Computer - Aided Circuit Design and Analysis 2022 Spring

Project 1

Parser & Stamping

Deadline: 2022/4/27 23:59

Project Introduction

In this project, you are going to write a parser which can be suitable for circuit description inputs. After you finish parsing the input file, you should generate a MNA matrix presentation of the circuit.

1. Part1-Parser

Write a parser that can read a circuit specification in terms of a simple "language" that we now describe. The language is quite limited and restrictive, and represents the bare minimum that will be needed for subsequent projects in this book. The parser should not be case-sensitive, and should interpret any contiguous sequence of spaces or tabs as equivalent to a single space. Every line of the input file should describe a single circuit element, and the description of every circuit element should be given wholly within a single input line. The order of lines in the input file is immaterial and any characters following a % in an input line should be considered as comments and ignored. Circuit node names should be non-negative integers, from the set {0,1,2,…}, and the node name 0 should be reserved and used for the ground or reference node.

The accepted circuit elements and their specifications are given below. In this, the symbol <node.*>, where * can be any single alphanumeric character, denotes a node name. Specifically, <node.+> denotes the node that is the positive voltage reference point for the element and <node.-> denotes the negative reference node. The positive direction of current in any element is assumed to be from <node.+> to <node.->. The symbol <int> denotes a non-negative integer, and <value> denotes a non-negative real number. The <value> given for a circuit parameter, like resistance or capacitance, should be in the standard units: Volt, Ampere, Ohm, Farad, or Henry, but it should not include the corresponding unit. Finally, anything inside brackets, such as [G2] or [<value>] is an *optional field*.

• Voltage source: Only independent DC voltage sources are allowed, specified as:

V<int> <node.+> <node.-> <value>

• Current source: Only independent DC current sources are allowed, specified as:

I<int> <node.+> <node.-> <value>

• Resistor: Only linear resistors are allowed, specified as:

R<int> <node.+> <node.-> <value> [G2 % this is a group 2 element]

• Capacitor: Only linear capacitors are allowed, specified as:

C<int> <node.+> <node.-> <value>

• Inductor: Only linear inductors are allowed, and they should be specified as:

L<int> <node.+> <node.-> <value>

• Diode: The diode model, and its parameter values, will not be part of the input description. Instead, the model will be built into any subsequent simulation code that you will write and only the terminals should be specified here. Optionally, a scale factor can also be included so as to allow the specification of diodes that are larger than minimum size. The specification is:

D<int> <node.+> <node.-> [<value>]

• BJT: Similar to the diode model, only the terminals and an optional scale factor are given. Let QN denote an npn device and QP denote pnp; the specification is:

QN<int> <node.C> <node.B> <node.E> [<value>]

QP<int> <node.C> <node.B> <node.E> [<value>]

where the nodes represent the collector, base, and emitter terminals, respectively.

• MOSFET: Similar to the above, we give only the terminals and an optional scale factor, and the body terminal is to be ignored:

MN<int> <node.D> <node.G> <node.S> [<value>]

MP<int> <node.D> <node.G> <node.S> [<value>]

where MN denotes an n-channel device and MP is p-channel, and the nodes represent the drain, gate, and source terminals, respectively.

The parser should create a data structure, as a linked list of records, where each record describes a separate circuit element, including its terminals and parameter values. Test your parser on circuits of your choice.

An example of circuit description file is given in “circuit_netlist_case1.txt” and Fig. 1.

Example (circuit_netlist_case1.txt and its reference circuit)

circuit_netlist_case1.txt

```
V1   n5    0     2
V2   n3    n2    0.2
V3   n7    n6    2
I1   n4    n8    1e-3
I2   0     n6    1e-3
R1   n1    n5    1.5
R2   n1    n2    1
R3   n5    n2    50    G2 % this is a group 2 element
R4   n5    n6    0.1
R5   n2    n6    1.5
R6   n3    n4    0.1
R7   n8    0     1e3
R8   n4    0     10    G2 % this is a group 2 element
```

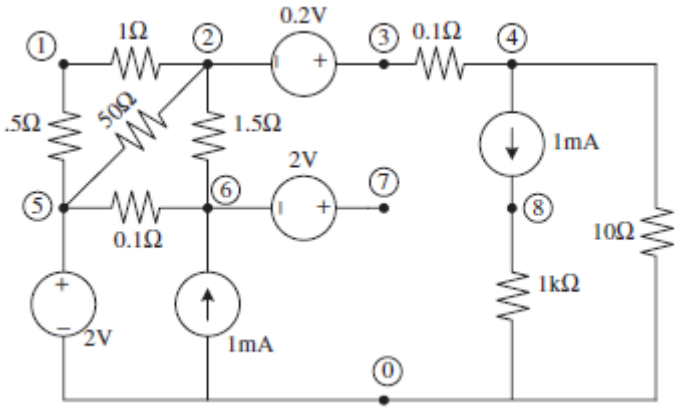

Figure 1: A reference circuit of input_circuitDiscription.txt

2. Part2-Stamping

Using the parser developed previously in **Part1** as a front-end, write a program that accepts a description of any ***resistive linear circuit***, with ***no controlled sources***, and constructs the corresponding MNA system using element stamps. In other words, your program should accept any network of linear resistors, independent current sources, and independent voltage sources. Your program should make use of the linked-list data structure created by the parser. It should interpret the optional field [G2] introduced earlier, in the specification of the parser, as indicating that an element belongs to group 2. The [G2] flag is not required for membership in group 2.

Test your program on the circuit shown in Fig. 1, where the 10Ω and 50Ω resistors are required to be in group 2. With the circuit description file given in input_circuitDiscription.txt, the correct solution is given in Fig. 2 and an example of the corresponding output format are given in output_mnaMatrix.txt, output_xVector.txt, and output_rhs.txt.

$$\begin{bmatrix} 10 & 0 & -10 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0.001 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -10 & 0 & 10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 5/3 & -2/3 & 0 & -1 & 0 & 0 & -1 & 0 & -1 & 0 \\ 0 & 0 & 0 & -2/3 & 32/3 & -10 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -10 & 32/3 & -2/3 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & -1 & 0 & -2/3 & 5/3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -10 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & -50 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} V(4) \\ V(8) \\ V(3) \\ V(2) \\ V(6) \\ V(5) \\ V(1) \\ V(7) \\ I(R8) \\ I(R3) \\ I(V3) \\ I(V2) \\ I(V1) \end{bmatrix} = \begin{bmatrix} -0.001 \\ 0.001 \\ 0 \\ 0 \\ 0.001 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 2 \\ 0.2 \\ 2 \end{bmatrix}$$

Figure 1: MNA matrix presentation

output_mnaMatrix.txt

$$\begin{bmatrix} 10 & 0 & -10 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0.001 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -10 & 0 & 10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 5/3 & -2/3 & 0 & -1 & 0 & 0 & -1 & 0 & -1 & 0 \\ 0 & 0 & 0 & -2/3 & 32/3 & -10 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -10 & 32/3 & -2/3 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & -1 & 0 & -2/3 & 5/3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -10 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & -50 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

output_xVector.txt

```
V(n4)
V(n8)
V(n3)
V(n2)
V(n6)
V(n5)
V(n1)
V(n7)
I(R8)
I(R3)
I(V3)
I(V2)
I(V1)
```

output_rhs.txt

```
-0.001
0.001
0
0
0.001
0
0
0
0
0
2
0.2
2
```

Language

C/C++

Platform

Linux (Please make sure your code is available on the given linux server).

Warning

(1) Several other circuit description files (hidden cases) will be put into your program for testing and be scored.

Naming, Programming, and Command rules

You should output the file of the results as the format shown in output.txt.
Your program should take the command-line arguments as follows:
./Project1 [input] [outputMNA] [outputXVec] [outputRHS]

example:
./Project1 circuit_netlist_case1.txt output_mnaMatrix.txt output_xVector.txt output_rhs.txt

Submission

Please upload the following materials in a .zip file (e.g. **Project_1_StudentID.zip**) to E3 before the deadline, specifying your student ID in the zip file.
(1) Source code (.cpp/.c, .h, or Makefile).
(2) Executable binary.
(3) A Readme file (Information of how to compile/execute your codes/program.)

Grading Policy

(1) **Plagiarism is not allowed.**
(2) **Correctness: 100%.** If the output files are correct, you will get 100 points. Note that several hidden cases will be tested on your program.
(3) The order of the output can be decided by yourself. The given order in the example is { V(n4), V(n8), V(n3), V(n2), V(n6), V(n5), V(n1), V(n7), I(R8), I(R3), I(V3), I(V2), I(V1)}. You can choose other order like {**V(n8), V(n4)**, V(n3), V(n2), V(n6), V(n5), V(n1), V(n7), I(R8), I(R3), I(V3), I(V2), I(V1)}, **but you should output the correct files of MNA matrix and RHS (rather than the files in previous example).**
(3) A Readme file (Information to how to compile and execute your code/program.)