1. Consider the parabolic PDE

\[ u_t = u_{xx}, \quad x \in [0, 1], \quad t \geq 0, \]
\[ u(x, 0) = \sin \pi x, \]
\[ u(0, t) = u(1, t) = 0. \]

The true solution is \( u(x, t) = e^{-\pi^2 t} \sin \pi x. \)

(a) Write a program based on the formula

\[
\frac{u_{i,j+1} - u_{i,j}}{k} = \lambda \frac{u_{i+1,j+1} - 2u_{i,j+1} + u_{i-1,j+1}}{h^2} + (1 - \lambda) \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h^2},
\]

where \( h = \Delta x, \quad k = \Delta t. \)

For each of the following cases, run the program for \( t = 0 \) up to \( t = 1, \) for \( n = 10, 20, 40, \)
\( h = 1/n, \) and find the maximum error at the grid points. Here error = absolute value of (true solution - numerical solution).

(b) Crank-Nicolson (\( \lambda = 1/2 \)) with \( k = h. \) Theory tells us that this will be stable.

(c) Forward Euler (\( \lambda = 0 \)) with \( k = h^2/2. \) This will also be stable.

(d) Forward Euler with \( k = h/2. \) Theory tells us that this will be unstable.

2. Consider the hyperbolic PDE

\[ u_{tt} = u_{xx}, \quad x \in [0, 1], \quad t \geq 0, \]
\[ u(x, 0) = \sin \pi x, \]
\[ u_t(x, 0) = 0, \]
\[ u(0, t) = u(1, t) = 0. \]

The true solution is \( u(x, t) = \cos(\pi t) \sin(\pi x). \)

(a) The finite difference formulas are

\[
\frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{k^2} = \lambda \frac{u_{i+1,j+1} - 2u_{i,j+1} + u_{i-1,j+1}}{h^2} + (1 - 2\lambda) \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h^2} + \lambda \frac{u_{i+1,j-1} - 2u_{i,j-1} + u_{i-1,j-1}}{h^2},
\]

where \( h = \Delta x, \quad k = \Delta t. \) This works out to the form

\[ A u_{j+1} = B u_j + C u_{j-1}. \]

Figure out what the matrices \( A, B, C \) are. They depend on \( \lambda \) and on \( m = k/h. \)

(b) 1 point extra credit) Assume the initial conditions are \( u(x, 0) = f(x), \quad u_t(x, 0) = g(t). \)

Obviously \( u_0 = f, \) but we also need \( u_1 \) to get going. Derive the formula

\[ (A - C)u_1 = Bf - 2kCg. \]

(c) Write a computer program based on the formulas from above.

For each of the following cases, run the program for \( t = 0 \) up to \( t = 1, \) for \( n = 10, 20, 40, \)
\( h = 1/n, \) and find the maximum error at the grid points. Here error = absolute value of (true solution - numerical solution).

(d) Implicit method with \( \lambda = 1/2, \quad k = h. \) Theory tells us that this will be stable.

(e) Explicit method (\( \lambda = 0 \)) with \( k = h^2/2. \) This will also be stable.

(f) Explicit method with \( k = 2h. \) Theory tells us that this will be unstable.

Let \( P_n \) be the matrix which results from the finite difference discretization of Laplace's equation \( \nabla^2 u = 0 \) or Poisson's equation \( \nabla^2 u = f \) in two dimensions. \( P_n \) is a block tridiagonal
matrix of size \((n-1)^2 \times (n-1)^2\), consisting of \((n-1) \times (n-1)\) blocks of size \((n-1) \times (n-1)\) each. The diagonal blocks are of the form
\[
\begin{pmatrix}
4 & -1 \\
-1 & 4 & -1 \\
& \ddots & \ddots & \ddots \\
& & -1 & 4
\end{pmatrix},
\]
the others are negative identity matrices. In terms of the notation I used in class, \(P_n = -h^2 L_h\).

I have posted a Matlab toolbox called templates on the course web site; it comes from the book Templates for the Solution of Linear Systems. Download the toolbox and install it somewhere in your Matlab path. I added two routines (poisson and cgssor) myself. They will not be mentioned in the documentation. Just use help.

3. Solve the Poisson equation
\[-\nabla^2 u = 1 \quad \text{in } [0,1] \times [0,1]\]
\[u(x,0) = 0 \]
\[u(x,1) = 0 \]
\[u(0,y) = 0 \]
\[u(1,y) = 0 \]

This problem is about experimenting with various solvers. The matrix is generated by routine poisson, and the right-hand side is just a vector of ones, except that a factor of \(h^2\) needs to go in there somewhere.

I don’t know what the correct answer is. The maximum value of the numerical solution is in the center, and it is near 0.074.

Use \(n = 500\) for all parts. For the iterative methods, use \(u_0 = (0,\ldots,0)^T\) (initial guess), \(\text{maxiter} = 1000\) (maximum number of iterations), \(\text{tol} = 10^{-5}\) (error tolerance).

(a) Set up the equation and solve it directly with \(u = P\backslash b\). You need to use a sparse matrix unless you have a terabyte of main memory. It is enough to make \(P\) sparse, the rest can be full vectors or matrices.

What is the execution time (\texttt{tic} and \texttt{toc})? Print out the value of \(u(1/2,1/2)\). As mentioned above, that should be near 0.074.

Plot the numerical solution as a surface. If you plot all the values, you get a picture that is solid black, because the boundary lines of the surface tiles cover everything. Reduce the number of points to the equivalent of \(n = 50\). You can do that by either recomputing the answer for \(n = 50\), or by simply using only every 10th row and column in your answer for \(n = 500\).

(b) The eigenvalues of the Jacobi iteration matrix \(J_n = -D^{-1}(L + U)\) for \(P_n\) are
\[
\lambda_{k\ell} = \frac{1}{2} \left[ \cos \frac{k\pi}{n} + \cos \frac{\ell\pi}{n} \right], \quad 1 \leq k, \ell \leq n - 1.
\]

Find the spectral radius \(\rho(J_n)\), and determine the optimal \(\omega\) for SOR.

Estimate how many Jacobi iterations it would take to reduce the initial error by a factor of \(10^{-5}\). Same for SOR with optimal \(\omega\).

(c) Solve the equation using SOR with optimal \(\omega\). The optimal \(\omega\) is close to 2. If you couldn’t figure it out in part (b), use \(\omega = 1.99\).

What is the execution time? How many iterations does it take? This number should be approximately the same as your estimate from (b). Find the maximum difference (in absolute value) at the grid points between the solution from (a) and this one. This should be of the order of \(10^{-4}\) or \(10^{-5}\), since that is the accuracy we requested.

(d) Solve the equation using Conjugate Gradients without preconditioning. Routine \texttt{cg} insists on a preconditioning matrix; use a (sparse) identity matrix. What is the execution
time? How many iterations does it take? Find the maximum difference between the solution from (a) and this one.

(e) Solve the equation using Conjugate Gradients with SSOR preconditioning for the optimal \( \omega \) (routine \texttt{cgssor}). What is the execution time? How many iterations does it take? Find the maximum difference between the solution from (a) and this one.

(f) (2 points extra credit) Solve the equation using Gauss-Seidel. For that you will have to raise the limit on number of iterations, and maybe you should go for lunch while it runs. What is the execution time? How many iterations does it take? It should take about half the number of Jacobi iterations you calculated in (b). Find the maximum difference between the solution from (a) and this one.

Part of the extra credit consists in figuring out where the GS routine is hiding in the \texttt{templates} toolbox. Think.

4. (Extra Credit) Consider the simple boundary value problem with periodic boundary conditions

\[
\begin{align*}
    u_{xx}(x) &= f(x), & x &\in [0, 1], \\
    u(0) &= u(1), \\
    u'(0) &= u'(1).
\end{align*}
\]

Derive the finite difference equations for this. This should be an \( n \times n \) system for \( u_0, \ldots, u_{n-1} \). You can assume \( f \) is also periodic.