A DC Circuit Simulator in MATLAB

Sebastian Magierowski

1 Objectives

1. To completely understand how the DC voltages and currents in a large resistive network can be calculate.
2. To automate the calculation of DC voltages and currents in a large resistive network in the form of a MATLAB program (version 7.0 recommended).

2 Introduction

The phenomenal success of the semiconductor electronics industry is in no small part aided by our ability to accurately simulate the performance of circuits before spending billions of dollars to build them. If you can make a better simulator you will be worth millions to semiconductor manufacturers. But the competition is stiff, Cadence, Synopsys, Avant, Mentor, are big players in the circuit simulation arena. Also leading IC manufacturers such as Intel, IBM, Texas Instruments, Analog Devices, have departments focused on supplying their designers with custom simulation tools. The people who write these simulators need to know their electric circuit fundamentals inside-out. Lest we forget other engineering disciplines it is important to point out that "circuit" simulators can have quite a broad range, especially nowadays as engineers seek ever closer links between once disparate designs. It all comes down to the fact that the KCL and KVL concepts can be generalized to many other phenomena. For example, today's simulators can model laser and photo diodes making them suitable for electro-optical systems (that's the backbone of the internet). They can also work with magnetic and power semiconductor models allowing them to handle power systems. The list can go on-and-on, even mechanical, thermal, acoustic and fluidic models can be incorporated within the context of the traditional circuit simulator.

In this project we are asking you to take the first step and write an automatic circuit simulator yourself. Don't worry, you are to program only the most basic of circuit simulators worthy of the name: a resistive network DC voltage and current solver. Here's how things will go:

1. We think of some crazy mixed up network that consists only of resistors and independent sources. The solution would not be very interesting if we did not have any independent sources alongside the resistors so expect your network to contain voltage sources or current sources or both.
2. We describe the circuit using a specialized text file, called a **netlist** (or a "deck" if you want to sound like an old pro in the circuit simulator game). You can find more on the netlist in Section 3.
3. We give you some MATLAB code that can read in a netlist and translate it into something better suited for MATLAB to handle.
4. At this point you've been given a circuit, you've been given code on how to read it into MATLAB, now all you have to do is to write a MATLAB program that calculates and **clearly** prints out:

   a. What the voltage between each node and ground (i.e. reference) is.

   b. What the current through each resistor in the circuit is.

   c. What the current through each independent voltage source in the circuit is.

And that's that, pretty easy, eh? Ok, here's some more info and background that you should probably know.

3 The Netlist

Check out the circuit in Fig. 1. How can you describe this in a text file? Well, it turns out that there's a standard way to do it and for the circuit in Fig. 1 it would look like this:

```
R1 4 0 199
R3 1 3 3.5
R4 1 2 12
R5 0 3 28
V8 1 4 1.8
V1 0 2 -5
I1 2 3 1
```

The above is the so-called netlist, introduced in the previous section. The first column of the netlist is

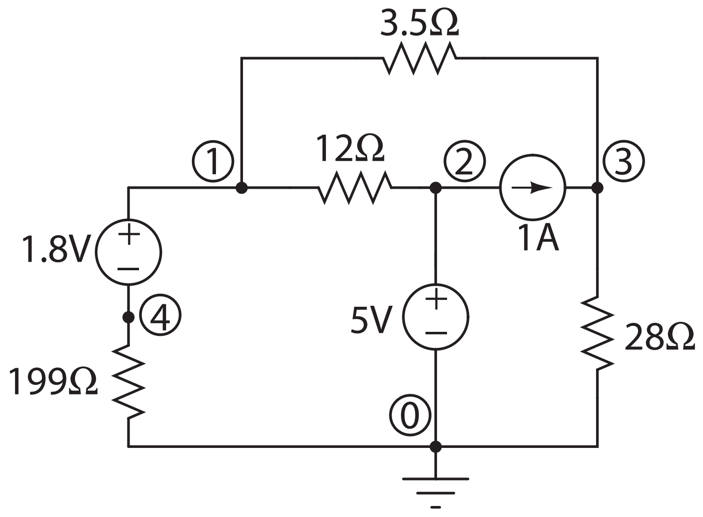

Figure 1: A very small example schematic of the type of circuit you may need to solve.

pretty self-explanatory, it contains all the names of each component. All resistor names must start with an 'R' and be followed by some unique (to the resistor) integer designation. Similarly, all independent voltage source names in the netlist must start with a 'V' and all independent current source names in the netlist must start with an 'I'. As with the resistors each is then followed by some integer designation to uniquely identify the object from other voltage or current sources. The second and third columns of the netlist denote the nodes between which each component lies. Although you can name a node anything you want in more sophisticated programs in this case we are only allowing netlists that label each node with some positive integer. How are the node numbers determined? For the most part this is completely arbitrary, in the netlist above we just decided to label one node with a '1' and another with a '3' and so on. The only reserved node number, the number '0', is set aside for the node you define as your reference value in the circuit that is, the ground node. As you know, every circuit needs some kind of reference node whose potential is fixed at zero so that you don't run into any ambiguities with KVL.

We will be providing you with a MATLAB function (more on this in Section 4) that reads in and translates netlists like the one shown above into a format easy for MATLAB to process. One limitation of this translator is that it requires the netlist node numbers to form a continuous sequence starting from 0. For example, it is perfectly fine to label a four-node-plus-reference circuit with `0,1,2,3,4`, but using `0,1,3,4,5` will not work.

For resistors it does not matter which order you enter their node connections, but you do have to be a little careful with the voltage and current sources. In the case of voltage sources the first node number listed (i.e. in the second column of the netlist) denotes the node connection of the positive end of the independent voltage source while the second node number listed (i.e. in the third column of the netlist) denotes the node connection of the negative end of the independent voltage source. In the case of current sources the order

of the node labels denotes the direction of the current flow. Thus, the line `I1 2 3 1` in the netlist example shown above indicates a current flowing from node 2 to node 3.
Incidentally, the value of the current from source I1 is 1 A. That's the purpose of the fourth and final column of the netlist, it contains the value of each component. Thus for resistors the fourth column denotes the component resistance in ohms, for voltage sources in denotes the component's voltage in volts, and, as already indicated, the fourth column denotes the current through current sources in amperes. Note that we were unnecessarily roundabout in describing the 5-V source between nodes 2 and 0. In the netlist we called this source V1 and recorded its voltage as being -5 V. What gives? Don't worry nothing is wrong, we did this just to give you more practice interpreting netlists. The netlist defines V1 as having its positive end connected to node 0 and having its negative end connected to node 2. Giving this arrangement a voltage of -5 V is equivalent to a voltage source with positive terminal connected to node 2, negative terminal connected to node 0, and value of +5 V between those nodes — exactly what is shown in Fig. 1. Similarly, we could have correctly described the current source as `I1 3 2 -1` in the netlist.

4 Reading in the Netlist

Out of great generosity we are providing you with MATLAB code that takes input netlists of the type described above and translates them into something a bit easier for MATLAB to digest. You are by no means obliged to use it. The code is shown and explained below.

```
clear all;
clc; % Clear the MATLAB Command Window

% 1. Read in the netlist text file into a netlist cell file
% ********************************************************
fid=fopen('ckt6.txt'); % Open the netlist text file
c=textscan(fid, '%s %s %s %s'); % Scan in the four columns into a cell
fclose(fid); % Close the netlist text file

% 2. Translate the netlist cell file into a netlist matrix
% ********************************************************
len=length(c{1}); % Total number of lines (rows) in the netlist
n=0;    % Initialize node counter, (total nodes - ref node) = n
for i=1:len
    for j=1:4
        f(i,j)=c{j}(i);
    end
    if str2num(f{i,2})>n
        n=str2num(f{i,2});
    end
    if str2num(f{i,3})>n
        n=str2num(f{i,3});
    end
end

% 3. Print out netlist matrix to MATLAB Command Window
% ***************************************************
fprintf('Input Circuit:\n');
for i=1:len
    fprintf('\t%s\t%s\t%s\t%s\n',f{i,1},f{i,2},f{i,3},f{i,4});
end

% 4. Count up resistors, voltage, and current sources in circuit
% *************************************************************
```

```
m=0;    %how many voltage sources
p=0;    %how many resistors
q=0;    %how many current sources
%count m,p,q
for i=1: len
    if (f{i,1}(1)=='V') m=m+1;
    end
    if (f{i,1}(1)=='R') p=p+1;
    end
    if (f{i,1}(1)=='I') q=q+1;%when use?
    end
end

% 5. Place resistor, voltage, and current data in their own matrices
% ****************************************************************
vdata=zeros(m,3);   %voltage source info matrix (mx3)
rdata=zeros(p,3);   %resistor info matrix (px3)
idata=zeros(q,3);   %current source info matrix (qx3)
nv=0;nr=0;ni=0;     %counter for voltage source, resistor and current source
%create matrix of vdata,rdata and idata
for i=1:len
    if (f{i,1}(1)=='V')
        nv=nv+1;
        vdata(nv,1)=str2num(f{i,2});
        vdata(nv,2)=str2num(f{i,3});
        vdata(nv,3)=str2num(f{i,4});
    end
    if (f{i,1}(1)=='R')
        nr=nr+1;
        rdata(nr,1)=str2num(f{i,2});
        rdata(nr,2)=str2num(f{i,3});
        rdata(nr,3)=str2num(f{i,4});
    end
    if (f{i,1}(1)=='I')
        ni=ni+1;
        idata(ni,1)=str2num(f{i,2});
        idata(ni,2)=str2num(f{i,3});
        idata(ni,3)=str2num(f{i,4});
    end
end
```

The operation of each main section is summarized in the comments. In a nutshell, this program reads in a netlist text file and generates three matrices: rdata, vdata, and idata. Each row of rdata stores the essentials of each resistor in the netlist, it's nodes in the first two columns and the resistance value in the third column. Same idea for vdata and idata.

5 Processing the Circuit

Now, with schematic descriptions and netlist processing out of the way you may actually be quite curious as to how your program will satisfy our demands for points **a.**, **b.**, and **c.**, listed in Section 2. Probably the best way to go about it is to have your program analyze the circuit using strictly KCL. You can probably appreciate this almost immediately since the nodes are automatically denoted for you (via their labels in the netlist) whereas finding loops for KVL analysis seems, at first glance, like a more challenging task to program.

If you settle on KCL, the next obvious step towards the circuit solution is to identify all nodal equations. Let's stick with the example shown in Fig. 1 and get KCL done for that (refer to Fig. **??** for the meaning of certain variables in the KCL equations below)

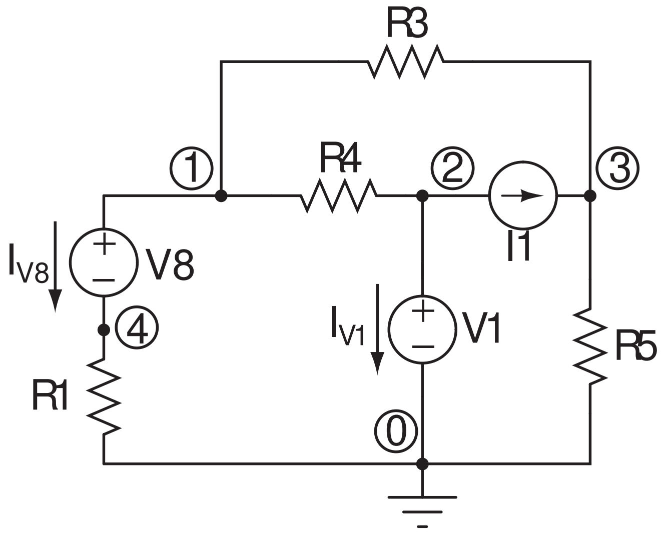

Figure 2: Writing the KCL equations for this circuit.

$$
\begin{aligned}
\text{node 1: } & \frac{V_1 - V_2}{R4} + \frac{V_1 - V_3}{R3} + I_{V8} = 0 \\
\text{node 2: } & \frac{V_2 - V_1}{R4} + I1 + I_{V1} = 0 \\
\text{node 3: } & \frac{V_3 - V_1}{R3} - I1 + \frac{V_3}{R5} = 0 \\
\text{node 4: } & \frac{V_4}{R1} - I_{V8} = 0.
\end{aligned}
$$

The equations as written leave us a little short, we have six unknowns (the nodal voltages V_1, V_2, V_3, V_4, and two voltage source currents I_{V1} and I_{V8}) and only four equations. Never fear however, since we can easily tack on two more equations by writing the voltage sources in terms of the nodal voltages, mainly

$$
\begin{aligned}
V_1 - V_4 &= V8 \\
V_2 &= V1.
\end{aligned}
$$

You'll find that you can do this with every circuit we'll be giving you (i.e. tack on a KVL of sorts around the independent voltage sources) to get the necessary number of equations. Now what? Easy, now solve for all the unknowns, rather, get the computer to solve all the unknowns. Good luck.