**Simulation Program for Integrate Circuits (SPICE)**

By Matthew Vater, Adam Mannheim, and Nathan Spangler

1. **Introduction**

For the final assignment we were able to work in teams in order to simulate integrated circuits. The goal of this simulation was to be able to upload an array that represents an integrated circuit and have the simulation program perform circuit analysis. The analysis would solve for current (Amps) at all the nodes of the circuit. The circuit analysis is able to handle different circuit components including resistors, capacitors, and diodes. This assignment reinforces an understanding of circuit analysis and knowing how different circuit components interact with one another. Whether it’s calculating voltage drop over a resistor or how diodes effect current. A variety of different mathematical methods were used, including linear algebra matrix multiplication and diode linear companion model.

1. **Methods**

To start off, we knew we needed a way to do a lot of linear algebra matrix multiplication. This is because of the [A] x [X] = [Z] expression, where we are given A and Z and we need to solve for X. So we decided to use the Armadillo C++ Linear Algebra Library. This gives us the advantage of entering the two known matrices and we receive the unknown ‘X’ matrix.

After figuring that out we set off to start with the basics, the basics being resistors. We also decided to split resistors, capacitors, and diodes into their own separate .cpp files and header files to keep the code clean and simplistic. This is where you can find each components respective functions that are needed to perform calculations. Some functions being get\_current() and get\_conductance(). So getting back to the resistors. No real calculations are done with the resistor values. They are given in a matrix and we simply keep the value in the matrix until it is time to solve.

So we then moved on to implementing both the capacitor and diode files. This was very simply. Each file contains two to three functions that perform simple math equations that will be used by the source file prior to solving for the final answer.

So now that we have gotten that out of the way we moved onto the source file which contains the process of how we created the necessary matrices to analyze the circuit correctly. In the beginning of main you will find that this is where the matrix that represents the circuit is constructed/loaded in. This is where we learn how many resistors, capacitors, diodes, voltage sources, and current sources are used alone with what nodes they are attached to.

In order to solve the equation [A] x [X] = [Z], we must split the [A] matrix into four sub matrices including G, B, C, and D. We start of by populating the G submatrix, which contains represents the sum of conductance at each node in the circuit. We then form B which shows all the power sources. Which is then followed by the submatrix C which is simply the transpose of submatrix B. Submatrix D is formed, which represents dependent sources. Finally all four of these submatrices are joined with one simple call to the armadillo join\_cols() and join\_rows() functions. We now have matrix [A] created and populated.

We then create [X] which is populated with zeros, that was easy. Now we move on to trying to populate the [Z] matrix. We first go and insert all the voltage values to each node where a power source exists. We then do the same with current sources by inserting the current (amps) value at each node where a current source exists. We then move on to iterating over all the capacitors going from node to node. If we find a capacitor we call the capacitor’s get\_current() function and adjust the current value at that node. Once we iterate over all the nodes and there are no more capacitors we move on to diodes. Where we again iterate over all the nodes and if we find a diode we call the diode’s get\_current() function and adjust the current value at that node.

Once we have done all of that fun stuff. The time has come, the moment we’ve all been waiting for. We can now call armadillo’s solve function. The function then populates the [X] matrix with all the fun and wonderful values we are looking for.

1. **Results**

*Image 1 shows something about circuit analysis*

*Image 2 shows something else about circuit analysis*

*Image 3 shows something else about circuit analysis*

1. **Discussion**

As scientists, from this simulation we were able to refresh our skills and knowledge of circuit analysis, but did not learn anything knew about circuit analysis. We did however learn that writing a program to analyze a circuit and calculate values is very complex with a large range of scenarios to consider. It was nice to be able to apply different concepts we learned in different classes, such as Linear Algebra and Circuit Analysis, in order to create this simulation. As students, we were able to get real world experience with coding a project as a team. For some of us (Nate), it is nice to be able to work with other students who are better programs, which allows you to learn how to become better yourself. Plus this project was very neat in the aspect that I(Nate) used SPICE before in the real world, and when using it I never thought about being able to code it myself.

1. **References**

Standard C++ Library

Armadillo: C++ Linear Algebra Library

<http://kona.ee.pitt.edu/1180wiki/doku.php?id=simulation_program_for_integrated_circuits_spice>