Sophia Ardell

Homework #3

Wednesday, September 23, 2015

**Problem 1:**

My code first requires the maximum number of equations for the question, as that will be the limit for the dimensions of all matrices and vectors. It then asks the user which part they are choosing to work on, as the code is broken down from there. In all cases, the code initializes the matrix a, and the vectors b, x, resistances, and voltage. The length of these is dependent on the given situation.

The resistance vector holds all resistances, with the index corresponding to the resistor’s location in the loop - 1 (because C++ counts from 0). The same relationship is true for the voltage vector.

The matrix a, and vectors b and x correspond with the system of equations for the entire circuit being set up in a linear format, such that A holds resistance values, x holds current, and b holds total voltage for a loop (given that Ax = b). If solved for any loop i, the equation would appear as:

*V*1*+ V*i *= R*i*I*i *+ I*1*R*1

(1)

This can be derived by applying Kirchoff’s laws to the circuit (see proof in 1A write up). In each situation, all resistances and voltages are provided, so using LU decomposition, one can solve for any current value.

The code has the functions setA and setB, which return a matrix and a vector of doubles respectively. The function setA creates a square matrix that has the dimensions equivalent to the amount of equations. In the first line, it writes out Kirchoff’s Current Law as applied to the specific circuit, which is:

(2)

The rest of the lines are set so that when the function is solved, each line will match the right hand side of (1). To achieve this, it takes values from the resistances vector.

The function setB returns a vector of doubles that is the same length as the number of equations. It sets the first index to match the right hand side of (2), and the rest to match the left hand side of (1). To achieve this, it takes values from the voltage vector.

*Part C:*

The dimensions of A, b, x, resistances, and voltage is set as the maximum amount of equations, as it will only solve the problem once for one given set of equations. The code was written generically so each time the user could input different voltage and resistance values depending on the problem they want to solve. These values are inputted to the resistance and voltage vectors, which are then used to set a and b, respectively. The code then uses LU decomposition on a, and solves it for x (the current vector) as it relates to b. It outputs the current value for each part of the loop.

*Part D:*

This function is designed to analyze I1 as a function of *n* (number of loops/ independent equations), given Ri = 1Ω and V1 = 1 *V* with all other Vi = 0 *V*. This code writes all values to a .dat file so it can be graphed. The code runs for n times, where n is the maximum amount of iterations established by the user. Each iteration, it increases the amount of loops by 1, starting from 1. It initializes matrix a, and vectors b, x, resistances, and voltage with the dimensions set as the current amount of loops. It sets the voltages and resistances as specified in the given conditions. It then sets a and b based on these conditions, does LU decomposition on a, solves for x, and writes the value of n along with I1 to the file.

Analytically, I predicted that , which does match the data from the file, as seen on the graph below.

[insert part D graph]

*Part E:*

This function follows a similar format as Part D, except all voltages are 1 *V* and all resistances are set so that Ri = iΩ. It is solved using the same method. At *n* = 100,

[insert graph]

*Note: see attached for data for Parts D and E*

**Problem 2:**

This code is specialized to solve the specific problem in 3.2. It first sets dimensions as a constant integer, then initializes matrix a as the following:

It then uses LU decomposition on it. The function is divided into two separate functions for part b and c.

*Part B:*

The function creates another matrix that will hold the inverse of a, which it calculates to be:

It then finds the condition number of each matrix. The condition number of a matrix is found by adding the absolute values of each item by row, then returning the greatest value. The same is done on the inverse of the matrix. These two values are then multiplied together to give the condition number of the matrix.

*Part C:*

The function initializes vector b and x to solve the system of equations (given that Ax = b). It sets b as the following:

It then solves for x, returning the following: