UNIVERSITY OF TORONTO, DEPARTMENT OF ELECTRICAL & COMPUTER ENGINEERING

ECE1254H Modeling of Multiphysics Systems
Report #1

Chen Sun
1001418799

October 14, 2014

1 Problem # 1

1.1 (a)

The function [G,b] = NodalAnalysis(filename) that generates the modified nodal analysis has been formulated in Project 1. It reads data.txt's input as specified and generates the wanted MNA equation.

1.2 (b)

For include the controlled source into modified nodal analysis formulation, we need further to stamp the gains at the control nodes than the independent sources. For controlled voltages sources, we need form extra equation in MNA, e.g. add a row in the bottom part.

For a voltage-controlled voltage source (VCVS), with the element equation $v^+v^- = k(v_x^+ - v_x^-)$, the "stamp" for a VCVS into MNA has already shown in the following chart in [1].

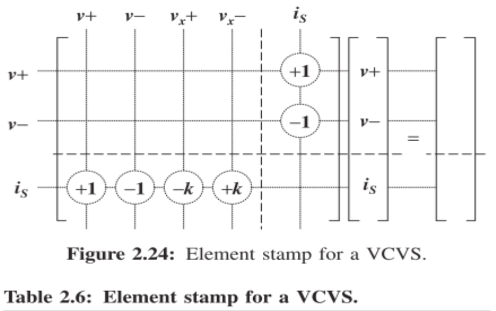


Figure 2.24: Element stamp for a VCVS.

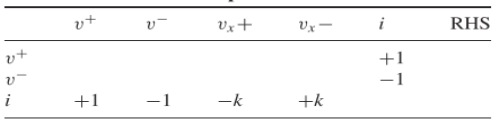


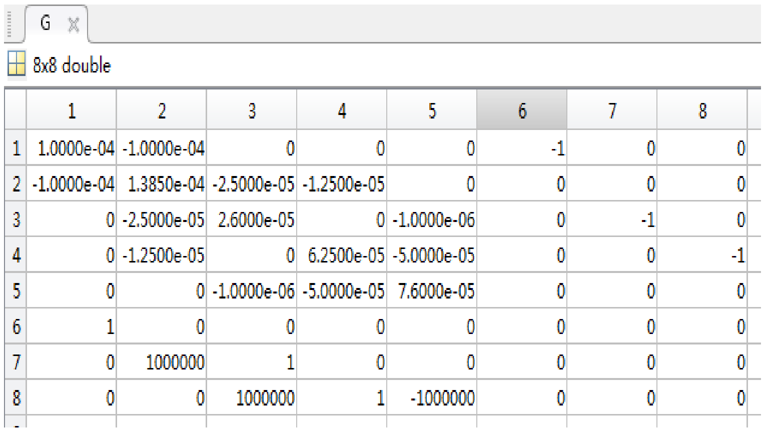
Table 2.6: Element stamp for a VCVS.

	v^+	v^-	v_x+	v_x-	i	RHS
v^+					$+1$	
v^-					-1	
i	$+1$	-1	$-k$	$+k$		

Figure 1: Element stamp for a VCVS

1.3 (c)

For the given circuit, a input file data.txt have been written for the netlist parser, which generate the matrix G as following Using the MATLAB command to solve the linear system and we have voltage V_0 as $V_0 = -11.4285V$


G

8x8 double

	1	2	3	4	5	6	7	8
1	1.0000e-04	-1.0000e-04	0	0	0	-1	0	0
2	-1.0000e-04	1.3850e-04	-2.5000e-05	-1.2500e-05	0	0	0	0
3	0	-2.5000e-05	2.6000e-05	0	-1.0000e-06	0	-1	0
4	0	-1.2500e-05	0	6.2500e-05	-5.0000e-05	0	0	-1
5	0	0	-1.0000e-06	-5.0000e-05	7.6000e-05	0	0	0
6	1	0	0	0	0	0	0	0
7	0	1000000	1	0	0	0	0	0
8	0	0	1000000	1	-1000000	0	0	0

Figure 2: Matrix G generated by the given circuit

1.4 (d)

Script solveproblem.m is the main program first using routine NodalAnalysis to generate matrix G, and then using my own LU factorization with partial pivoting routine (LUpartialpivot.m) and forward/backward substitution (FSM.m /BSM.m) routines to solve the circuit equations.
The computed V_0 by my own routines is still : $V_0 = -11.4285$

2 Problem # 2

2.1 (a)

The function netlistgenerator.m generates a netlist satisfy the conditions. After we get the netlist.txt, we can then solve them implementing my own LU routine and find the voltages, as the same procedure in Problem $1.d$.

For given $N = 50, R = 0.2$ condition, substitute them in SolvenodeV.m, we then first generate the netlist, and further solve and plot the node values as Z-axis in $3-D$, which shown as below. And here is an another view point of the picture.

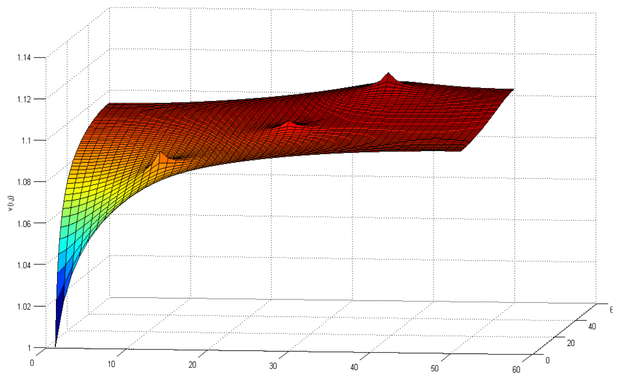


Figure 3: Node voltages for 50×50 grid

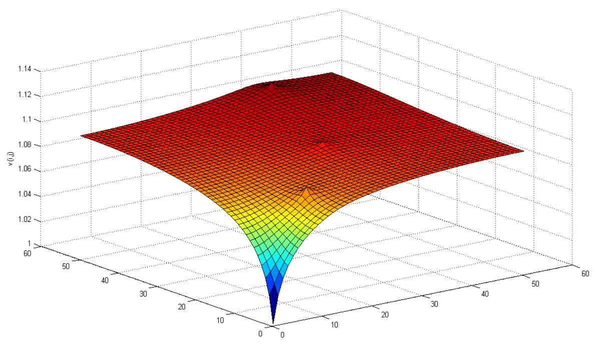


Figure 4: Node voltages for 50×50 grid

2.2 (b)

The routine $CPU_time.m$ plots the CPU time taken by the system solver I designed and its relationship between the size of MNA matrix G, as shown below

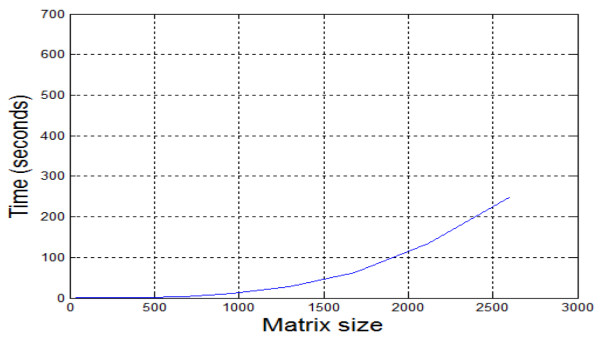


Figure 5: CPU time taken by the system

2.3 (c)

Since we have $t_{cpu}(n) \simeq Kn^{\alpha}$, which can be stated as $\log_n t = \alpha \log_n K$. From the routine we developed in Problem $2.(b)$, we can use variables like $\log tim_s tamp$ and $\log mat_s ize$ to compute α, which roughly at $n = 50$, $\alpha = 2.3$. However, this is not the case of large n.
Since my laptop's memory doesn't allow me to further plot the CPU time when $n > 55$, thus I can only say for a large n, the exponent will be nearly lie in a position around $\alpha = 3$, which in turn shows the cost time of this solver is $O(n^3)$.

2.4 (d)

For the dense LU method, e.g. as in (c) $n \to \infty$, the time complexity of this method is $O(n^3)$, and time cost only depends on size. For sparse LU, we can reach a lower time consume typically between $O(n)$ and $O(n^3)$, the time cost depends on both size and matrix sparsity.

3 Problem # 3

3.1 (a)

From $\frac{\partial^2 T(x)}{\partial x^2} = \frac{\kappa_a}{\kappa_m}(T(x) - T_0) - \frac{H(x)}{\kappa_m}$ and $\frac{\partial^2 T(x)}{\partial x^2} \simeq \frac{\frac{T(x+\Delta x)-T(x)}{\Delta x} - \frac{T(x)-T(x-\Delta x)}{\Delta x}}{\Delta x}$, we have following approximation

$$[\kappa_a(T(x) - T_0) - H(x)]\Delta x \simeq \kappa_m \frac{T(x+\Delta x) - T(x)}{\Delta x} - \kappa_m \frac{T(x) - T(x-\Delta x)}{\Delta x} \qquad (1)$$

Notice that resistance is analogy to the thermal conductance as $R_m = \frac{\Delta x}{\kappa_m}$, and $R_a = \frac{\Delta x}{\kappa_a}$, node temperature equivalent to the node voltage as $T(x) \leftrightarrow V(x)$ and heat flux analogy to the current source as $H(x) \leftrightarrow I(x)$, we can then rearrange (1) as KCL form

$$(\Delta x)^2 \frac{V(x) - V(0)}{R_a} - \Delta x I(x) - \frac{V(x+\Delta x) - V(x)}{R_m} + \frac{V(x) - V(x-\Delta x)}{R_m} = 0 \tag{2}$$

For computational implementation, if we choose n nodes, e.g. $i = 1 : n$ from $x \in [0, 1]$, the step $s = \frac{1}{n}$, we can obtain

$$\frac{1}{n^2}\frac{V_i - V_0}{R_a} - \frac{1}{n}I_i - \frac{V_{i+1} - V_i}{R_m} + \frac{V_i - V_{i-1}}{R_m} = 0 \tag{3}$$

Then the equivalent circuit is show below:

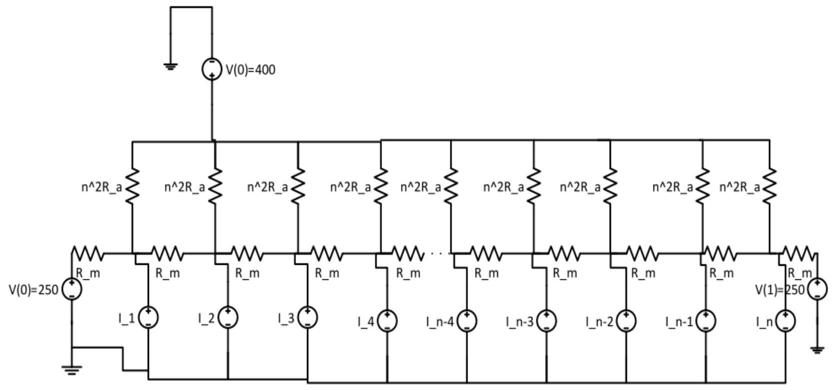


Figure 6: Equivalent Circuit (a)

3.2 (b)

First we need to generate input file barList.txt using the routine barList. Then, we should pass the input file to the routine NodalAnalysis developed previously in Problem 1a that generates G and b matrix which is then passed to system solver to obtain the temperature along the bar. Then, we have the $T(x)$ in $x \in [0, 1]$, as Fig. 7.

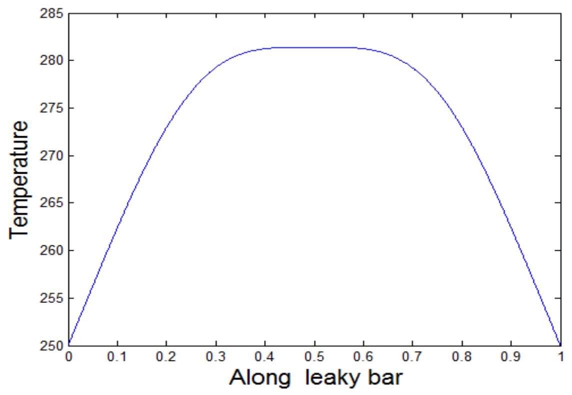


Figure 7: Temperature along the leaky bar

3.3 (c)

Since $H(x) = 50(sin(2\pi x))^2$, for $x \in [0, 1]$, for better simulation purpose, Δx should be small enough to make sure 1. correctly to represent the H(x) on discretization form; 2. make sure it give a smooth curve of the temperature distribution.
And for time saving purpose, we do not want the Δx be too small to make the computer run out of memory and be very time consuming.
Thus, I think step, e.g. Δx would be better in the range of $\Delta x = 0.01\ 0.001$.

3.4 (d)

Since there is given a zero-heat-flow boundary condition at both end, thus, analogy to circuit, which is that both end are open. Except for this, other elements follow the same as in (a). The analogy circuit is shown below in Fig 8.

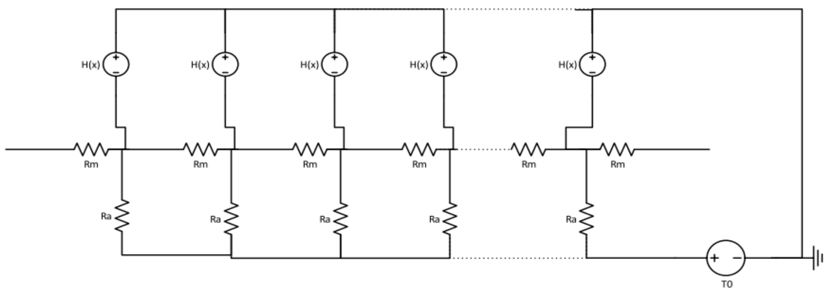


Figure 8: Equivalent Circuit (d)

Then, we can formulate routine as the same procedure in (b) to settle this problem. $barList2.m$ generates the input file and solved by the script $leakyBar_partD$.

3.5 (e)

The plotted temperature is shown below in Fig. 9

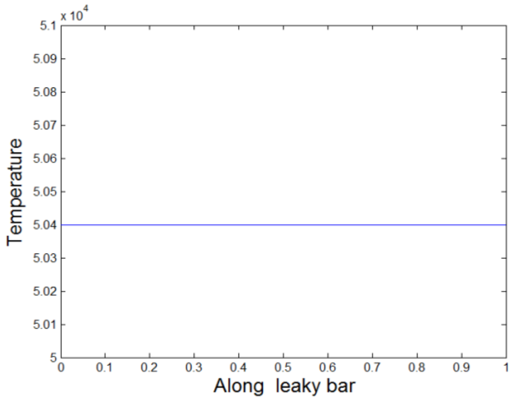


Figure 9: Temperature along the leaky bar (d)

3.6 (f)

In (b), the heat generated H(x) along the bar has a function of $[sin(x)]^2$, and so the temperature profile in turn, is related to $[sin(x)]^2$. The heat at the boundary point is fixed constant, thus the temperature along the bar dissipate quickly at the both ends of the bar, whereas the middle temperature is the highest.
In the second case, the heat generated H(x) is constant along the bar and in turn, the temperature profile along the bar is constant. The boundary are open, thus, the temperature can not be dissipated to the ends. Every node point share the same temperature. And also, we can see that the temperature goes very high, since there is no cool metal cabinet at boundary.

References

[1] F. N. Najm, Circuit simulation. John Wiley & Sons, 2010.