Computer Lab Assignment # 4
Using the Finite Difference Method to solve BVPs

Lab: Wednesday, 10/9, 10/16
Due: 8:30 a.m., Wednesday, 10/23
Submit write up, code and any auxiliary files to Dave Witman, dw11d@my.fsu.edu

Goals: In this lab you are asked to first write a finite difference code to approximation the solution of a two point BVP with homogeneous Dirichlet boundary conditions. You are asked to verify that your code is converging at the correct rate. Next you will modify your code to solve the same differential equation but with a homogeneous Neumann boundary condition. In the second part of this lab you are asked to extend your code from one dimension to solve the Poisson equation in two dimensions with Dirichlet boundary conditions. After you have verified that your results are converging at the correct rate, then you are asked to solve a problem which models the steady state temperature distribution in a thin plate. Lastly, you are asked to solve a Poisson equation where the source term \( f(x, y) \) is a set of discrete points rather than an explicit formula.

1. Consider the two point BVP

\[
-u''(x) + q(x)u = f(x) \quad a < x < b \quad u(a) = u_L, \quad u(b) = u_R
\]

Write a computer program which solves this problem using a second order accurate finite difference scheme and a uniform grid. Use your tridiagonal solver from you previous homework assignment to solve the resulting system.
a. Test your code on the problem where

\[
q(x) = x, \quad f(x) = 12x^2 - 20x^3 - x^5 + x^6, \quad a = 0, \quad b = 1, \quad u_L = u_R = 0
\]

whose exact solution is \( u(x) = x^4(x - 1) \). Compute the normalized \( \ell_2 \)-norm (standard Euclidean norm) of the error vector \( \vec{E} \) where \( E_i = |u(x_i) - U_i|, \) \( x_i \) are the grid points and \( U_i \approx u(x_i) \). Normalize by the \( \ell_2 \) norm of the exact solution at the grid points, i.e., calculate

\[
\frac{\|\vec{E}\|_2}{\|\vec{u}\|_2}
\]

where \( \vec{u} \) is the exact solution at the grid points. Tabulate your errors for \( h = 1/4, 1/8, 1/16, 1/32 \) and \( 1/64 \) and determine the numerical rate of convergence. Compare this with the theoretical rate.

b. In (a) you used your tridiagonal solver for a symmetric positive definite matrix to solve the resulting system where you stored your matrix as two vectors or as an \( n \times 2 \) or \( 2 \times n \) matrix. This is how one would store and solve a matrix if Matlab is not used. However, we will not always be able to use this specific solver but we will typically have banded
matrices with bandwidth greater than three. In Matlab, it is easy to construct a banded matrix. There are several ways to do this but a straightforward way (but not the most efficient) is to declare the matrix as a sparse matrix and then you enter the elements as usual and can use the backslash command. For example, to set up the tridiagonal with 2 on the diagonal and -1 on the off diagonals we could use the following commands.

```matlab
a = sparse([],[],[],n,n,3*n);
a(1,1)=2.; a(1,2)=-1.;
for i=2:n-1
    a(i,i)=2.; a(i,i-1)=-1.;a(i,i+1)=-1;
end
a(n,n-1)=-1; a(n,n)=2.;
```

Make the modifications in your code in (a) to use Matlab’s sparse storage and solve using the backslash command. You can either use the way I suggested or if you are familiar with the command `spdiags` you can use this because it is more efficient. Rerun your example for \( h = 1/8 \) and make sure you are getting the same results as in (a).

c. Now we want to modify our BVP to handle a homogeneous Neumann boundary condition at \( x = 0 \), i.e., \( u'(0) = 0 \). This means we have an unknown \( U_i \) for all points except the right boundary where \( u(1) = 0 \). Use a centered difference to approximate the boundary condition \( u'(0) = 0 \), i.e.,

\[
  u'(0) \approx \frac{u(h) - u(-h)}{2h} \quad \text{or in terms of } U \quad \frac{U_1 - U_{-1}}{2h} = 0
\]

where we have introduced the fictitious point \( x_{-1} = -h \) since our domain begins at \( x = 0 \). Write the difference equation at \( x_0 = 0 \) which will include \( U_{-1} \); then use your boundary equation to substitute in an expression for \( U_{-1} \). Indicate the modification to your matrix and right hand side (if any) that must be made to satisfy this problem. Is your matrix still symmetric? If not, you can no longer use your solver in (a). Solve your problem for the same BVP in (a) except with \( u'(0) = 0 \). Note that the exact solution \( u(x) = x^4(x-1) \) satisfies \( u'(0) = 0 \). Output your errors as in (a).

2. Make a copy of your code for one dimension and modify it to solve Poisson’s equation in two dimensions using a second order finite difference scheme on a uniform grid \( \Delta x = \Delta y = h \). Assume that Dirichlet boundary conditions are imposed. Test out your code on the following three BVPs, tabulate your errors for \( h = 1/4, 1/8, 1/16, 1/32 \) and \( 1/64 \) and determine the numerical rate of convergence. Discuss your results and compare with the theoretical rate.

(i) \(- \Delta u = 2\pi^2 \sin(\pi x) \sin(\pi y) \) in \( (0,2) \times (0,2) \) \( u = 0 \) on \( \Gamma \)

whose exact solution is \( u(x,y) = \sin(\pi x) \sin(\pi y) \).

(ii) \(- \Delta u = -2(y^2 - y) - 2(x^2 - x) \) in \( (0,1) \times (0,1) \) \( u = 0 \) on \( \Gamma \)

whose exact solution is \( u(x,y) = (x^2 - x)(y^2 - y) \).
\[(iii) \quad - \Delta u = 2\pi^2 \sin(\pi x) \cos(\pi y) \quad \text{in } (0, 1) \times (0, 1) \]

\[u = 0 \text{ for } x = 0, 1 \text{ and } u = \sin(\pi x) \text{ for } y = 0 \text{ and } u = -\sin(\pi x) \text{ for } y = 1\]

whose exact solution is \(u(x, y) = \sin(\pi x) \cos(\pi y)\).

3. The Poisson equation in two dimensions can be used to model the steady state temperature distribution in a thin plate. Suppose we have a square domain of side one and the temperature is held fixed at each side of the domain as follows:

\[u = 0^\circ \text{ on the bottom of the plate } (y = 0)\]

\[u = 50y^\circ \text{ on the left side of the plate } (x = 0)\]

\[u = 75y^\circ \text{ on the right side of the plate } (x = 1)\]

\[u = (-150x^2 + 175x + 50)^\circ \text{ on the top of the plate } (y = 1)\]

Assume that there is a temperature source \(f(x, y) = 2\pi^2 \sin(\pi x) \cos(\pi y)\). Now that you have confidence that your code is running correctly, solve this problem with \(h = 1/32\). What is the approximate temperature in the center of the plate? Output a contour plot of the temperature. Choose several contours ranging from your minimum to maximum temperature so that your plot is readable. Label your contours or provide a legend.

4. (Grads only, UG for extra credit) First we will load the Matlab data set called penny. The data was obtained in 1984 from the National Bureau of Standards by an instrument that makes precise measurements of the depth of a mold used to mint the U.S. one cent coin. The data is automatically loaded into a variable called \(P\) and is an \(128 \times 128\) array. The following commands access the file

```matlab
load penny
P=flipud(P)
```

a. Make a contour plot of this data.

b. We now want to solve the Poisson equation with homogeneous Dirichlet data using this discrete penny data for the right hand side \(f(x, y)\) in our PDE defined on a square. (It doesn’t really matter what the dimensions are because these are artificial coordinates.) Set up a grid where you use all of the data and solve the Poisson equation. Output your data in a contour plot.

c. What do we do if we want to refine the mesh? We only have data on a \(128 \times 128\) grid and suppose now we want it on a \(257 \times 257\) grid, i.e., we want to add data at all the midpoints of the initial grid. We can use the Matlab command `interp2` to do this. Look up the command and notice that you can use linear or cubic interpolation; we will use
cubic. The write up is a bit confusing but what we want to do is first set up an $xy$ grid where all the data is given and then set up one where we want the data. For example

$$
x = 1:128; \quad y = x; \quad [X,Y] = \text{meshgrid}(x,y);
$$

will get the grid where the data is given. We do the analogous thing to get the interpolated grid $X_I, Y_I$ which we want at $x = 1, 1.5, 2, 2.5,$ etc. and then use the command `interp2` with the cubic option.

Perform cubic interpolation on your penny data and do a contour plot so that you are confident you have done it correctly. Now solve the Poisson equation from (b) on this finer grid, output your data in a contour plot.