OCEN 689
Fall 2019

Homework 2, due Tue Oct 1

In this assignment you will write a 1D and a 2D numerical solver for steady-state non homogeneous diffusion equations. You will experiment with different types of boundary conditions, with variable diffusion coefficients and with different source terms. You will also learn how to set up the corresponding systems of equations, which is not so hard to do in 1D, but a bit trickier in 2D (we will take you through step-by-step).
Some guidelines for coding assignments

- Please submit all code used in the assignment, including any function defined by you.
- All codes must be properly commented and understandable variable names must be used.
- All figures must have readable title and axis labels. If more than one plot is given in the same figure, a legend should be provided and colors used to distinguish the plots.

1. **Finite Differences in 1D**

   We would like to solve the steady non homogeneous diffusion equation

   $$-c(x)u'' + u = 2x, \quad x \in [0, 1].$$

   To find a unique solution, we must prescribe two boundary conditions. We start with the Dirichlet boundary conditions $u(0) = u(1) = 0$.

   We solve this equation using finite differences on an equidistant grid with grid step size h. The grid has $N + 1$ grid points $x_0, x_1, \ldots, x_N$, with $N = 1/h$.

   The second order derivative is discretized using the finite difference approximation

   $$u''(x_i) \approx \frac{u(x_{i-1}) - 2u(x_i) + u(x_{i+1})}{h^2}, \quad i = 1, \ldots, N - 1.$$

   We start by making our lives easier and setting $c(x) = 1$.

   (a) Find the *exact* solution to the differential equation. We will use this solution later to compute errors in our discrete solutions. Also compute the exact value of $\int_0^1 u(x)dx$.

   (b) Write down the resulting system of linear equations for the unknowns u_1 through u_{N-1} in the form $A\vec{u} = \vec{b}$, where $\vec{u}$ contains the unknowns and A is a $(N-1)\times(N-1)$ matrix. Explain how you incorporate the boundary conditions.

   (c) Write a MATLAB code that can solve the system efficiently for large N. To allow this, you must use MATLAB's `sparse` matrix environment.

(d) Run the code for $N = 5, 10, 20, 100, 500$ and 1000. Plot the solutions for $N = 5, 10$ and 100, together with the exact solution.

(e) For all values of N, compute the error in the numerical solution (as compared to the exact solution) using the $2-$norm, the $1-$norm and the maximum norm. Plot the errors as a function of h in a log-log plot. For the $2-$norm, what is the slope of the error plot? Is this as expected? Explain.

(f) Use the Trapezoidal rule and Simpson's rule to compute an approximation to $\int_0^1 u(x)dx$ for all values of N. Plot the errors in these numerical approximations as a function of h in a log-log plot. What is the slope of the error plot? Is this as expected? Explain.

(g) Let's change the boundary conditions to $u'(0) = 0, u(1) = 2$. How can you incorporate the Neumann boundary condition in the numerical scheme? Give the new A and $\vec{b}$.

(h) We now make $c(x)$ a bit more exciting and set it to $c(x) = 1+2x(1-x)$. How would you treat this variable diffusion coefficient? Give the resulting system $A\vec{u} = \vec{b}$.

(i) Create a new version of your code for the new boundary conditions and new $c(x)$ (always a good idea to save old working versions of a code instead of overwriting them). Solve the system for $N = 5, 10, 20, 100, 500$ and 1000. Plot the solutions for $N = 5, 10, 100$ and $N = 1000$. Here, we will use the solution for $N = 1000$ as reference solution (we will consider it the "exact" solution - of course your code needs to converge before we can do this!). Compute the $2-$norm of the errors for $N = 5$ through $N = 100$ and plot these as a function of h in a log-log plot. Check the slope of this plot. Is it as expected?

(j) Now we take $c(x) = 1 + 50e^{-(x-1/2)^2}$. Compute solutions for $N = 5, 10, 20$ and 100. Plot the corresponding numerical solution for $N = 100$. What do you observe? Estimate the derivative of the solution at $x = 0$ for $N = 5, 10, 20$ and 100 and plot them as a function of h. Explain how you estimate the derivative and comment on the observed behavior in the plot.

(k) **Extra credit** How would the numerical scheme change if we were solving

$$-(c(x)u)'' + u = 2x, \quad x \in [0, 1] \quad u'(0) = 0, u(1) = 2.$$

repeat the last section and plot the numerical solution for $N = 10, 20, 100, 500$

2. **Finite differences in 2D**

   We will now extend to two dimensions. We consider the equation

   $$c(x,y)\Delta u + u = 2x^2 \sin(y), \quad (x,y) \in [0,1] \times [0,1],$$

   where Δ corresponds to the Laplacian operator. We start with homogenous Dirichlet boundary conditions everywhere on the boundary, that is $u(x,0) = u(x,1) = u(0,y) = u(1,y) = 0$, and $c(x,y) = 1$.

   (a) We will discretise the equation on a Cartesian grid with grid step size h in both the $x-$ and $y-$directions. Thus, we have $N \times N$ grid cells in the domain, with $N = 1/h$. The discrete solution in the point (ih, jh) is denoted as $u_{i,j}$. Give a second order discretisation of Δu on this grid.

   (b) In 1D, we write the system of discrete equations in the form $A\vec{u} = \vec{b}$. Here, we want to do the same thing. However, now our unknowns live in 2D. They are $u_{i,j}$, $i,j = 1,\ldots,N-1$. To allow the desired matrix-vector form, we need to store these unknowns in a 1D vector of length $(N-1)\times(N-1)$. A straightforward way to do this is to store the unknown row by row: the first $N-1$ elements of $\vec{u}$ are the unknowns $u_{i,1}$ for $i = 1, \ldots N$, the next $N-1$ elements are the unknowns $u_{i,2}$, etc. Of course, we could also store the unknowns column by column. You are free to use either ordering system. Choose your favorite ordering and find the corresponding A and $\vec{b}$. Note that the matrix A will be of size $M \times M$, with $M = (N-1)^2$.

   (c) Write a MATLAB code to solve the system of equations for general N. Again, use sparse MATLAB. You may find the MATLAB functions `meshgrid,kron,reshape` useful.

   (d) Use your code to solve the system for $N = 5, 10, 20, 40$ and $N = 100$. The solution for $N = 100$ will be used as reference solution in lieu of an exact solution. Plot your solutions using the function `pcolor` for $N = 5$ and $N = 40$.

   (e) Compute the $2-$norm of the error for $N = 5, 10, 20, 40$ and plot as a function of h in a log-log plot. What is the slope of the plot and is it expected?

   (f) Now take $c(x,y) = 1 + 50e^{-(x-1/2)^2}$. Update the system of equations and solve for $N = 100$. Plot the solution. What do you observe? Comment.

   (g) Take $c(x,y) = 1 + |\eta(x,y)|$, where η is random correlated Gaussian noise with mean 0. You can use the following code to construct η

   ```
   [Xh,Yh]=meshgrid(-0.25:0.01:0.25,-0.25:0.01:0.25);
   eta=randn(n);
   eta=conv2(eta,exp((-Xh.^2-Yh.^2)/(2*0.2^2)),'same');
   ```

   Update your system and solve for $N = 10$ and $N = 40$. Plot the solutions. What do you observe? Comment.