Gauss-Seidel Solver (with Relaxation)

Owen Moogk (21006256)

# Equations to be Solved

## Circuit 1

A math equations on a piece of paper

Description automatically generated

A close-up of a number

Description automatically generated

## Circuit 2

A math equations on a piece of paper

Description automatically generated

A close-up of numbers

Description automatically generated

# Gauss Seidel Solution

## Solution Demonstration

### Circuit 1

A screenshot of a computer code

Description automatically generated

### Circuit 3

A screenshot of a computer

Description automatically generated

## Gauss Seidel Discussion

1. Partial Pivoting: Partial pivoting was required to solve these systems of equations, as originally they were not in a diagonally dominant form. However, it was simpler to do this by hand than to do it in Matlab, so this was done manually before inputting them into the Matlab worksheets.
2. The initial values I selected were the corresponding zero vectors for each system size. Because each system is diagonally dominant, the initial value doesn’t really matter, as convergence is guaranteed.
3. Note: The following findings were found using Circuit 1. A relaxation parameter of 2 cause divergence, however a relaxation parameter of 1.99 converged. However, this took very long, specifically 988 iterations (seen below). This is actually much larger than no relaxation (parameter = 1), which took 69 iterations[[1]](#footnote-1). The parameter that seemed optimal was 1.6, as it only took 22 iterations to converge with the same error. And of course, with a lower relaxation parameter, it took longer to converge. Using 0.8, it took 98 iterations to converge.  
     
   The proof of all of this is pictured below, with the found solutions, compared to the real solution, and number of iterations.

Note that these matrices actually solve for voltages at all of the nodes. This is much simpler than solving for current, and required less circuit manipulation beforehand. From the solutions, we can then solve for currents through resistors.

The solution is as follows:

## Circuit 1

A close-up of a math problem

Description automatically generated

## Circuit 3

Note the node numbering chosen in the image.

A screenshot of a cell phone

Description automatically generated

A screenshot of a computer

Description automatically generated

Figure 1: Relaxation = 1.99

A screenshot of a computer

Description automatically generated

Figure 2: Relaxation = 1.6

A screenshot of a computer

Description automatically generated

Figure 3: Relaxation = 1

A screenshot of a computer

Description automatically generated

Figure 4: Relaxation = 0.8

## Matlab Equation Solver Discussion

The solution from the Matlab equation solver was very similar to the solutions found with Gauss Seidel, and were only usually off by less than 0.1. The Matlab `linsolve` command uses direct methods to solve the system (Source: <https://www.mathworks.com/help/matlab/math/iterative-methods-for-linear-systems.html>), meaning that it is more accurate than the solution found using Gauss Seidel. This is because it is exact (approximately, if you ignore floating point errors), so it will always be better than an iterative method. The downside of this is that it usually takes longer, so for very large matrices with a lot of zeroes, also known as “Large and sparse”, an iterative method will have far superior performance.

# Appendix

A screenshot of a computer program

Description automatically generated

Figure 5: Circuit 1 Code

A screenshot of a computer

Description automatically generated

Figure 6: Circuit 3 Code

A screenshot of a computer code

Description automatically generated

Figure 7: Gauss Seidel Code

1. Nice. [↑](#footnote-ref-1)