Math 228B
Homework 2
Due Friday, 2/17

1. Consider the forward time, centered space discretization
$$\frac{u_j^{n+1}-u_j^n}{\Delta t}+a\frac{u_{j+1}^n-u_{j-1}^n}{2\Delta x}=b\frac{u_{j-1}^n-2u_j^n+u_{j+1}^n}{\Delta x^2},$$
to the convection-diffusion equation,
$$u_t+au_x=bu_{xx},\quad b>0.$$
   (a) Let $\nu=a\Delta t/\Delta x$ and $\mu=b\Delta t/\Delta x^2$. Since the solution to the PDE does not grow in time, it seems reasonable to require that the numerical solution not grow in time. Use von Neumann analysis to show that the numerical solution does not grow (in 2-norm) if and only if $\nu^2\le 2\mu\le 1$.
   (b) Suppose that we use the mixed implicit-explicit scheme
$$\frac{u_j^{n+1}-u_j^n}{\Delta t}+a\frac{u_{j+1}^n-u_{j-1}^n}{2\Delta x}=b\frac{u_{j-1}^{n+1}-2u_j^{n+1}+u_{j+1}^{n+1}}{\Delta x^2}.$$
   Use von Neumann analysis to derive a stability restriction on the time step.

2. Write programs to solve
$$u_t=\Delta u \text{ on } \Omega=(0,1)\times(0,1)$$
$$u=0 \text{ on } \partial\Omega$$
$$u(x,y,0)=\exp\left(-100((x-0.3)^2+(y-0.4)^2)\right)$$
to time $t=1$ using forward Euler and Crank-Nicolson. For Crank-Nicolson use a fixed time step of $\Delta t=0.01$, and for forward Euler use a time step just below the stability limit. For Crank-Nicolson use Gaussian elimination to solve the linear system that arises, but make sure to account for the banded structure of the matrix.
   (a) Time your codes for different grid sizes and compare the time to solve using forward Euler and Crank Nicolson.
   (b) In theory, how should the time scale as the grid is refined for each algorithm? How did the time scale with the grid size in practice?
   (c) For this problem we could use an FFT-based Poisson solver which will perform the direct solve in $\mathcal{O}(N\log(N))$, where N is the total number of grid points. We could also use multigrid and perform the solve in $\mathcal{O}(N)$ time. How should the time scale as the grid is refined for Crank-Nicolson if we used an $\mathcal{O}(N)$ solver?
   I will make a FFT-based solver available if you want to experiment with it.

At the end of this assignment is MATLAB code to form the matrix for the 2D discrete Laplacian.

```
%
% lap2d.m
%
%   form the (scaled) matrix for the 2D Laplacian for Dirichlet boundary
%   conditions on a rectangular node-centered nx by ny grid
%
%   input:  nx -- number of grid points in x-direction (no bdy pts)
%           ny -- number of grid points in y-direction
%
%   output: L2 -- (nx*ny) x (nx*ny) sparse matrix for discrete Laplacian
%
function L2 = lap2d(nx,ny);

    % make 1D Laplacians
    %
    Lx = lap1d(nx);
    Ly = lap1d(ny);

    % make 1D identities
    %
    Ix = speye(nx);
    Iy = speye(ny);

    % form 2D matrix from kron
    %
    L2 = kron(Iy,Lx) + kron(Ly,Ix);

%
% function: lap1d -- form the (scaled) 1D Laplacian for Dirichlet
%                    boundary conditions on a node-centered grid
%
%   input:  n -- number of grid points (no bdy pts)
%
%   output: L -- n x n sparse matrix for discrete Laplacian
%
function L = lap1d(n)
    e = ones(n,1);
    L = spdiags([ e -2*e e], [-1 0 1],n,n);
```