ME 388F Homework 1

Erik Hansen

January 22, 2025

1 Discussion of Matrix Solution

- The biggest factor influencing my work due to my computer's limitations was the memory constraint. I had anticipated that I would not be able to use a matrix inverse directly to solve these problems for large $N \sim 10^9$ because that would require $\sim 2^9$GB. Based on what I know about the size of a double precision number it made sense that I was unable to use a full matrix at $N = 10^5$ or use the current form of the banded solver in LAPACK at 10^9, which would require 10^{10} or more elements. But with that in mind for the other examples, I was actually surprised that my computer was able to solve the tri-diagonal problems at our largest N in around thirty seconds given that more than my 18 GB of memory would have been in use.

- Because the size of a full matrix was too large for my computer, I exploited the fact that the super and sub-diagonals were constant for the triangular matrices to use a recursive formula to solve the triangular matrix problems.[7] This allowed solutions at $N = 10^5$ and $N = 10^6$ for a triangular matrix where the full matrix would have been intractable due to memory constraints.

  However, it surprised me that even though in the first problem the matrix was the identity and the solution was obvious, the variable assignment took a non-neglible portion of the time to compute the solution with the tri-diagonal method.

- Since I anticipated that we would have memory constraints, I used LAPACK to handle the diagonal, tri-diagonal, and banded matrix problems. [1] This made the tri-diagonal problems much easier than the triangular problems, for example, although I still had to think carefully about indexing for the banded matrix.

  I also made a guess based on some examples in Mathematica that the full matrix was positive definite, in addition to symmetric, so that I could use alternative LAPACK routines which saved some compute time but without changing the intrinsic scaling of the problems. This may have also been possible for the banded matrix, but I wasn't able to use the banded Cholesky routine.

While reading more about the types of matrices we encountered, I did learn about the Levinson $O(N^2)$ algorithm or even superfast $O(N\log(N)^2)$ algorithms that I could have written to solve Toeplitz matrices, which may have been able to improve intrinsic scaling and dodge memory constraints for these matrices.[6] [2] However, this wouldn't reduce the time complexity of the triangular matrices or alleviate the memory burden of the full matrix because not all diagonal elements were the same in those cases.

- I had originally intended to use MATLAB for some of these exercises, but immediately ran into the abovementioned issue where a $10^5 \times 10^5$ matrix was too big. However, I gravitated towards Fortran and LAPACK out of familiarity with Fortran in my numerical simulations (although I haven't used LAPACK before). With that being said, the structure of LAPACK's tri-diagonal function greatly simplified the process of solving that problem. I did struggle for a few hours with how to link the LAPACK files and how to use the banded storage to solve the banded matrix equation, so there may be some dispute of a balance between development time and runtime. (See [5][1][3] for information I used on understanding of the leading dimension). For sake of performing a quick review, SciPy has a similar banded matrix solver in its linear algebra library. SciPy also has the Levinson $O(N^2)$ Toeplitz matrix solver I alluded to earlier, so if I were to use that platform I could readily test if that approach is useful for the other cases.

Note: my code also uses information from Stack Overflow [9] to link the LAPACK code on Mac devices and [4] to read an input of the power of 10 to consider for size.

2 View Factors and Transport Connection

(1) As we discussed in lecture on January 14, half of the radiation from the hot plate moves towards the cold plate and half goes away, so $\frac{1}{2}$ of the heat from the hot plate can see the cold plate.

(2) This argument does not change for the 1cm^2 hot plate, as any radiation emitted in the direction of an infinite cold plate will reach it. So $\frac{1}{2}$ of the heat from the hot plate can see the cold plate in this scenario as well.

(3) Given that radiation is emitted in all directions from the hot wire, as a wire thickness decreases, the cold wire occupies less solid angle from the perspective of an element of the hot wire. Consequently, none of the heat from the hot wire can see the cold wire.

(4) To obtain the view factor in general, we know that the view factor from a point source on the hot plate should be the fraction of the solid angle that the cold plate subtends. This is heavily inspired by the description of the Nusselt analog picture of view factors on the view factor Wikipedia page, but to find this factor for a point source, we would project the cold plane onto a unit sphere about the point source and find the ratio of the area of the projected plane to the surface area of the sphere, 4π [8]. At this point, to find the total view factor of the hot plate, we would average the point view factors over the area of the hot plate.

(5) Intuitively, as the hot plate approaches the cold plate, the cold plate subtends a larger solid angle from the perspective of each piece of the hot plate, so the fraction of heat seeing the cold plate increases as the hot plate grows closer and decreases as the hot plate grows farther away. This is consistent with the formula given in Table 1 of the Wikipedia page on solid angle [8]

$$F_{ij} = \frac{\sqrt{(w_i/L + w_j/L)^2 + 4} - \sqrt{(w_i/L - w_j/L)^2 + 4}}{2w_i/L} = L\left(\sqrt{1 + 1/L^2} - 1\right) \approx \frac{1}{2L} \quad (1)$$

for large distances between the parallel plates L and $w_i = w_j = 1$cm the width of the plates.

There is a beam of neutrons with an incident angular flux of one that is parallel with the x-axis and perpendicular to the left-surface of a purely-absorbing material with a macroscopic cross section of $\Sigma = 1$ reactions per cm of neutron travel. The transport equation for this simplified problem is $\frac{d\psi}{dx} + \Sigma_t \psi = 0$ where $\psi(x = 0) = 1$.

(1) Given the transport equation and the cross section $\Sigma = 1cm^{-1}$, we find that

$$\frac{d\psi}{dx} = -\psi \quad (2)$$

which we can integrate via separation of variables to obtain

$$\ln(\psi(x)) - \ln(\psi(0)) = \ln(\psi(x)) = -x \quad (3)$$

or $\psi(x) = \exp(-x)$. This solution is plotted in Excel in Figures 1 and 2 on the next page.

(2) As described explicitly below, we have solved this problem in the upwind matrix above and show the result on the same graph as above. This shows that while we see the same pattern of an exponential decrease, the solution from the matrix problem is an underestimate of the solution, which we can interpret as a discretization error due to the size of Δx.

(3) If we discretize the x-axis with nodes $x_i = i(\Delta x)$, take $\psi_n = \psi(x_n)$ and rewrite the derivative $\frac{d\psi}{dx} = \frac{\psi_{n+1} - \psi_n}{\Delta x}$, we obtain

$$\psi_{n+1} - \psi_n + (\Sigma_t \Delta x)\psi_n = \psi_{n+1} - (1 - \Sigma_t \Delta_x)\psi_n = 0 \quad (4)$$

Substituting $\Sigma_t = 1$ and $\Delta x = 0.1$,

$$-0.9\psi_n + \psi_{n+1} = 0 \quad (5)$$

With the condition $\psi_0 = 1$, we see that the equation for many $\psi_0, \psi_1, \psi_2, \psi_3, \ldots$ becomes a linear system

$$\begin{bmatrix} 1 & 0 & 0 & 0 & \ldots \\ -0.9 & 1 & 0 & 0 & \ldots \\ 0 & -0.9 & 1 & 0 & \ldots \\ 0 & 0 & -0.9 & 1 & \ldots \\ \ldots & & & & \end{bmatrix} \begin{bmatrix} \psi_0 \\ \psi_1 \\ \psi_2 \\ \psi_3 \\ \ldots \end{bmatrix} = \begin{bmatrix} 1.0 \\ 0 \\ 0 \\ 0 \\ \ldots \end{bmatrix} \quad (6)$$

We identify this matrix as the upwind matrix we encountered earlier, and aside from a scaling of our initial condition, this is Problem 7 we solved earlier.

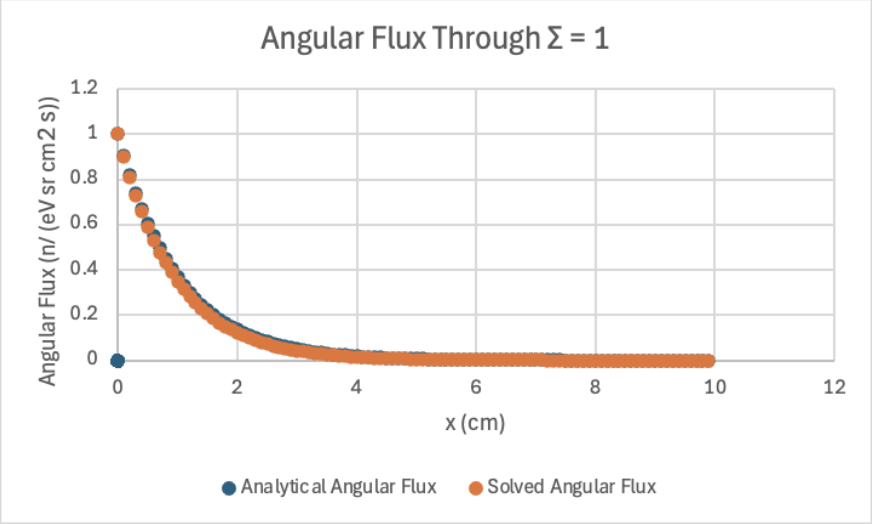

Figure 1: Angular flux due to the neutron beam.

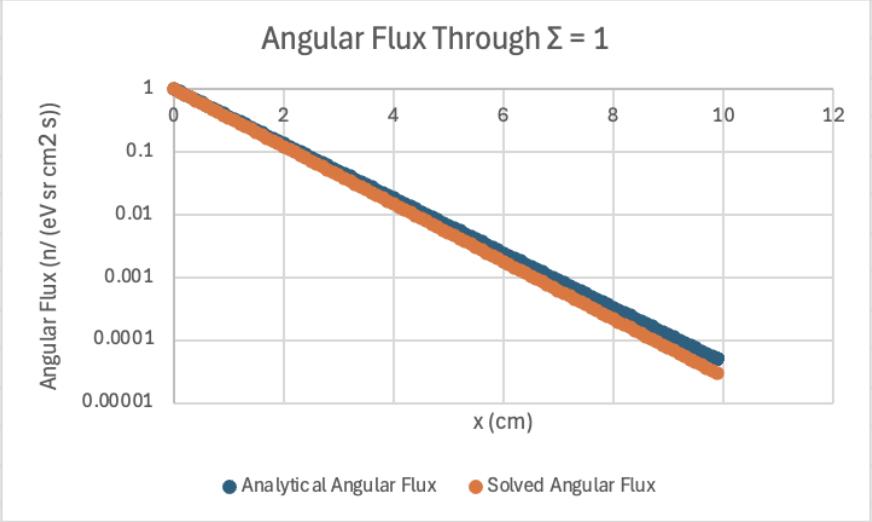

Figure 2: Angular flux due to the neutron beam as shown on a log scale for closer comparison.

References

[1] Lapack documentation. Technical report, Netlib, 2025.

[2] R. P. Brent, F. G. Gustavson, and D. Y. Yun. Fast solution of toeplitz systems of equations and computation of padé approximants. *Journal of Algorithms*, 1980.

[3] IBM. How leading dimension is used for matrices. Documentation, 2021.

[4] M. Thorne. Sactools. Github Repository.

[5] P. Warden. An engineer's guide to gemm. Blog; used for leading dimension description.

[6] Wikipedia. Levinson recursion.

[7] Wikipedia. Triangular matrix.

[8] Wikipedia. View factor. Most information from Incropera, Frank P. et al. (2013). Principles of heat and mass transfer (7. ed., international student version ed.). Hoboken, NJ: Wiley., 2024.

[9] . How do i link the accelerate framework to a c program in macos? StackOverflow.