CSPICE

劉彥廷 B97901121
陳韋翰 B97901108

January 13, 2011

Review of previous researches

Our project is based on the FORTRAN code provided by professor. The reference code provided complete functions of time response and frequency response simulation. Our goal is to implement the functions in C/C++.

The method in the reference code is to find out the transfer function of the circuit, then simulate the circuit with the transfer function. The transfer function is generated by finding all the spanning trees in the graph model. The detail operation can be found in [SPC].

Description of methods and algorithms

We devided the project into two parts: parser and simulator. There are less technical problems in the parser part. We used Yacc and Lex to do the parsing job (see reference [LY]). We would like to discuss the details of simulator and the design of the data structure to store circuit information.

Circuit data structure

Since we are using node analysis in finding the transfer function, we must have a class Node to handle the node information. And there is another class Connection defines the connections between nodes, that is, the possible direction of currents. Combine nodes and connections, a graph $G = (V, E)$ is formed where V is the set of nodes and E is the set of connections.

We keep an array of Connections in the data structure Node indicating the connections that starts from this node. Also, a node pointer is kept in the Connection data structure to show the destination of the connection. Data structure Element is defined to represent the physical eletrical elements like resistors, capacitors, inductors, voltage controlled current source ... etc. There is a pointer in Connection pointing to Element to show what kind of element is on the path.

To make the work of simulation simple, we treated voltage sources and current sources differently than Elements. This is because superposition principle is used in simulating time response, thus using a different data structure may help when enumerating voltage and current sources.

Simulation

To find the denumerator and nominator of the transfer function, a DFS search on the graph is needed. The method is decribed in [SPC]. Complex arithmetics is contained in the C++ library,

which gives a lot of help on evaluating transfer function.

There is a problem occured when displaying transfer function: there are too many terms. We have to cross out the same terms to solve this problem. A hash function called FNV-1a (see reference [FNV]) is used to hash the string of term into integer, so that it's easier to find out which term should be crossed out.

The simulation of frequency response is rather easy, just plotting the data of the transfer function. To perform a simulation of time response, we need to do a reverse Laplace transform (we know that transfer function is the Laplace transform of the circuit). Trapezoidal rule is used in numerical integration since it's easy to implement and strike a balance between overestimate and underestimate of the integration value.

Program list

There is only one program generated after the compilation of the source code, and the usage is:

```
cspice [NETLIST] [PLOT DATA]
```

Test data format

Netlist format

```
# This is comment
# Elements
V[ID] [NODE1] [NODE2] [VALUE] # Voltage source
I[ID] [NODE1] [NODE2] [VALUE] # Current source
R[ID] [NODE1] [NODE2] [VALUE] # Resistor
L[ID] [NODE1] [NODE2] [VALUE] # Inductor
C[ID] [NODE1] [NODE2] [VALUE] # Capacitor
G[ID] [NODE1] [NODE2] [NODE3] [NODE4] [VALUE]
# VCCS: I = (VALUE) * (V1 - V2)

# Simulations
FREQ  [START] [END] [STEP] [SRC] [OUTPUT] # Frequency response
TIME  [START] [END] [STEP] [OUTPUT]       # Transient response
```

Sample Netlist

```
VIN 1   0   0   1
OUT 6   0
R1  1   2   0.5
R2  2   0   0.4
C1  2   6   0.001
C2  2   3   0.001
R3  3   6   88.0
R4  7   0   0.05
R5  7   6   100.0
```

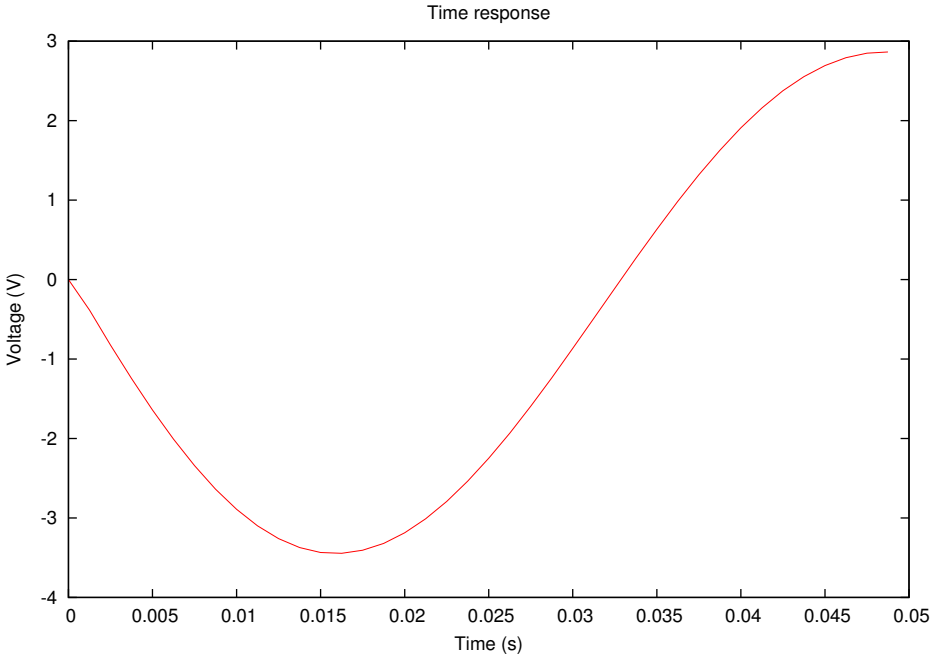

Figure 1: Time response

```
GM1 3   7   4   0   100.0
CI  4   0   1.0
RI1 4   0   100.0
GM2 4   0   5   0   -100.0
RI2 5   0   0.1
RO  5   6   -0.1

FREQ 10.0 100.0 20  VIN  freq.eps
TIME 0.0  0.05  0.002 time.eps
```

Results comparison

The simulation results are plotted with GNUPLOT software (see reference [GP]). Figure 1 is the time response and figure 2 is the frequency response. Both figures are match to the result in the reference program.

Conclusion

In these term project, we successfully transfered netlists into data stuctures, and used it to find out transfer function of the circuit and completed the simulation in both time response and frequency response. Several tools are used in this project, such as Yacc, Lex, Gnuplot, FNV-1a ... etc. Yet benchmarks of the performance is not complete because it is not easy to generate a meaningful large circuit. This is the direction we will continue to work on.

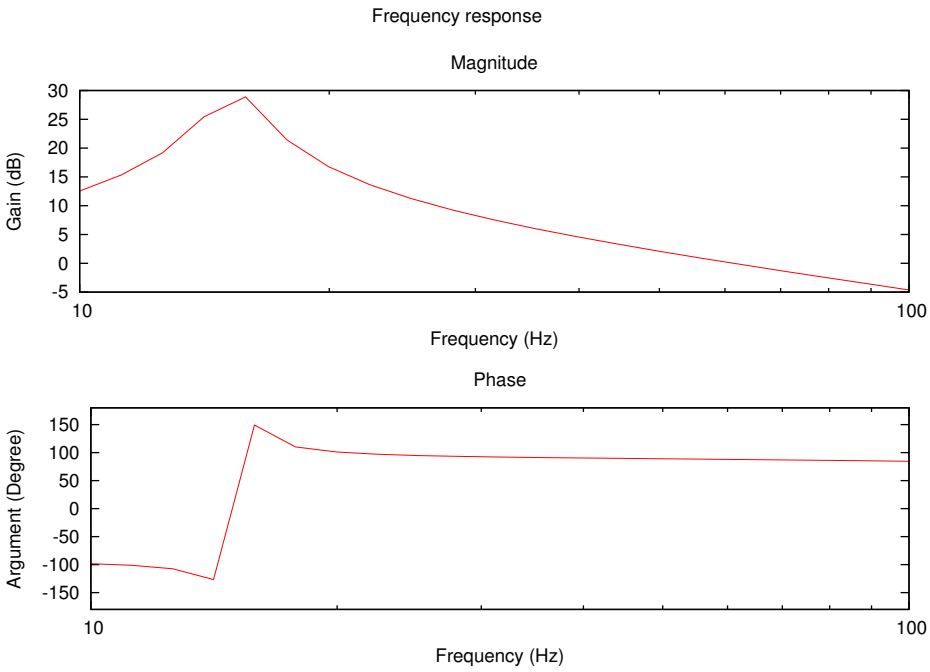

Figure 2: Frequency response

References

- [SPC] Shu-Park Chan, Graph Theory and Some of its Applications in Electrical Network Theory, SIAM-AMS proceedings Volume III, p45-64
- [LY] A Complete Guide to Lex & Yacc, http://epaperpress.com/lexandyacc/
- [GP] GNUPLOT 4.2 - A Brief Manual and Tutorial, http://www.duke.edu/∼hpgavin/gnuplot.html
- [FNV] FNV hash, http://isthe.com/chongo/tech/comp/fnv/