0,1]$, our β might take the form

$$\beta(x,y) = \begin{cases} \beta^- & x < 0.5 \\ \beta^+ & x \geq 0.5 \end{cases} \tag{1}$$

Another way to model this is with a smooth function

$$\beta(x,y) = \beta^- + (\beta^+ - \beta^-)\frac{1 + \tanh((x-0.5)/\varepsilon)}{2} \tag{2}$$

As $\varepsilon \to 0$, the smooth function (2) approaches the piecewise constant function (1).

For this problem, you will solve the variable coefficient Poisson problem on the domain $[0,1] \times [0,1]$ using the right hand side function

$$f(x,y) = -100\exp(-100((x-0.5)^2 + (y-0.5)^2))$$

Impose Dirichlet boundary conditions $u(x,y) = 1$ on the boundary of the domain and discretize the equation on a 160×160 mesh using the discretization approach described in equation 2.72 of your book (page 36).

β^+/ε	0.5	0.1	0.05	0.01	0.005	0
1	488	488	488	488	488	488
10	1104	1659	1725	–	–	–
100	1249	–	–	–	–	–
1000	1338	–	–	–	–	–

Table 1: Table of convergence rates for Problem 2 as function of β^+ (left column) and ε (top row). For this problem, $N = 160$ and $\beta^- = 1$. For $\varepsilon = 0$, use the piecewise constant defintion of $\beta(x, y)$ given in (1). For non-zero ε, use the smooth definition given in (2).

(a) Check that you have coded the matrix-vector multiply for the variable coefficient problem correctly by evaluating the truncation error in your discretization and showing that for the smooth $\beta(x, y)$, your truncation error is second order. Recall that the truncation error can be computed as

$$\tau \approx A\mathbf{u} - \nabla\beta(x, y)\nabla u(x, y)$$

for some function $u(x, y)$. You might choose $u(x, y)$ from Problem 1. Or another a simple polynomial which you can easily differentiate. For example,

$$u(x, y) = Ax^p + By^q$$

for integers p and q. Then

$$\nabla\beta(x, y)\nabla u(x, y) = \beta(x, y)\nabla^2 u(x, y) + \nabla\beta(x, y) \cdot \nabla u$$

Use the inf-norm to plot the truncation error vs. N, for a range of N values, and show that you are getting second order accuracy. **Note:** Just because the truncation error is second order isn't a guarantee in general that your global error will be second order. Recall that we need some sort of stability for this. But for this problem, we have stability, and so checking the truncation error does guarantee us global accuracy.

(b) One goal of this problem is to see how convergence of the conjugate gradient algorithm depends on both ϵ and on the jump $[\beta] = \beta^+ - \beta^-$. Set $\beta^- = 1$. To explore this dependence, vary ε and β^+, using the values

```
ewvec = [0.5 0.1 0.05 0.01 0.005 0];   % epsilon
bpvec = [1 10 100 1000];               % beta_plus
```

and complete Table 1 of convergence rates.

(c) Draw conclusions about how the convergence rate is affected by both the smoothness of $\beta(x, y)$ (as measured by ε) and the strength of the discontinuity, as measured by the ratio β^+/β^-.

(d) Plot a few representative figures showing the solution for $\varepsilon = 0$, and different values of β^+. Can you explain the resulting plots using your physical intuition about how heat flow behaves through materials of varying conductivity?

(e) The temperature field $u(x, y)$ obtained from the smooth $\beta(x, y)$ is second order accurate. However, the solution obtained using the piecewise constant $\beta(x, y)$ is only first order accurate. What goes wrong in this case?

3. (**Runge-Kutta Methods**) Prove the consistency requirement in Equation (5.35), page 126, from your book.

4. (**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),

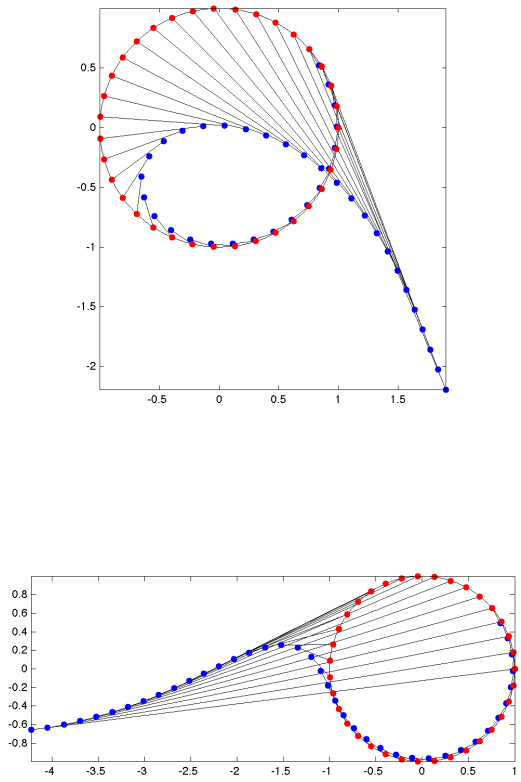

Figure 2: Pursuit curves from two different starting locations. The fox (blue) is chasing the rabbit (red). Both are moving at unit speed.

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

$$\mathbf{F}'(t) = \| \mathbf{R}'(t) \| \frac{\mathbf{R} - \mathbf{F}}{\| \mathbf{R} - \mathbf{F} \|}$$

Suppose the rabbit runs in a circular path around a duck pond. This path is given by $\mathbf{R}(t) = (\cos(t), \sin(t))$. Write a fourth order Runge-Kutta scheme to solve the pursuit path traced out by the fox, and plot your results. 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 2.