Numerical Analysis 1 – Class 5

Friday, February 10th, 2023

Subjects covered

- Finite difference derivatives.
- Iterative solvers for Ax = b: Jacobi and Gauss-Seidel.
- Ax = b as a minimization problem (SPD matrices).
- Method of gradient descent (steepest descent).
- Conjugate gradient method and its offspring.
- Preconditioning.

Readings

- Kutz, Chapter 2.
- “Notes on Conjugate Gradient”, by S. Brorson (on Canvas).
- “An Introduction to Conjugate Gradient Without the Agonizing Pain”, by J. R. Shewchuk. (on Canvas)

Problems

Most of the following problems require you to write a program. For each program you write, please make sure you also write a test which validates your program.

Problem 1

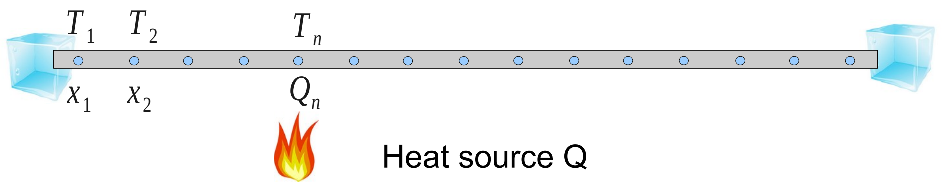
Recall the iteration matrix used in the Jacobi demo presented in class,

$$(D^{-1}N)=\begin{bmatrix} 0 & -1/2 & 0 & 0 & 0 & \cdots \\ -1/2 & 0 & -1/2 & 0 & 0 & \cdots \\ 0 & -1/2 & 0 & -1/2 & 0 & \cdots \\ 0 & 0 & -1/2 & 0 & -1/2 & \cdots \\ 0 & 0 & 0 & -1/2 & 0 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

Prove that the eigenvalues of this matrix are cosines, and therefore satisfy $|\lambda_p|<1$ for all *p*. You may complete the argument started in class: make a guess for the eigenvectors, $u_p = A_p \sin(\theta_n) + B_p \cos(\theta_n)$, then work out the corresponding eigenvalue. This is a pencil-and-paper exercise – please turn in your written work. Hint: assume $\theta_n = p\pi n/(N+1)$, where *p* is the eigenvalue number, *n* is the index into the eigenvector, and *N* is the matrix size.

Problem 2

Consider the situation where you hold a flame under a metal bar as described in class (and shown in the figure below). In this problem you will write a program to solve the steady-state 1-dimensional heat equation to get the temperature T at each point along the bar.



The equation to solve is

$$-\alpha \frac{\partial^2 T(x)}{\partial x^2} = Q(x)$$

Where α is the thermal diffusivity of the metal (a parameter depending upon the type of metal involved), $Q(x)$ is the heat source as a function of position, and $T(x)$ is the temperature. Take $Q(x)$ to be a delta function about 23 cm from the left of the bar.

Upon discretization, the heat equation becomes the the following sparse, linear system:

$$\left(\frac{1}{h^2}\right)\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\ 1 & -2 & 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 1 & -2 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & -2 & 1 & \cdots & 0 & 0 \\ 0 & 0 & 0 & 1 & -2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \cdots & -2 & 1 \\ 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 1 \end{bmatrix} T = -\frac{1}{\alpha h}\begin{bmatrix} Q_1 \\ Q_2 \\ Q_3 \\ Q_4 \\ \vdots \end{bmatrix} \qquad (1)$$

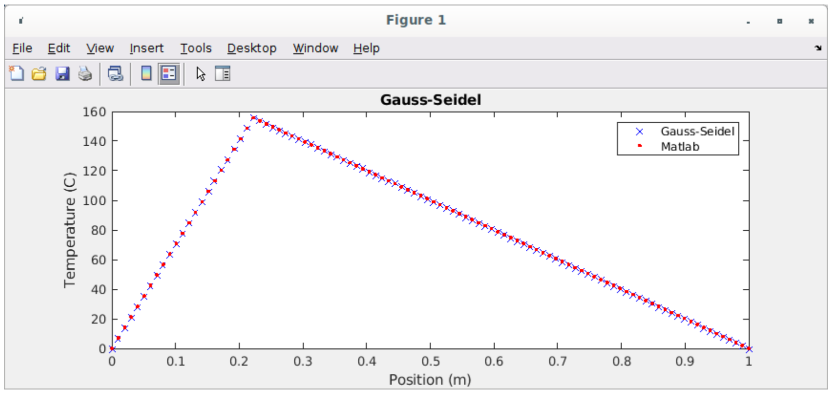
Note that the values in the upper left and lower right are different from the rest of the matrix. This implements the boundary conditions $T_1 = T_N = 0$ – something you don't need to worry about to solve this problem.

Your task: write a program to solve (1) system using Gauss-Seidel iteration. You may use the implementation and test wrapper for Gauss-Seidel as a starting point. Use the following information as you write your program:

- Number of nodes in the simulated metal bar: $N = 100$. This is the number of elements in your Q vector.
- Assume the metal bar is 1.0m long. This information and the number of points N to use will determine the finite difference distance h.
- Boundary conditions: The ends of the bar are fixed at the temperature 0 C using ice cubes. That means the boundary conditions are $T_1 = T_N = 0$. You don't need to do anything special to achieve this – just solve using the above matrix.
- Assume the bar is made of copper. This material has thermal diffusivity α = 1.11e-4 m^2/sec.

- Assume all heat is applied to the one element at node 23. (This is about 23cm from the right end of the bar.) Assume the heat applied is Q = 0.1 C/sec. All other Q nodes are zero.

Your program should produce a plot of temperature vs. position on rod, similar to the one shown below. Also, please report the maximum temperature along the rod.



How to check your work? Using the above inputs, I found that the computed maximum temperature of the bar is around 156 C. I performed an independent analytical calculation which confirmed this result. Also, a good check is to run the program for varying numbers of nodes. If you have implemented the simulation correctly, then the computed temperature should be roughly independent of the node count N. (Variations of a few degrees are OK since discretizing a continuous system represents an approximation.)

Now what is the max temperature if you replace copper with AISI 1010 carbon steel, which has thermal diffusivity α = 1.88e-5 m^2/sec? This grade of steel is commonly used to manufacture automobile bodies.

Problem 3

We again consider solving the linear system

$$Ax = b \tag{2}$$

using an iterative method such as gradient descent or conjugate gradient. We take *A* to be a sparse, SPD matrix. To accelerate convergence, this problem considers the so-called "incomplete Cholesky" preconditioner. This preconditioner relies on something like a Cholesky decomposition of A,

$$A \rightarrow L\, L^T \tag{3}$$

but instead of performing a complete decomposition, the method stores values into *L* ***only*** at elements where *A* itself is non-zero, thus preserving the same sparsity pattern as *A*. Matlab provides the function, "ichol()" which performs this operation.

We'll call the output of ichol() $\hat{L}$. Since $\hat{L}$ is not the exact Cholesky decomposition, the product $\hat{L}\hat{L}^T$ only approximates *A*, i.e. $A \approx \hat{L}\,\hat{L}^T$. Therefore, this approximation holds:

$$\hat{L}^{-1} A \left(\hat{L}^T\right)^{-1} \approx I \tag{4}$$

That is, $\hat{L}^{-1} A \left(\hat{L}^T\right)^{-1}$ is "close to" the identity matrix – and should accordingly have better conditioning than *A*. Moreover, since $\hat{L}$ is triangular, it is easy to compute $\hat{L}^{-1}$ via back-substitution, so using the preconditioner is computationally cheap.

With this background, the system solved by incomplete Cholesky preconditioning is

$$\left(\hat{L}^{-1} A \left(\hat{L}^T\right)^{-1}\right) y = \hat{L}^{-1} b \tag{5a}$$

and the desired solution vector is

$$x = \left(\hat{L}^T\right)^{-1} y \tag{5b}$$

The idea is that an iterative method applied to the preconditioned system (5a) will converge much faster than the same method applied to the raw system (2), then the desired solution (5b) can be obtained using $O(N^2)$ backsubstitution.

Please do the following:

- Please derive (4) and (5) starting with the definition of the incomplete Cholesky decomposition, $A \rightarrow \hat{L}\hat{L}^T$. This is an easy exercise in manipulation of matrices. Please write down and hand in your derivation.
- Please take my implementation of gradient descent (steepest descent) mysd() from canvas and modify it to return the number of iterations required to converge.
- Please download the program K2D from Canvas. This is a simple matrix creator which creates a Laplacian operator for a problem defined on a two-dimensional grid. It accepts two inputs – the X and Y dimensions of the grid. You don't need to know about the Laplacian – just accept K2D as a creator which returns a symmetric, positive-definite matrix which often pops up in applications.
- Write a test program for mysd() which creates *A* matrices of varying sizes using K2D, creates random *b* vectors, sends them to mysd() for solution, then reports the number of iterations required to solve the *Ax* = *b* system. Choose a reasonable value for the stopping tolerance – I used 1e-13.
- Then modify your test program to call mycg first with *A* directly and then with the preconditioned version $\left(\hat{L}^{-1} A \left(\hat{L}^T\right)^{-1}\right)$ (and also the modified *b*). Please use Matlab's ichol() to perform the decomposition, and Matlab's backslash "\" when computing with $\hat{L}^{-1}$. Your test should repeat the comparison for matrices of different sizes and plots the number of iterations to convergence for both. My plot is shown at right. The goal is to show the effect of preconditioning on the convergence of gradient descent.

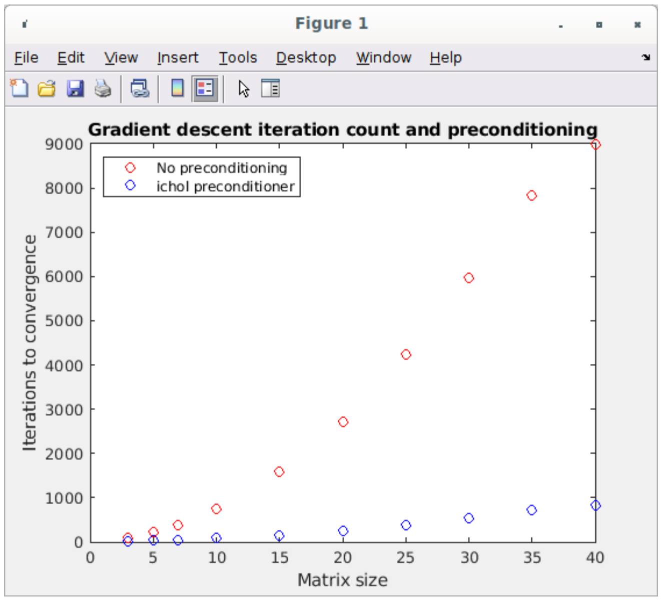


Figure 1
File
Edit
View
Insert
Tools
Desktop
Window
Help
Gradient descent iteration count and preconditioning
No preconditioning
ichol preconditioner
Iterations to convergence
Matrix size