Numerical Analysis 1 – Class 5

Thursday, February 18th, 2021

Subjects covered

- Jacobi iteration and its convergence criteria.
- Gauss-Seidel iteration.
- Solving Ax = b as a minimization problem for SPD matrices.
- Method of steepest descent (gradient descent).
- Conjugate gradient method and its offspring.
- Preconditioning.

Readings

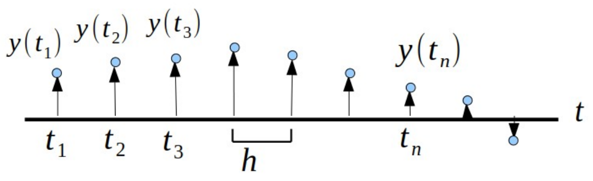
- Kutz, Chapter 2.
- "Notes on Conjugate Gradient", by S. Brorson (on Canvas).
- "An Introduction to Conjugate Gradient Without the Agonizing Pain", by J. R. Shewchuk. (on Canvas)

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. Please use Canvas to upload your submissions under the "Assignments" link for this problem set.

Problem 1

The goal of this problem is to derive matrices useful for taking first and second derivatives of a sampled function $y_n = f(t_n)$. We used a second derivative matrix in lecture. You can consider the t_n to be, for example, the points in time where the function $f(t)$ is sampled, and y_n to be the sampled values of $f(t)$. We take the sample period to be constant, $h = t_{n+1} - t_n$.



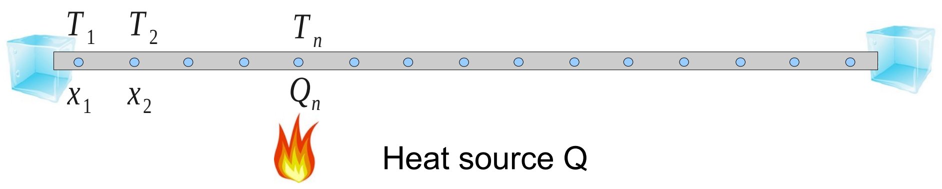
This is a pencil-and-paper exercise (except number 9). Please do the following:

1. Write down the Taylor's series expansion for the next point $y_{n+1} = f(t_{n+1})$ if you know the value of $f(t)$ at $t = t_n$ as well as all derivatives of $f(t)$ at that point.
2. Manipulate the Taylor's series to get an approximation for the derivative which is good to first order. The expression you should get is $dy/dt \approx (y_{n+1} - y_n)/h$. This is known as the "forward difference" approximation to the derivative.
3. Now write down the Taylor's series expansion for previous point $y_{n-1} = f(t_{n-1})$ if you know the value of $f(t)$ at $t = t_n$ as well as all derivatives of $f(t)$ at that point.

4. Then manipulate this expansion to obtain an approximation to the derivative good to first order. This is known as the “backward difference” approximation to the derivative.
5. Now combine the Taylor series expansions in 1 and 3, find an expression which approximates the second derivative of $f(t)$ at point $t=t_n$.
6. Now consider the forward difference first derivative as a matrix which operates on the vector $[y_1, y_2, y_3, \cdots y_{n-1}, y_n, y_{n+1} \cdots]^T$ to return an approximation to the first derivative, $[y'_1, y'_2, y'_3, \cdots y'_{n-1}, y'_n, y'_{n+1} \cdots]^T$. (That is, $\vec{y}' = D_1 \vec{y}$ where D_1^+ is the first derivative operator.) From step 2, please write down the matrix.
7. For fun, write a quick program which creates a vector $[1, 2, 3, \cdots 7, 8, 9, 10]^T$ and takes its derivative using D_1^+ . Regarding a test, please make a plot and compare the computed derivative against what you think should be the derivative. You can ignore any discrepancies at the end points – just think about the elements in the middle of the vector.
8. Please write down the matrix corresponding to the backwards difference, D_1^-
9. Please write down the matrix corresponding to the second derivative, D_2 .
10. For fun, write a quick program which creates D_1^+ and D_1^- and then computes the product $D_1^+ D_1^-$. What is the resulting matrix? Note: for the purposes of answering this question it’s fine to ignore the first or last element on the matrix diagonal – just think about the elements in the middle of the matrix.

Problem 2

Consider the situation where you hold a flame under a metal bar as described in class (and shown in the figure below). 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 30 cm from the left of the bar.

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

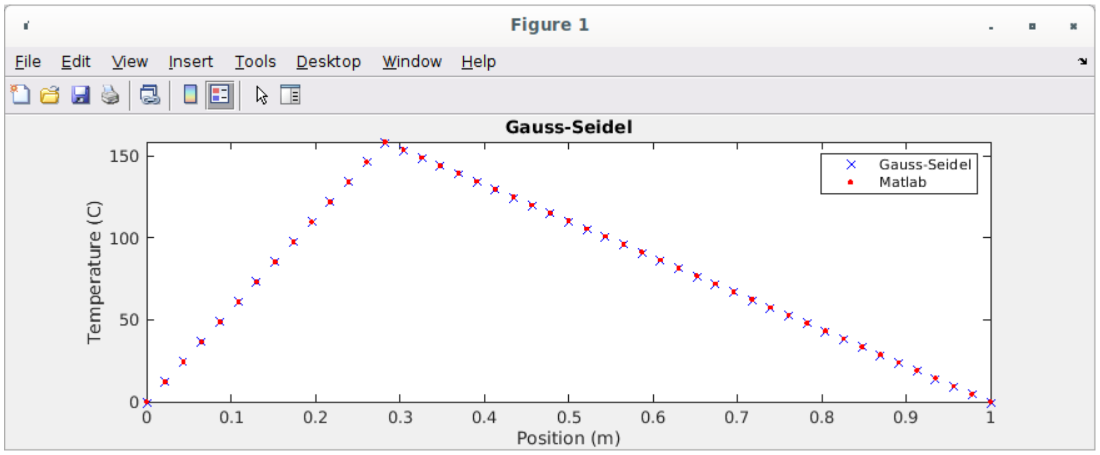
$$\left(\frac{1}{h^2}\right)\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\ 1 & -2 & 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 1 & -2 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & -2 & 1 & \cdots & 0 & 0 \\ 0 & 0 & 0 & 1 & -2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots &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

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 this system using Gauss-Seidel iteration. You may use the implementation and test wrapper for Jacobi posted on Blackboard as a starting point; modify the Jacobi program to implement Gauss-Seidel iteration. Use the following information as you write your program:

- Number of nodes in the simulated metal bar: N = 47. 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 6061 aluminum. This material has thermal diffusivity α = 6.4e-5 m^2/sec. (6061 is a common industrial alloy of aluminum.)
- Assume all heat is applied to the one element at node 14. (This is about 30cm from the right end of the bar.) Assume the heat applied is Q = 0.05 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 used an independent computation to find that the maximum temperature of the bar is around 160 C. 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 independent of the node count N.

Now what is the max temperature if you replace Alunimum with iron, which has thermal diffusivity 2.3e-5 m^2/sec?

Problem 3

This problem involves writing a program which performs an A-orthogonalization process on an input SPD matrix A. We'll start with standard Gram-Schmidt routine, but modify it so the returned matrix has columns which are all A-orthogonal (conjugate) to each other. This is one of the things the conjugate gradient algorithm does (under the covers) when solving the linear system Ax = b.

The algorithm proceeds as follows:

1. Take as input an NxN matrix A. We take the columns of A to be a set of column vectors u_k :

$$A = \begin{pmatrix} \vdots & \vdots & \vdots & & \vdots \\ u_1 & u_2 & u_3 & \cdots & u_N \\ \vdots & \vdots & \vdots & & \vdots \end{pmatrix}$$

   The goal is to use the space spanned by the columns of A to create a set of vectors d_k which also span the space of A, and additionally satisfy $d_j^T A d_i = 0$ for $i \neq j$, and $d_j^T A d_i = C_i$ for $i = j$, where C_i is some constant.

2. Find the first vector $d_1 = u_1$.
3. Find the second vector $d_2 = u_2 - \beta_{21} d_1$ where $\beta_{21} = \frac{u_2^T A d_1}{d_1^T A d_1}$. Note that this removes the part of u_2 which is pointing in the same conjugate direction as d_1 , leaving d_2 A-orthogonal to d_1 . Please prove this is the case for d_2 using pencil and paper, and hand in your proof along with your program.
4. Find the remaining d_k vectors using the iteration

$$d_k = u_k - \sum_{j=1}^{k-1} \beta_{kj} d_j$$

5. Assemble the result matrix and return it.

$$D = \begin{pmatrix} \vdots & \vdots & \vdots & & \vdots \\ d_1 & d_2 & d_3 & \cdots & d_N \\ \vdots & \vdots & \vdots & & \vdots \end{pmatrix}$$

Feel free to use my implementation of Gram-Schmidt (on Canvas, Class 4 under QR decomposition) as a starting point for your program. Please make sure you include a test function which verifies that all columns in your D matrix are A-orthogonal to each other. Keep in mind that CG works for symmetric positive-definite (SPD) matrices, so you only need to test your algo against SPD inputs. Also note that testing is tricky because the Gram-Schmidt process is not very stable numerically. Therefore, it can accumulate large round-off errors. Feel free to use a generous testing tolerance to verify your results.