Introduction to Computers and Engineering Problem Solving

Spring 2012

Problem Set 10: Electrical Circuits

Due: 12 noon, Friday May 11, 2012

I. Problem Statement

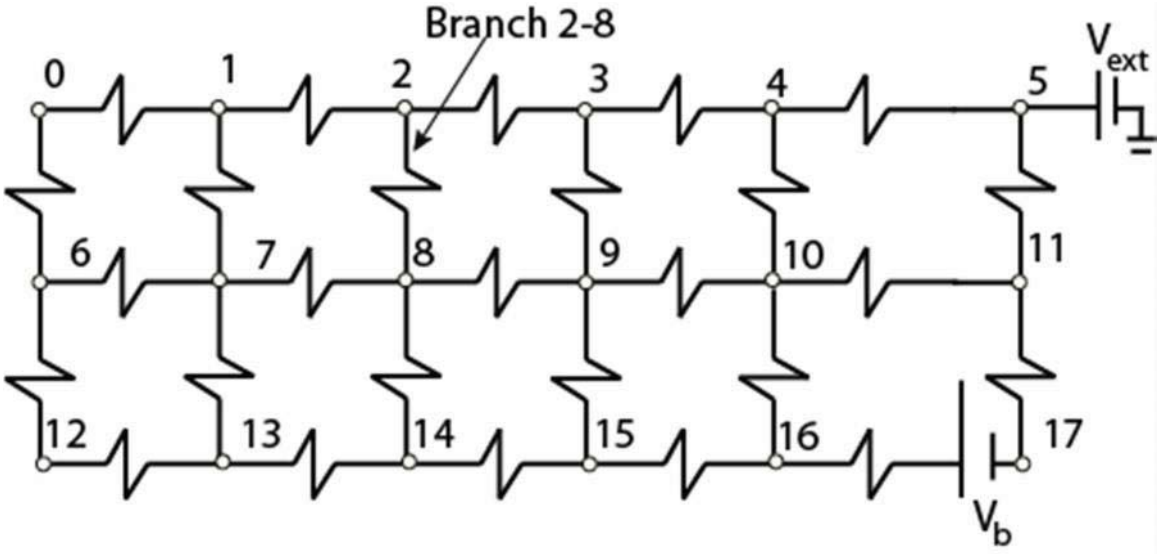


Figure 1. Electric circuit

The electric circuit displayed in Figure 1 can be solved applying Kirchhoff's and Ohm's circuit laws, which deal with the conservation of charge and energy in electrical circuits. There is a battery, near node 17, with V_b=12 volts, and an external voltage source that keeps node 5 at a fixed voltage V_{ext}=6 volts. A program fragment that you will download (PSet10Files.zip) provides the value of each resistor. Your program should compute the voltage at each node and the current through each resistor.

Figure 2 shows a simple network with 4 resistors, which we use to illustrate the solution technique that you will use to solve the network in Figure 1.

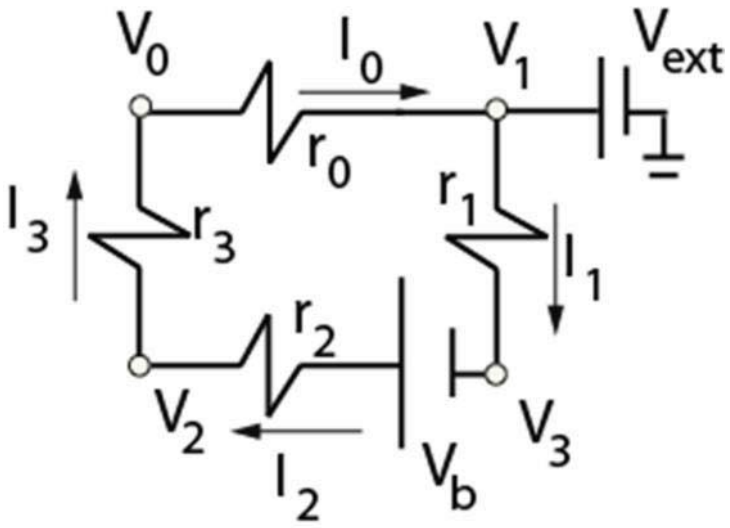


Figure 2. A simple circuit

Ohm's Law: The voltage drop V across a resistor is the product of the current I and the resistance R: V=IR. To find the current I, we use I=V/R at each node in the network:

$$I_0 = (V_0 - V_1)/R_0 \quad \text{(eq. 1)}$$
$$I_1 = (V_1 - V_3)/R_1 \quad \text{(eq. 2)}$$
$$I_2 = (V_3 - V_2 - V_b)/R_2 \quad \text{(eq. 3)}$$
$$I_3 = (V_2 - V_0)/R_3 \quad \text{(eq. 4)}$$

Kirchhoff's Law: The sum of the currents flowing into a node is equal to the sum of the currents flowing out. We apply this at each node in the network:

$$I_0 - I_3 = 0 \quad \text{(eq. 5-node 0)}$$
$$I_1 - I_0 = 0 \quad \text{(eq. 6-node 1)}$$
$$I_3 - I_2 = 0 \quad \text{(eq. 7-node 2)}$$
$$I_1 - I_2 = 0 \quad \text{(eq. 8-node 3)}$$

The matrix representation of this system, in the form A=xb, can be obtained substituting eqs. (1-4) into eqs. (5-8). For simplicity, we write the inverse of the resistances: $r_i=1/R_i$. After putting all terms with v_i on the left hand side, and constant terms in v_b on the right hand side of each equation, we obtain:

$$\begin{bmatrix} r_0+r_3 & -r_0 & -r_3 & 0 \\ -r_0 & r_1+r_0 & 0 & -r_1 \\ -r_3 & 0 & r_2+r_3 & -r_2 \\ 0 & -r_1 & -r_2 & r_1+r_3 \end{bmatrix} \begin{bmatrix} V_0 \\ V_1 \\ V_2 \\ V_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -V_b r_2 \\ V_b r_2 \end{bmatrix} \quad \text{(eq. 9)}$$

$$A \quad x = b$$

In equation 9:

- The voltages are the unknowns (x).
- The resistances are known constants.
- The right hand side is zero, except when a battery or external voltage is at the other end of a resistor. Battery and external voltages are known (constants).
- Current I is not used. You will compute currents later, after you find the voltages.

You should make sure you can write equation 9 systematically. By understanding this small example, you'll be able to work with the instructions given below to construct the A and b matrices for the large, 18 node problem in Figure 1.

Node 1 is a special case, with constant voltage V_{ext} which is known (6 volt). Without going through all the algebra, the matrix system (eq. 9) must be modified to be:

$$\begin{bmatrix} r_0+r_3 & 0 & -r_3 & 0 \\ 0 & 1 & 0 & 0 \\ -r_3 & 0 & r_2+r_3 & -r_2 \\ 0 & 0 & -r_2 & r_1+r_3 \end{bmatrix} \begin{bmatrix} V_0 \\ V_1 \\ V_2 \\ V_3 \end{bmatrix} = \begin{bmatrix} V_{ext} r_0 \\ -V_{ext}(r_0+r_1) \\ -V_b r_2 \\ V_b r_2 + V_{ext} r_1 \end{bmatrix} \quad \text{(eq. 10)}$$

To go from (eq. 9) to (eq. 10), which represents the external voltage at node 1, the matrix must be modified in the following way:

- In the b matrix, set $b_i = b_i - V_{ext} a_{i1}$
- After doing the operation above, set all coefficients = 0 in row 1 and column 1 of matrix A, except set $a_{11} = 1$

Once we know the voltage at each node, we can compute the current in each resistor from equations (1-4).

II. Program

1. **Download PSet10Files.zip from the Problem Set 10 webpage**. In it you will find 3 files:

- GridTest.java contains a main method you will need to complete.
- Resistor.java models the resistors
- Matrix.java contains gaussian() and other useful matrix methods.

The main method in GridTest.java defines six variables that represent the 18 node network in Figure 1:

- `iext=5`, the index of the node connected to the external power
- `ib=16`, the index of the node connected to the battery.
- `Vext=6` :the voltage at node `iext`, is 6
- `Vb=12`:the battery voltage

- `n=18`, the number of nodes.
- `resistors`, an `ArrayList` of `Resistor` objects, with a from-node, to-node, and r (R^{-1}), the inverse resistance of each resistor.

Node 5 is the only node whose voltage is determined by the external voltage

You will need to add code to this main method to complete it.

2. Write a class to model the electric grid.

The class that models the grid network must have:

- 6 private data members, to store each of the 6 variables above,
- A constructor with 6 arguments that initialize those data members, and
- 2 additional data members to store the A and b matrices that its methods will generate.

Within your grid network class, create three methods to prepare A and b using the following logic:

A. **Create A:** A is initially filled with zeros, from Java initialization. For each resistor t connected to nodes i and j
   - **Add** r_t to a_{ii} and a_{jj} cells of matrix A, i.e.:
     - $a_{ii} = a_{ii} + r_t$ and $a_{jj} = a_{jj} + r_t$
   - **Add** $-r_t$ to a_{ij} and a_{ji} cells of matrix A
     - $a_{ij} = a_{ij} - r_t$ and $a_{ji} = a_{ji} - r_t$

B. **Create b:**
   - b is initially filled with zeros, from Java initialization.
   - Find resistor ib—the resistor between node ib and node ib+1.
     - Add $Vb*r_{ib}$ to row ib in matrix b
     - Add $-Vb*r_{ib}$ to row ib+1 in matrix b. (This works only for the specific network given in Figure 1; it is not general.)

C. **Modify A & b:** This models the external source at node i_{ext}=5 with a known voltage V_{ext}. Modify A and b as follows:
   - Modify each entry in matrix b:
     - $b_i = b_i - V_{ext}\, a_{i\, iext}$
   - Set all entries in column `iext` and all entries in row `iext` of matrix A equal to zero, except set $a_{iext\, iext} = 1$

   These steps adjust for V_{ext} being known, and effectively remove one equation, so that we have n-1 equations for n-1 unknowns. These steps do the algebra to tie node i_{ext} with its neighbor nodes.

3. **The following items can be done in either your grid network class, or the main method**:

- Create the A and b matrices:
- Run the Gaussian elimination method from class `Matrix`, call the output x.

(This is provided to you. It's identical to the one used in lecture.)

- After Gaussian elimination, set $x_{iext} = V_{ext}$
- Output Matrix x, with the modified value of x_{iext}
- Output (System.out.println) Matrix x with 6 values per row, over 3 rows, to see the pattern of voltages in the same layout as the grid.
- Output the flows in each resistor. Compute:
    - $I_k = r_k (V_i - V_j)$, by looping through the list of resistors
    - With the exception of k=ib for which $I_{ib} = r_{ib} (V_b - V_i + V_j)$

Use `Matrix.java` as provided; it is the same as in lecture. You may find it convenient to use the `incrementElement()` method.

4. Complete the main() method in GridTest.java by creating an instance of your Grid network, calling its matrix generation methods, calling the Gaussian elimination method, and displaying the output.

Sample Output:

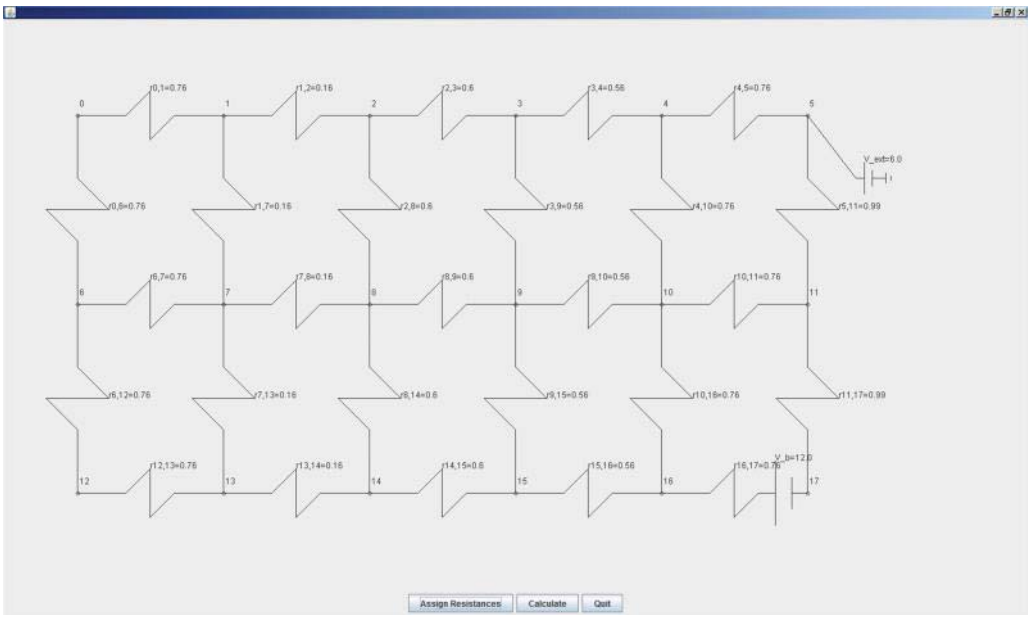
```
Voltages:
8.425212019003387
8.393268529067338
8.181187030628996
7.883149807710638
7.139553992500386
6.0
8.457155508939435
8.453618450309454
8.42266918729713
8.307420026936933
7.731195279056376
5.125190874444146
8.492636057505464
8.528116606071494
8.771147367522174
9.184433750800725
10.504254313891543
2.24981271100416

Voltages (matrix format):
 [+8.425] [+8.393] [+8.181] [+7.883] [+7.140] [+6.000]
 [+8.457] [+8.454] [+8.423] [+8.307] [+7.731] [+5.125]
 [+8.493] [+8.528] [+8.771] [+9.184] [+10.504] [+2.250]
Currents:
Resistor ( 0, 1): I=+0.024
Resistor ( 1, 2): I=+0.034
Resistor ( 2, 3): I=+0.179
Resistor ( 3, 4): I=+0.416
Resistor ( 4, 5): I=+0.866
Resistor ( 6, 7): I=+0.003
Resistor ( 7, 8): I=+0.005
Resistor ( 8, 9): I=+0.069
Resistor ( 9,10): I=+0.323
```

```
Resistor (10,11): I=+1.981
Resistor (12,13): I=-0.027
Resistor (13,14): I=-0.039
Resistor (14,15): I=-0.248
Resistor (15,16): I=-0.739
Resistor (16,17): I=+2.847
Resistor ( 0, 6): I=-0.024
Resistor ( 1, 7): I=-0.010
Resistor ( 2, 8): I=-0.145
Resistor ( 3, 9): I=-0.238
Resistor ( 4,10): I=-0.450
Resistor ( 5,11): I=+0.866
Resistor ( 6,12): I=-0.027
Resistor ( 7,13): I=-0.012
Resistor ( 8,14): I=-0.209
Resistor ( 9,15): I=-0.491
Resistor (10,16): I=-2.108
Resistor (11,17): I=+2.847
```

III. Extra Credit

In addition to your solution above, you may implement a Swing GUI for this problem set and get up to 40 extra credit points. **You must first complete and submit the entire non-GUI solution as outlined above.** Do not attempt to develop your GUI until you have completed the normal assignment since it will be graded separately. You should understand that a GUI solution often requires changes to the rest of your code. Therefore we require you to develop your GUI solution in a separate Eclipse project, copying all the files you need. When you submit your solution to the 1.00 Web site, first submit your original solution. Then upload your extra credit solution as a second .zip file. Both versions should contain all the files needed to compile and run your solution. **You can get extra credit on only one homework from problem sets 8 to 10.**

For extra credit, draw the network as shown in Figure 1. The user will input the number of east-west and north-south resistors in a grid pattern. The user will also input the node and voltage for the battery and external voltage and the inverse resistances for each resistor. After finding the solution, label each node with its voltage, and label each resistor with its current and an arrow indicating the direction of flow. Have a ‘calculate’ and ‘quit’ button. An example GUI is shown below:



(You don't have to include a button to assign resistance, but it might be helpful.)

Turn In

1. Place a comment with full name, section, TA name and assignment number at the beginning of all `.java` files in your solution. In this homework, you will have multiple `.java` files.
2. Place all of the files in your solution in a single zip file. **Do not include the extra credit in this file. You must submit a working, non-GUI solution to receive any credit.**
    a. Do not turn in electronic or printed copies of compiled byte code (`.class` files) or backup source code (`.java~` files)
    b. Do not turn in printed copies of your solution.
3. Submit this single zip file on the 1.00 Web site under the appropriate section and problem set number. For directions see **How To: Submit Homework** on the 1.00 Web site.
4. Your solution is due at noon. Your uploaded files should have a timestamp of no later than noon on the due date.
5. After you submit your solution, please recheck that you submitted your .java file. If you submitted your .class file, you will receive **zero credit.**
6. Place all of the extra credit files in a single zip file. Clearly label the file as extra credit and submit it on the 1.00 Web site.

Penalties

- 30 points off if you turn in your problem set after Friday noon but before noon on the following Monday. You have one no-penalty late submission per term for a turn-in after Friday noon and before Monday noon.
- No credit if you turn in your problem set after noon on the following Monday.

1.00 / 1.001 / 1.002 Introduction to Computers and Engineering Problem Solving
Spring 2012