Computational Physics: Linear Algebra

Ujwal Kumar

March 24, 2021

1 Introduction

In this report, we study methods to computing resistances from a number of resistors in a grid as seen in Figure 1.

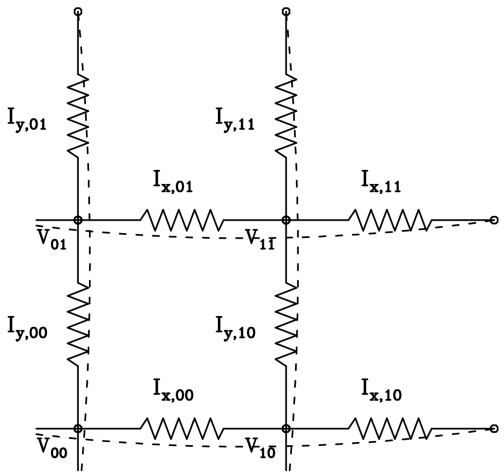

Figure 1: Resistors in a grid with currents flowing through the wires

With a finite number of resistors, it is easy to calculate the resistance using either an Ohmmeter or simple resistor in a circuit math. In order to do this, we can convert the above problem into a matrix problem and by using Kirchoff's loop and junction rules. The loop rule states that the sum of voltages in an arbitrary selected loop must be 0, where the voltage, V equals the current, I, times the resistance, R, i.e., $V = I \times R$. The junction rule state that the current going into a junction (or vertex in the grid) is the current coming out of the junction.

This allows us to rewrite this grid as a number of equations that can then be converted into a matrix. In our problem, we set each resistance equal to 1 Ω and our goal is to compute the net resistance of the grid with our Ohmmeter placed between two points on the grid (x_1, y_1) and (x_2, y_2).

Computationally, we create a routine that takes in the number of vertices, the number of resistors. The circuit is specified by the array of struct resistors where each struct resistor has the input vertices and R, the resistance. Additionally, we choose to use an enumeration method to represent the vertices such that on our NxN grid, we have nr = $2N^2$ and nv = N^2 with each site of vertex indicated by iN+j. The currents flowing through the resistors are I. We then use a Gaussian elimination and back substitution method to solve the matrix.

2 Problems

2.1 Question 1

For the 2x2 grid as seen in Figure 1, we aim to compute the resistances between (0,0) and (1,1). This result can be computed analytically and then compared to our implementation in code.

From the code, we find a resistance of 0.5 Ω. Analytically, we can simplify this circuit into one circuit with two pairs of parallel resistors in series with each other. The parallel resistors, each with 1 Ω combined with two sets of these in series net a total of 0.5 Ω in total resistance.

2.2 Question 2

Next, we look at a 5x5 grid between (1,1) and (3,2) as well as between (0,0) and (2,1) and examine the differences between the results.

We find the resistance between $(1, 1)$ and $(3, 2)$ on a 5×5 grid = 0.68 Ω. We find the same resistance between $(0, 0)$ and $(2, 1)$ and this can be attributed to the idea that the resistance is simply dependent on the "real" or "actual" separation between the two points.

2.3 Question 3

Next, we look at the resistance for larger grids from sizes of N = 10 to N = 40 between the points (0,0) and (1,2).

We find the 10×10 grid to yield a net resistance of 0.748564 Ω, a 20×20 yields a net resistance of 0.767009 Ω, a 30×30 yields a net resistance of 0.770466 Ω, and a 40×40 yields a net resistance of 0.771678 Ω.

We can plot this and fit a polynomial of the form $a + \frac{b}{N^2}$ and with a polynomial of the form $a + \frac{b}{N^2} + \frac{c}{N^4}$.

Our polynomial of second order yields a fit with a = 0.773199 and b = 2.46406. Our polynomial of the fourth order yields a fit with a = 0.77324, b = -2.50052, c = 3.29325. We can fit these plots to the data to see the results in Figures 2 and 3

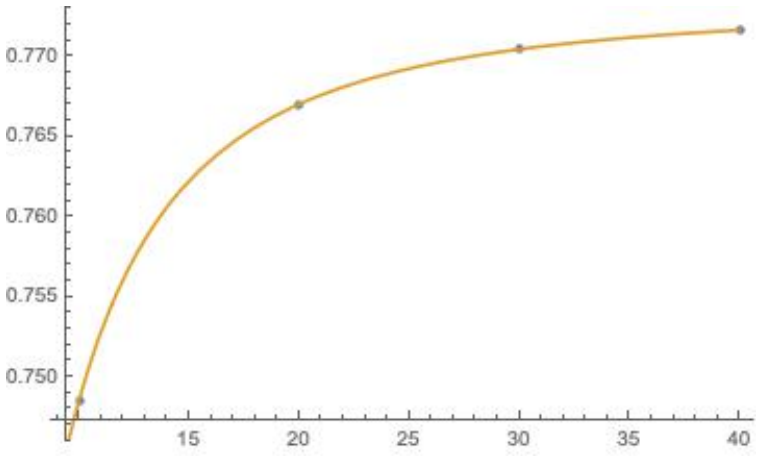

Figure 2: Fit of second order polynomial to data plotting net resistance in an NxN grid of resistors against N

The fits are very close visually, and it can be argued that Figure 3 yields a slightly tighter fit. The parameter values are also incredibly close. Therefore, the second order fit may be appropriate for this exercise since we use only two terms and achieve nearly the same precision.

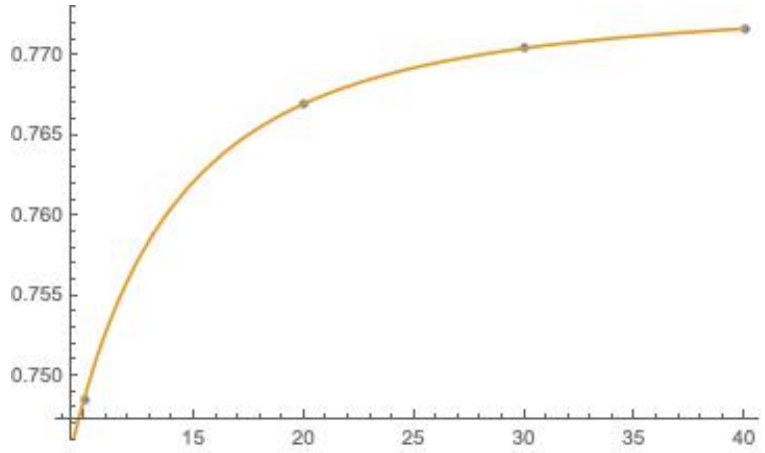

Figure 3: Fit of fourth order polynomial to data plotting net resistance in an NxN grid of resistors against N

2.4 Question 4

Finally, we study the computational time taken to compute each of these results. We find the 10×10 grid to yield a time of 0.057 s, the 20×20 to yield a time of 1.744 s, the 30×30 to yield a time of 17.189 s, and lastly, the 40×40 to yield a time of 81.416 s. The reported times are CPU times and not real times.

We plot these results in log-log space, as seen in Figure 4 and study the scaling law for computational time.

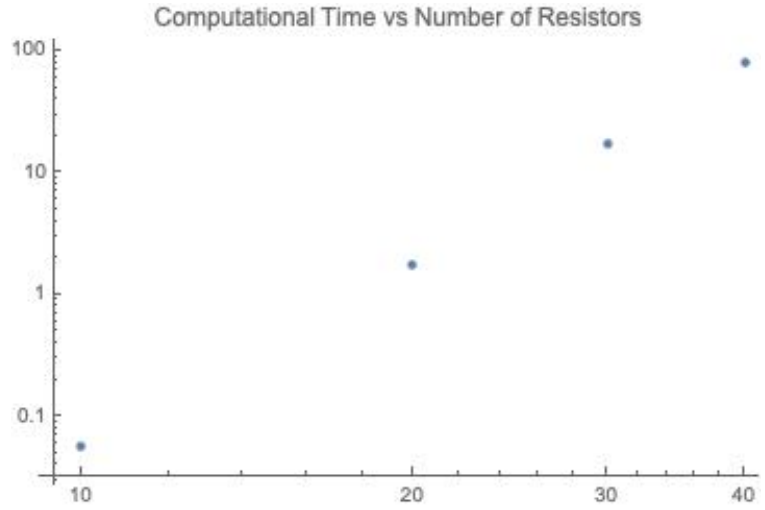

Figure 4: Computational Time (in seconds) plotted against the number of resistors

The slope of this logarithmic curve, i.e., the index of the power law, derived using the coordinates of the minimum and maximum points is between the minimum and maximum points is 2.711 s. We thus find that the a 100×100 grid should have a computational time of 271.16 s, using this log-log extrapolation.

3 Conclusion

This report exposed me to using linear algebra in computational contexts and allowed me to see the power of it when the number of variables scales largely which is much more appropriate in real world contexts. I hope to build on this skill to see how linear algebra can be used to compute solutions in other contexts also.

The initial problem regarding the analytic solution was confusing personally but when I was explained to that the resistors on the outside of each segment can be treated as a single resistor looping around, that problem became much simpler. I found the Fit function on Mathematica to be a bit non-intuitive and would prefer using Python to fitting or modelling data, but this is simply a case of practice/experience.