CS 378: Programming for Performance
Assignment 2: Computational science methods
Due: February 17th

February 14, 2015

Late submission policy: Submission can be at the most 2 days late. There will be a 10% penalty for each day after the due date (cumulative).

The code for the plots and experimentation must be written in MATLAB, which is available on the public machines in the CS department.

1. (Iterative solution of linear systems) Consider the linear system

   $$3x - 4y = -1$$
   $$x + 2y = 3$$

   (a) Write down the recurrence relation that corresponds to solving this system using the Jacobi method, starting with the initial approximation ($x_1 = 0, y_1 = 0$). Use the first equation to refine the approximation for x and the second equation to refine the approximation for y. Express this recurrence as a computation involving matrices and vectors.

   (b) Compute the first 25 approximations (x_i, y_i) and plot a 3D plot (x, y, i) in which the z-axis is the iteration number i. Give an intuitive explanation of this 3D plot.

   (c) Estimate experimentally how many iterations you need to obtain an approximate solution in which both x and y are within 1% of the exact solution.

   (d) Repeat these three parts for the Gauss-Seidel method. You can find a description of the Gauss-Seidel method in many online resources - Wikipedia has a good description.

2. (ODE's) Consider the differential equation

   $\frac{d^2y}{dx^2} = -y$

   with initial conditions $y(0) = 0, y'(0) = 1$.

(a) Show that the exact solution to this equation is $y = sin(x)$.

(b) Consider the numerical solution of this equation using the backward-Euler approximation
$\frac{d^2y}{dx^2}|_{i\Delta x} \approx \frac{y_i - 2y_{i-1} + y_{i-2}}{\Delta x^2}$
Suppose we wish to compute the value of $y(10 * pi)$. Divide the interval $[0, 10 * pi]$ into N steps (we will vary N), and use the finite-difference formula to calculate an approximation to $y(10 * pi)$ for different values of N. What is the calculated value of $y(10 * pi)$ when $N = 10$? When $N = 100$? Estimate experimentally how large N must be for your computed value to be within 1% of the exact value.

(c) Repeat the previous computations using the centered-differences approximation
$\frac{d^2y}{dx^2}|_{i\Delta x} \approx \frac{y_{i+1} - 2y_i + y_{i-1}}{\Delta x^2}$
Compare and contrast backward-Euler approximation and centered-differences approximation using the plots.

3. (PDE's) In this problem, we will solve the one-dimensional diffusion equation, which models how heat spreads through a material of uniform conductivity and similar problems. The diffusion equation is usually written as follows
$\frac{\delta f}{\delta t} = D\frac{\delta^2 f}{\delta x^2}$
The solution $f(x, t)$ depends on both x and t. Assume that we have a rod of length 10 meters, and that the two ends of the rod are kept at a fixed temperature of 0^oC, so $f(0, t) = 0.0$ and $f(10, t) = 0.0$. Assume that initially the temperature in the interior of the rod is $f(x, 0) = e^{-4(x-5)^2}$.

The approximate solution $\hat{f}$ can be thought of a two-dimensional array in which $\hat{f}(i, j) \approx f(i\Delta x, j\Delta t)$. Intuitively, $\hat{f}(i, j)$ is the computed solution after j time steps at a spatial position i spatial steps away from the origin.

(a) Compute the array $\hat{f}$ using the following discretization scheme:
$\Delta x = 0.25$
$\Delta t = 0.025$
$D = 1$
$0 \leq x \leq 10$
$0 \leq t \leq 10$
$\frac{\delta f}{\delta t}|_{(i*\Delta x, j*\Delta t)} \approx \frac{\hat{f}(i,j+1) - \hat{f}(i,j)}{\Delta t}$ (Forward-Euler)
$\frac{\delta^2 f}{\delta x^2}|_{(i*\Delta x, j*\Delta t)} \approx \frac{\hat{f}(i+1,j) - 2*\hat{f}(i,j) + \hat{f}(i-1,j)}{\Delta x^2}$ (Centered differences)
Plot the temperatures for a number of different time steps on the same graph (so the x-axis is the distance from the origin in the rod and the y-axis is the temperature). Does this graph jive with your intuition?

(b) Change the time step to 0.050 and repeat the previous steps. Explain your observations briefly.

4. (PDE's) In the previous problem, we used the forward-Euler method to discretize time and centered differences to discretize space. As you should have observed in that problem, this discretization scheme is stable only if the time step is below some critical threshold. For this reason, an implicit scheme called the Crank-Nicolson method is often used to discretize time when solving the heat equation. The Crank-Nicolson scheme is similar to the well-known trapezoidal method for integration that you may have learnt in freshman calculus. A good description of the Crank-Nicolson method can be found online - Wikipedia has a good description.

   Repeat the previous problem using the Crank-Nicolson method. Experiment with different time-steps. Does the instability you should have observed for forward-Euler discretization show up for this method?

   PS: As we discussed in class, implicit methods usually require solving linear systems. You do not have to write code for this - use the built-in Matlab functions for solving linear systems.

Deliverables: Submit (to canvas) your answers, MATLAB code and graphs for all problems. No credit will be given for problems in which you are asked to write code unless you include your code and we can run it.