User's Guide for SCAM: Symbolic Circuit Analysis with MATLAB®

Overview

This document describes SCAM, a MATLAB® script that performs all of the manipulations needed to construct the Modified Nodal Analysis matrix equation describing a circuit, and then *solves* it symbolically. Circuits can include independent sources, resistors, reactive elements, OpAmps, and dependent sources that are described in a netlist similar to that required by industry-standard SPICE software.

Getting Started

First of all, download the compressed file `scam.zip` from the class website and save it on your computer. Unzip it and save the extracted folder and its contents in your user directory as `C:\Users\`*`username`*`\Documents\MATLAB\scam\`. Then make sure your MATLAB path includes that folder.

Netlists for the examples discussed in this document are a part of the zip file you downloaded, so they are also included in this folder.

Note: You must have the MATLAB® Symbolic Math and Control System Toolboxes installed in order to run this script.

This document, and the software described herein, are adapted with permission from software originally created by Eric Cheever of Swarthmore College. The associated license information is cited below.

Introduction

The SCAM code is written as a MATLAB® script instead of a function so that it can be stepped through to make learning easier, and all of the variables created by the code appear in the workspace so users can examine and manipulate them. If you don't want all of the variables in your workspace, it is straightforward to add a line at the top to turn it into a function. If you don't want all the intermediate results printed, simply comment out the lines you don't want.

There is essentially no error checking, so if you enter a netlist that isn't correct the program will probably fail without explaining why.

Notational conventions:

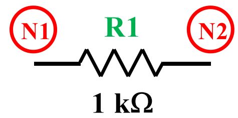
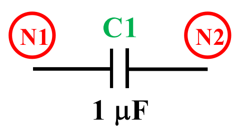
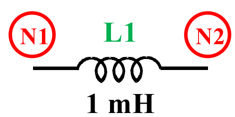
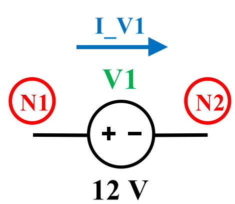
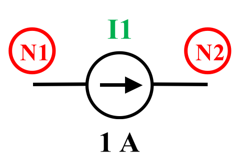
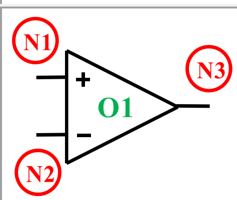
- Ground, or reference node, must be labeled as node 0.
- The remaining nodes must be labeled consecutively from 1 to *n*.
- The voltage at node 1 is referred to as v_1, at node 2 as v_2, and so on. Note the ***lower-case*** v's, and that the underscore must be included.
- Independent voltage source names must start with the letter "V" and must be unique from any node names. In addition, independent voltage sources have no underscore ("_") in their names. So, Va, Vsource, V1, and Vxyz123 are all legitimate names, but V_3, V_A, and Vsource_1 are not. Note the ***upper-case*** V's.
- The current through a voltage source will be labeled with "I_" followed by the name of the voltage source. Therefore, the current through Va is I_Va, the current through VSource is I_VSource, etc...
- Independent current source names must start with the letter "I" and must have no underscore ("_"). So, Ia, Isource, I1, and Ixyz123 are all acceptable names, but I_3, I_A, and Isource_1 are not.

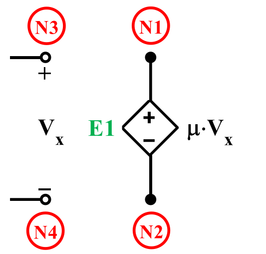
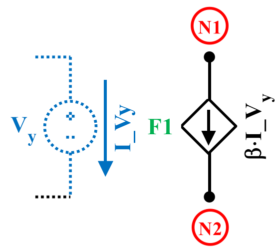
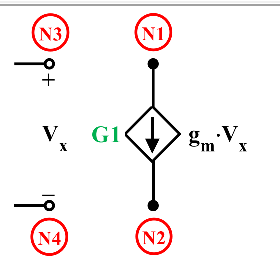
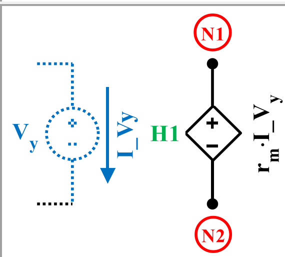
The Netlist

SCAM requires the circuit to be described by a text-based netlist file that describes the circuit elements and defines the interconnections between them. If you have used SPICE (Simulation Program with Integrated Circuit Emphasis) this is a familiar concept. As in SPICE, the file name for the netlist must have the extension ".cir", such as Sample.cir, Example_1.cir, or Voltage_Divider.cir.

The netlist has one line for each component in the circuit. Each type of component is described by a unique syntax, as shown in Table 1.

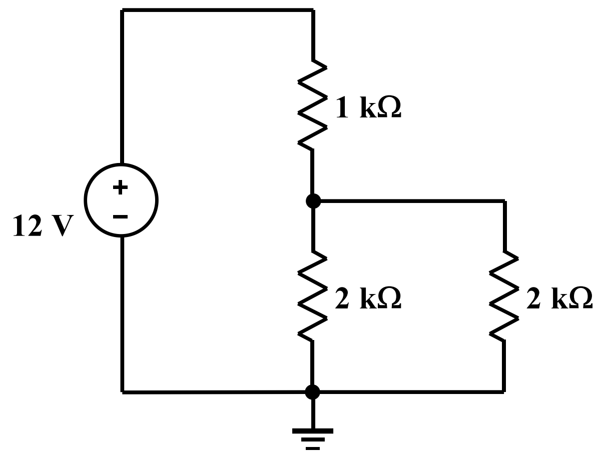
Table 1. SCAM Netlist Syntax

Component Type	Symbol	Netlist Description
Resistor		**R1 N1 N2 1000** • Resistor R1 is connected between nodes N1 and N2 and has a value of 1 kΩ. • The value of the resistor must be written out as a number (no abbreviations like k or kohms). • The name of the component is Rx, where "x" can be any combination of letters and numbers. R1, Ra, and Rload are all valid names.
Capacitor		**C1 N1 N2 1E-6** • Capacitor C1 is connected between nodes N1 and N2 and has a value of 1 μF. • The value of the capacitor must be written out as a number (no abbreviations like u or uF). • The name of the capacitor is Cx, where "x" can be any combination of letters and numbers. C1, C, and Cap are all valid names.
Inductor		**L1 N1 N2 1E-3** • Inductor L1 is connected between nodes N1 and N2 and has a value of 1 mH. • The value of the inductor must be written out as a number (no abbreviations like m or mH). • The name of the inductor is Lx, where "x" can be any combination of letters and numbers. L1, La, and Lprimary are all valid names.
Independent Voltage Source		**V1 N1 N2 12** • V1 is connected between nodes N1 and N2 and has a value of 12 V. • The positive terminal is connected to node N1, and the negative terminal is connected to node N2. • The value of the source must be written out as a number (no abbreviations like m or mV). • The name of the source is V*x*, where "x" can be any combination of letters and numbers. V1, Vs, and Vsrc are all valid names. • The current through the source is one of the unknowns, with reference direction chosen to satisfy the Passive Sign Convention as shown.
Independent Current Source		**I1 N1 N2 1** • I1 is connected between nodes N1 and N2 and has a value of 1 A. • Current flows through the source from node N1 to node N2. • The value of the source must be written out as a number (no abbreviations like m or mA). • The name of the source is Ix, where "x" can be any combination of letters and numbers. I1, Is, and Isrc are all valid names.
Op Amp		**O1 N1 N2 N3** • The non-inverting terminal of O1 is connected to node N1, the inverting terminal is connected to node N2, and the output terminal is connected to node N3. • The opamp is ideal. ($R_{in} = \infty$, $A_v = \infty$, $R_{out} = 0$).

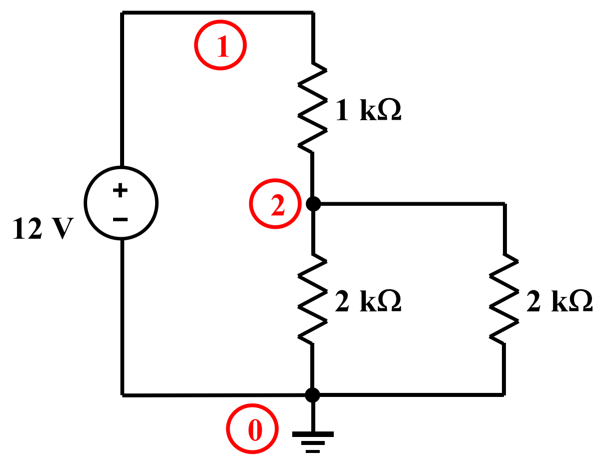
		• The name of the opamp is Ox, where “x” can be any combination of letters and numbers. O1, OpAmp, and OA are all valid names.
VCVS		**E1 N1 N2 N3 N4 μ** • The positive terminal of E1 is connected to node N1, the negative terminal is connected to N2, and the value of the controlled-source voltage is μ·Vx V, where Vx is defined between two nodes elsewhere in the circuit. • The current through E1 is one of the unknowns. • The name of the VCVS is Ex, where “x” can be any combination of letters and numbers. E1, Ea, and Eamp are all valid names.
CCCS		**F1 N1 N2 Vy β** • F1 is connected between nodes N1 and N2, and the value of the controlled-source current is β·I_Vy A, where Vy is an independent voltage source elsewhere in the circuit. • Current flows through F1 from node N1 to node N2. • The name of the CCCS is Fx, where “x” can be any combination of letters and numbers. F1, Fxyz, and Fsrc are all valid names.
VCCS		**G1 N1 N2 N3 N4 g_m** • F1 is connected between nodes N1 and N2, and the value of the controlled-source current is g_m·Vx A, where Vx is defined between two nodes elsewhere in the circuit. • Current flows through G1 from node N1 to node N2. • The name of the VCCS is Gx, where “x” can be any combination of letters and numbers. G1, Gxyz, and Gabc are all valid names.
CCVS		**H1 N1 N2 Vy r_m** • The positive terminal of H1 is connected to node N1, the negative terminal is connected to N2, and the value of the controlled-source voltage is r_m·Vy, where Vy is defined between two nodes elsewhere in the circuit. • The current through H1 is one of the unknowns. • The name of the CCVS is Hx, where “x” can be any combination of letters and numbers. H1, Ha, and Hello are all valid names.

Example 1:

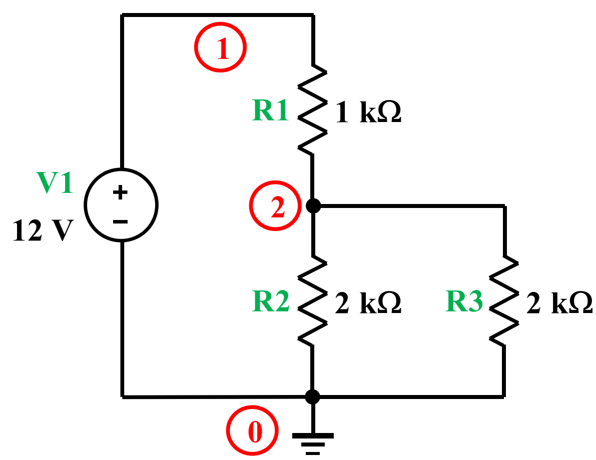
Use SCAM to analyze the circuit below:



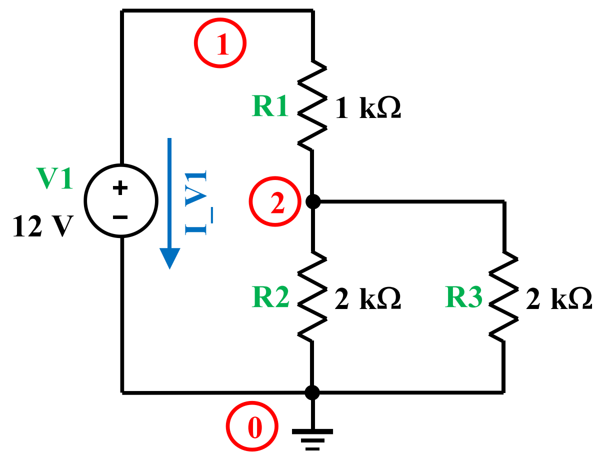
Start by defining the nodes. The only restriction here is that the nodes must be labeled such that ground is node 0, and the other nodes are numbered consecutively starting at 1. The choice of which number to assign to which node is entirely arbitrary.



Now label the components.



Identify the current through the voltage source (must satisfy the PSC). This is not really necessary while creating the netlist, but will be useful later when interpreting the results.



This circuit has 4 components (3 resistors and 1voltage source) and will require 4 lines to describe it.

The netlist is then:

```
V1 1 0 12
R1 1 2 1000
R2 2 0 2000
R3 2 0 2000
```

Create a text file containing the netlist and save it in your MATLAB® working directory. You can use the MATLAB® editor to do this. (Suggestion: name it *example1.cir*. All of the example files discussed here are included in the package you downloaded when installing SCAM earlier.) To run the program, assign the filename of the circuit to be analyzed to the variable *fname*, and then call the program. The interaction from the MATLAB® window is shown below:

Note: The version of the software provided here at ISU is called "ISU_scam" to distinguish it from the original file that had some minor errors in it. Most, but not all, of the time they work the same way.

MATLAB

```
>> fname="example1.cir";
>> ISU_scam

Started -- please be patient.

Netlist:
V1 1 0 12
R1 1 2 1000
R2 2 0 2000
R3 2 0 2000
```

```
The A matrix:
[ 1/R1,              -1/R1, 1]
[-1/R1, 1/R1 + 1/R2 + 1/R3, 0]
[    1,                  0, 0]


The x vector:
 v_1
 v_2
I_V1


The z vector:
 0
 0
V1


The matrix equation:
           I_V1 + v_1/R1 - v_2/R1 == 0
v_2*(1/R1 + 1/R2 + 1/R3) - v_1/R1 == 0
                             v_1 == V1


The solution:
                                      v_1 == V1
      v_2 == (R2*R3*V1)/(R1*R2 + R1*R3 + R2*R3)
I_V1 == -(V1*(R2 + R3))/(R1*R2 + R1*R3 + R2*R3)


Elapsed time is 0.172522 seconds.
```

The netlist is displayed, followed by the **A** matrix, and the **x** and **z** vectors, and the equations they describe are written out. Finally, the values of the unknown variables are displayed ***symbolically***. If you want to know the expression for the value of v_2 (the voltage at node 2), just type "v_2" at the prompt:

```
>> v_2

v_2 =

(R2*R3*V1)/(R1*R2 + R1*R3 + R2*R3)
```

or for the current through the voltage source, type "I_V1":

```
>> I_V1

I_V1 =

-(V1*(R2 + R3))/(R1*R2 + R1*R3 + R2*R3)
```

In addition to the unknowns (the node voltages and the voltage-source current), several other variables are created in the workspace (this is why the SCAM program is a *script* instead of a *function*). The important variables created, in addition to the unknowns, include a value corresponding to each of the circuit elements. We can examine the value of any element by typing its name at the prompt. For example, type "V1" for the value of the voltage source, or "R2" for the value of resistor R2:

```
>> V1

V1 =

    12

>> R2

R2 =

        2000
```

We can use also these values to get ***numerical*** values for the unknowns, as follows:

```
>> eval(v_2)

ans =

     6

>> eval(I_V1)

ans =

   -0.0060
```

Here we see that the voltage at node 2 is 6 volts, and the current through V1 is -6 mA.

What if we are interested in the current through R2? We know that the current through R2 is just the voltage drop across R2, divided by the value of the resistance. We can determine the solution both symbolically and numerically:

```
>> v_2/R2

ans =

(R2*R3*V1)/(2000*(R1*R2 + R1*R3 + R2*R3))

>> eval(ans)

ans =
```

```
    0.0030
```

The current through R1, connected between node 1 and node 2, is just the voltage across R1 divided by its value:

```
>> (v_1-v_2)/R1

ans =

V1/1000 - (R2*R3*V1)/(1000*(R1*R2 + R1*R3 + R2*R3))

>> eval(ans)

ans =

    0.0060
```

Other quantities can be similarly determined. For example, the ratio of v_2 to V1:

```
>> v_2/V1

ans =

(R2*R3*V1)/(12*(R1*R2 + R1*R3 + R2*R3))

>> eval(ans)

ans =

    0.5000
```

The SCAM program also determines the **A** matrix and the **x** and **z** vectors for the MNA method.

```
>> A

A =

[ 1/R1,              -1/R1, 1]
[-1/R1, 1/R1 + 1/R2 + 1/R3, 0]
[    1,                  0, 0]

>> x

x =

 v_1
 v_2
I_V1
```

```
>> z

z =

 0
 0
V1
```

We can use these variables to recreate the circuit equations. To get the left side of the matrix equation just multiply A*x:

```
>> A*x

ans =

           I_V1 + v_1/R1 - v_2/R1
v_2*(1/R1 + 1/R2 + 1/R3) - v_1/R1
                              v_1
```

MATLAB's "*pretty*" command will display the result in a slightly easier to read form:

```
>> pretty(A*x)
/              v_1    v_2      \
|      I_V1 + --- - ---        |
|              R1     R1       |
|                              |
|     /  1     1     1 \   v_1 |
| v_2 | -- + -- + -- | - --- |
|     \ R1    R2    R3 /    R1 |
|                              |
\              v_1             /
```

The right side of the matrix equation is given by **z**:

```
>> z

z =

 0
 0
V1
```

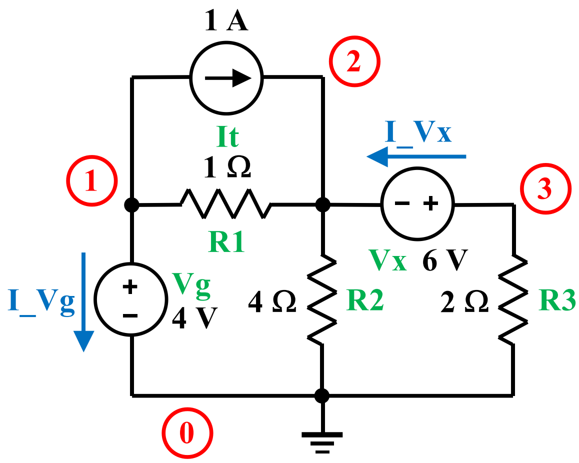
Using the information above, we can get any of the MNA equations. For example, to get the equation for node 2, simply take the 2nd row of the left and right sides of the equation as follows:

```
>> q=A*x==z

q =

           I_V1 + v_1/R1 - v_2/R1 == 0
```

```
v_2*(1/R1 + 1/R2 + 1/R3) - v_1/R1 == 0
                                v_1 == V1

>> q(2)

ans =

v_2*(1/R1 + 1/R2 + 1/R3) - v_1/R1 == 0

>> pretty(q(2))
    /  1     1     1 \    v_1
v_2 | -- + -- + -- | - --- == 0
    \ R1    R2    R3 /     R1
```

Example 2:

We can also apply the program to more complex circuits, such as the following. Here a value has been given to each component, nodes are already labeled, and the currents through the voltage sources are also labeled.



The netlist for this circuit is given by

```
Vg 1 0 4
It 1 2 1
R1 1 2 1
R2 2 0 4
Vx 3 2 6
R3 3 0 2
```

Enter this into a file and name it *example2.cir*. To analyze the circuit, set the *fname* variable, and start the program:

MATLAB

```
>> fname="example2.cir"

fname =

    "example2.cir"

>> ISU_scam

Started -- please be patient.

Netlist:
Vg 1 0 4
It 1 2 1
R1 1 2 1
```

```
R2 2 0 4
Vx 3 2 6
R3 3 0 2

The A matrix:
[ 1/R1,       -1/R1,    0, 1,  0]
[-1/R1, 1/R1 + 1/R2,    0, 0, -1]
[    0,           0, 1/R3, 0,  1]
[    1,           0,    0, 0,  0]
[    0,          -1,    1, 0,  0]


The x vector:
 v_1
 v_2
 v_3
I_Vg
I_Vx


The z vector:
-It
 It
  0
 Vg
 Vx


The matrix equation:
          I_Vg + v_1/R1 - v_2/R1 == -It
v_2*(1/R1 + 1/R2) - v_1/R1 - I_Vx == It
                     I_Vx + v_3/R3 == 0
                              v_1 == Vg
                        v_3 - v_2 == Vx


The solution:

v_1 == Vg
                v_2 == (R2*(R3*Vg - R1*Vx + It*R1*R3))/(R1*R2 + R1*R3
+ R2*R3)
        v_3 == (R3*(R2*Vg + R1*Vx + R2*Vx + It*R1*R2))/(R1*R2 + R1*R3
+ R2*R3)
I_Vg == -(R2*Vg + R3*Vg + R2*Vx + It*R1*R2 + It*R1*R3)/(R1*R2 + R1*R3
+ R2*R3)
           I_Vx == -(R2*Vg + R1*Vx + R2*Vx + It*R1*R2)/(R1*R2 + R1*R3
+ R2*R3)


Elapsed time is 0.279261 seconds.
```

We can solve for the voltage at node 3 either symbolically or numerically:

```
>> v_3

v_3 =

(R3*(R2*Vg + R1*Vx + R2*Vx + It*R1*R2))/(R1*R2 + R1*R3 + R2*R3)

>> eval(v_3)

ans =

    7.1429
```

and we can find the current through R1 (symbolically or numerically):

```
>> (v_1-v_2)/R1

ans =

Vg - (R2*(R3*Vg - R1*Vx + It*R1*R3))/(R1*R2 + R1*R3 + R2*R3)

>> eval(ans)

ans =

    2.8571
```

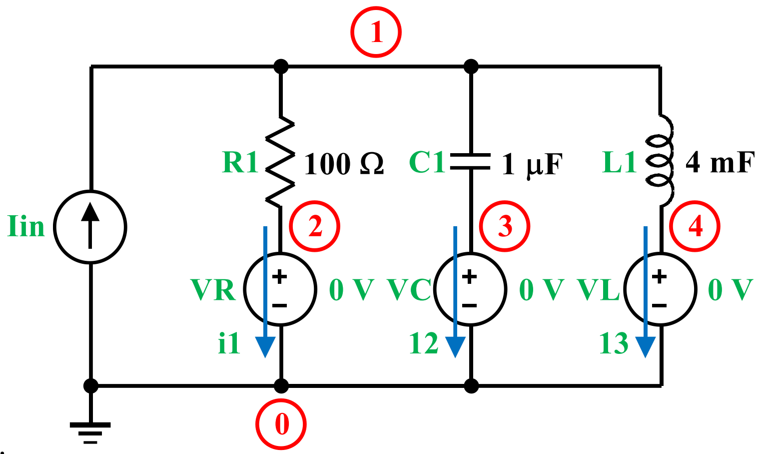
We can find the MNA equation for node 3 as follows:

```
>> q=A*x==z

q =

          I_Vg + v_1/R1 - v_2/R1 == -It
v_2*(1/R1 + 1/R2) - v_1/R1 - I_Vx == It
                     I_Vx + v_3/R3 == 0
                              v_1 == Vg
                        v_3 - v_2 == Vx

>> q(3)

ans =

I_Vx + v_3/R3 == 0

>> pretty(q(3))
       v_3
I_Vx + --- == 0
        R3
```

Example 3:

Now consider the circuit shown below.

Circuit



Netlist

```
Is 0 1 Symbolic
R1 1 2 100
VR 2 0 0
C1 1 3 1E-6
VC 3 0 0
L1 1 4 0.004
VL 4 0 0
```

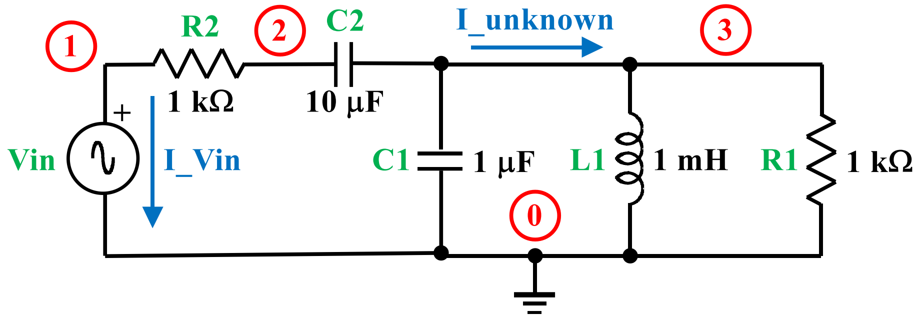
MATLAB

```
>> fname="example3.cir";
>> ISU_scam

Started -- please be patient.
```

Plot currents vs frequency

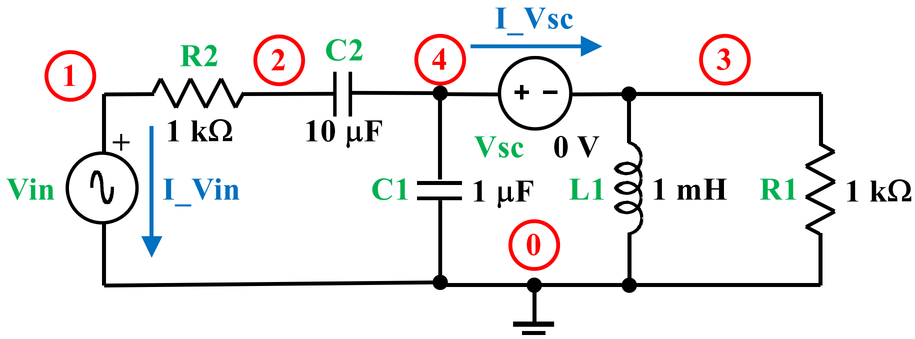
Example 4:

Now consider the case of finding the current through a wire. In particular, consider the circuit below, with the requirement that we would like to find the current shown as I_unknown:



How do we do this? One way would be to find I_C2 and I_C1 and apply KCL, another equivalent approach would be to determine I_L1 and I_R1 and apply KCL. Alternately, we could introduce a voltage source of zero volts (i.e., a short circuit), and solve for I_unknown as part of the MNA analysis. Note the additional node 4 that is introduced because the voltage source now splits the original node 3 into two nodes. The corresponding netlist is *example4.cir*.

Circuit



Netlist

```
Vin 1 0 Symbolic
R2 1 2 1000
C2 2 4 10E-6
C1 4 0 1E-6
L1 3 0 0.001
R1 3 0 1000
Vsc 4 3 0
```

This voltage source does not affect the circuit, but forces the computation of the current I_Vsc.

MATLAB

```
>> fname="example4.cir";
>> ISU_scam
```

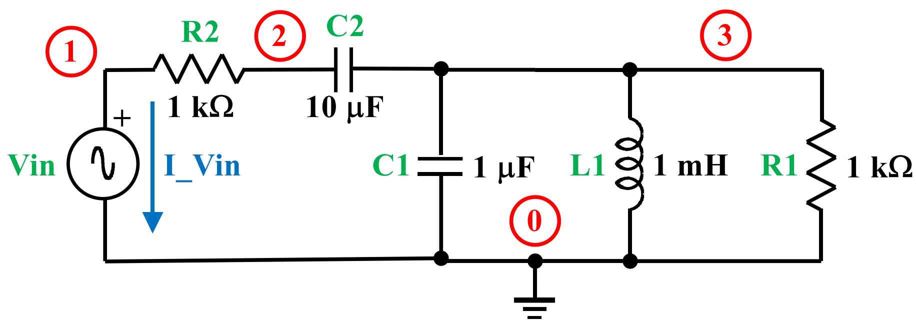
```
Started -- please be patient.

Netlist:
Vin 1 0 Symbolic
R2 1 2 1000
C2 2 4 10E-6
C1 4 0 1E-6
L1 3 0 0.001
R1 3 0 1000
Vsc 4 3 0


The A matrix:
[ 1/R2,       -1/R2,                0,           0, 1,  0]
[-1/R2, C2*s + 1/R2,                0,       -C2*s, 0,  0]
[    0,           0, 1/R1 + 1/(L1*s),           0, 0, -1]
[    0,       -C2*s,                0, C1*s + C2*s, 0,  1]
[    1,           0,                0,           0, 0,  0]
[    0,           0,               -1,           1, 0,  0]


The x vector:
  v_1
  v_2
  v_3
  v_4
I_Vin
I_Vsc


The z vector:
  0
  0
  0
  0
Vin
Vsc


The matrix equation:
              I_Vin + v_1/R2 - v_2/R2 == 0
v_2*(C2*s + 1/R2) - v_1/R2 - C2*s*v_4 == 0
        v_3*(1/R1 + 1/(L1*s)) - I_Vsc == 0
 I_Vsc + v_4*(C1*s + C2*s) - C2*s*v_2 == 0
                              v_1 == Vin
                        v_4 - v_3 == Vsc


The solution:
```

```
v_1 == Vin
v_2 == (R1*Vin + L1*Vin*s + C2*R1*R2*Vsc*s + C1*L1*R1*Vin*s^2 + C2*L1*R1*Vin*s^2 + C2*L1*R2*Vsc*s^2)/(R1 + L1*s + C1*L1*R1*s^2 + C2*L1*R1*s^2 + C2*L1*R2*s^2 + C2*R1*R2*s + C1*C2*L1*R1*R2*s^3)
v_3 == -(L1*R1*s^2*(C1*Vsc - C2*Vin + C2*Vsc + C1*C2*R2*Vsc*s))/(R1 + L1*s + C1*L1*R1*s^2 + C2*L1*R1*s^2 + C2*L1*R2*s^2 + C2*R1*R2*s + C1*C2*L1*R1*R2*s^3)
v_4 == (R1*Vsc + L1*Vsc*s + C2*R1*R2*Vsc*s + C2*L1*R1*Vin*s^2 + C2*L1*R2*Vsc*s^2)/(R1 + L1*s + C1*L1*R1*s^2 + C2*L1*R1*s^2 + C2*L1*R2*s^2 + C2*R1*R2*s + C1*C2*L1*R1*R2*s^3)
I_Vin == -(C2*s*(R1*Vin - R1*Vsc + L1*Vin*s - L1*Vsc*s + C1*L1*R1*Vin*s^2))/(R1 + L1*s + C1*L1*R1*s^2 + C2*L1*R1*s^2 + C2*L1*R2*s^2 + C2*R1*R2*s + C1*C2*L1*R1*R2*s^3)
I_Vsc == -(s*(R1 + L1*s)*(C1*Vsc - C2*Vin + C2*Vsc + C1*C2*R2*Vsc*s))/(R1 + L1*s + C1*L1*R1*s^2 + C2*L1*R1*s^2 + C2*L1*R2*s^2 + C2*R1*R2*s + C1*C2*L1*R1*R2*s^3)


Elapsed time is 0.544359 seconds.
>> eval(I_Vsc)

ans =

(Vin*s*(s/1000 + 1000))/(100000*(s^3/100000000 + (21*s^2)/1000000 + (10001*s)/1000 + 1000))

>> simplify(ans)

ans =

(Vin*s*(s + 1000000))/(s^3 + 2100*s^2 + 1000100000*s + 100000000000)

>> pretty(ans)
            Vin s (s + 1000000)
------------------------------------------
 3         2
s  + 2100 s  + 1000100000 s + 100000000000
```

Example 5:

Sometimes we would like to take SCAM results and use them in MATLAB® for further analysis. This is especially true when working with transfer functions. The example below shows how this can be accomplished. The netlist is called *example5.cir*.

Circuit



Netlist

```
Vin 1 0 Symbolic
R2 1 2 1000
C2 2 3 10E-6
C1 3 0 1E-6
L1 3 0 0.001
R1 3 0 1000
```

Now solve and get the transfer function symbolically

MATLAB

```
>> fname="example5.cir"

fname =

    "example5.cir"

>> ISU_scam

Started -- please be patient.

Netlist:
Vin 1 0 Symbolic
R2 1 2 1000
C2 2 3 10E-6
C1 3 0 1E-6
L1 3 0 0.001
R1 3 0 1000


The A matrix:
```

```
[ 1/R2,       -1/R2,                             0, 1]
[-1/R2, C2*s + 1/R2,                         -C2*s, 0]
[    0,       -C2*s, C1*s + C2*s + 1/R1 + 1/(L1*s), 0]
[    1,           0,                             0, 0]


The x vector:
  v_1
  v_2
  v_3
I_Vin


The z vector:
  0
  0
  0
Vin


The matrix equation:
                       I_Vin + v_1/R2 - v_2/R2 == 0
         v_2*(C2*s + 1/R2) - v_1/R2 - C2*s*v_3 == 0
v_3*(C1*s + C2*s + 1/R1 + 1/(L1*s)) - C2*s*v_2 == 0
                                     v_1 == Vin


The solution:

v_1 == Vin
v_2 == (Vin*(R1 + L1*s + C1*L1*R1*s^2 + C2*L1*R1*s^2))/(R1 + L1*s + C1*L1*R1*s^2 + C2*L1*R1*s^2 + C2*L1*R2*s^2 + C2*R1*R2*s + C1*C2*L1*R1*R2*s^3)
                             v_3 == (C2*L1*R1*Vin*s^2)/(R1 + L1*s + C1*L1*R1*s^2 + C2*L1*R1*s^2 + C2*L1*R2*s^2 + C2*R1*R2*s + C1*C2*L1*R1*R2*s^3)
       I_Vin == -(C2*Vin*s*(C1*L1*R1*s^2 + L1*s + R1))/(R1 + L1*s + C1*L1*R1*s^2 + C2*L1*R1*s^2 + C2*L1*R2*s^2 + C2*R1*R2*s + C1*C2*L1*R1*R2*s^3)


Elapsed time is 0.293142 seconds.
>> H=v_2/Vin

H =

(R1 + L1*s + C1*L1*R1*s^2 + C2*L1*R1*s^2)/(R1 + L1*s + C1*L1*R1*s^2 + C2*L1*R1*s^2 + C2*L1*R2*s^2 + C2*R1*R2*s + C1*C2*L1*R1*R2*s^3)

>> H=collect(H)
```

```
H =

((C1*L1*R1 + C2*L1*R1)*s^2 + L1*s + R1)/(C1*C2*L1*R1*R2*s^3 + (C1*L1*R1
+ C2*L1*R1 + C2*L1*R2)*s^2 + (L1 + C2*R1*R2)*s + R1)

>> pretty(H)
                                                   2
                      (C1 L1 R1 + C2 L1 R1) s  + L1 s + R1
----------------------------------------------------------------------
----------
                3                                      2
C1 C2 L1 R1 R2 s  + (C1 L1 R1 + C2 L1 R1 + C2 L1 R2) s  + (L1 + C2 R1
R2) s + R1
```

We can also get a numerical transfer function:

```
>> Hnumbers=eval(H)

Hnumbers =

((6493253913945763*s^2)/590295810358705651712      +      s/1000      +
1000)/(s^3/100000000  +  (3099053004383205*s^2)/147573952589676412928  +
(10001*s)/1000 + 1000)
```

While this answer is correct, it is not in a very convenient form, and we can't do any actual simulation with it. However, we can easily convert the expression to a MATLAB® transfer function object. First, we separate the numerator and denominator and then convert them to MATLAB® polynomials:

```
>> [n,d]=numden(Hnumbers)

n =

2536427310135063671875*s^2      +      230584300921369395200000*s      +
230584300921369395200000000000


d =

2305843009213693952*s^3       +       4842270319348757812500*s^2       +
2306073593514615321395200000*s + 230584300921369395200000000000

>> mySys=tf(sym2poly(n),sym2poly(d))

mySys =

         2.536e21 s^2 + 2.306e23 s + 2.306e29
  ---------------------------------------------------
  2.306e18 s^3 + 4.842e21 s^2 + 2.306e27 s + 2.306e29
```

```
Continuous-time transfer function.
```

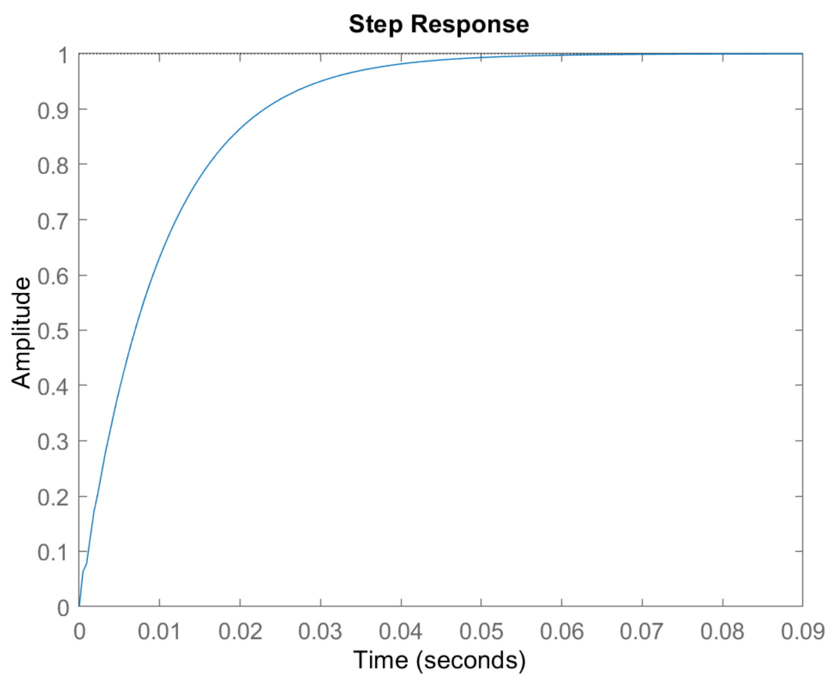
The transfer function is still in an odd form (we'd like the highest power of 's' in the denominator to have a coefficient of 1). Fortunately, the MATLAB® "`minreal()`" function will do this normalization for us.

```
>> mySys=minreal(mySys)

mySys =

    1100 s^2 + 100000 s + 1e11
  ------------------------------
  s^3 + 2100 s^2 + 1e09 s + 1e11

Continuous-time transfer function.
```

We are now free to perform any further MATLAB® functions relating to polynomials. Shown below are the unit-step response and a Bode plot for this example.

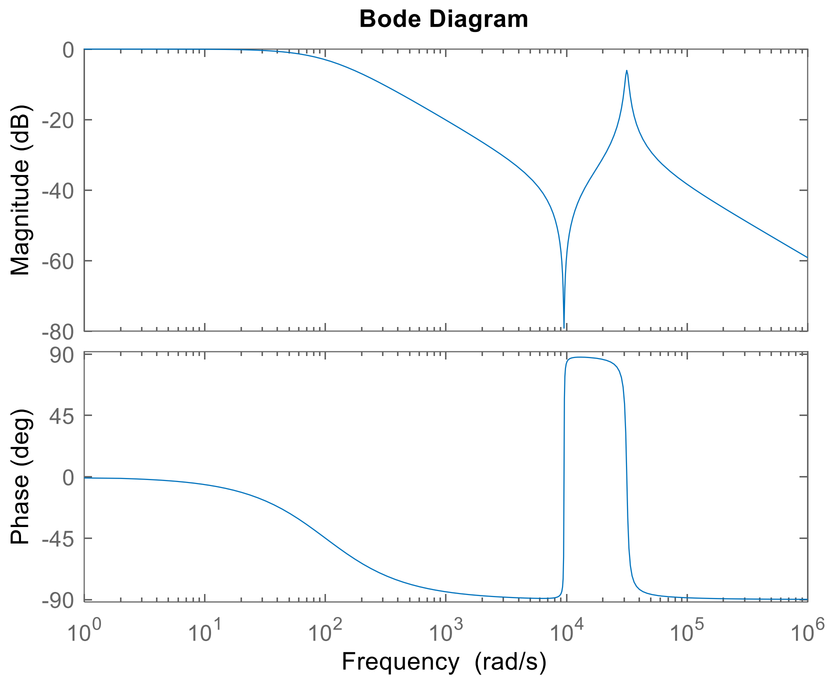
For the Step Response:

```
>> step(mySyS)
```



For a Bode Plot:

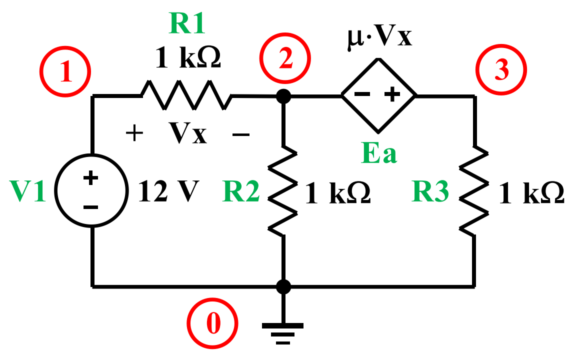
```
>> bode(mySyS)
```



Example 6:

Now consider the following circuit that contains a VCVS:

Circuit



Netlist

```
V1 1 0 12
R1 1 2 1000
R2 2 0 1000
R3 3 0 1000
Ea 3 2 1 2 mu
```

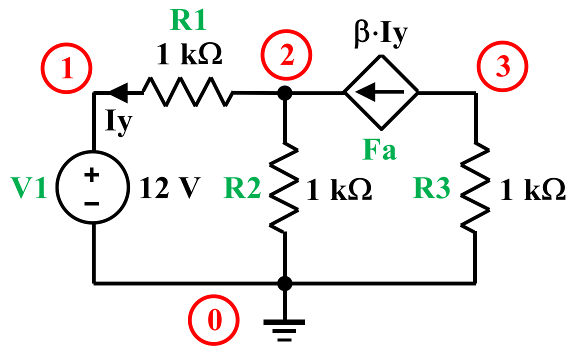
MATLAB

```
>> fname="example6.cir";
>> ISU_scam

Started -- please be patient.

Netlist:
V1 1 0 12
R1 1 2 1000
R2 2 0 1000
R3 3 0 1000
Ea 3 2 1 2 mu

The A matrix:
[ 1/R1,       -1/R1,     0, 1,  0]
[-1/R1, 1/R1 + 1/R2,     0, 0, -1]
[    0,           0, 1/R3, 0,  1]
[    1,           0,     0, 0,  0]
[  -Ea,      Ea - 1,     1, 0,  0]


The x vector:
 v_1
```

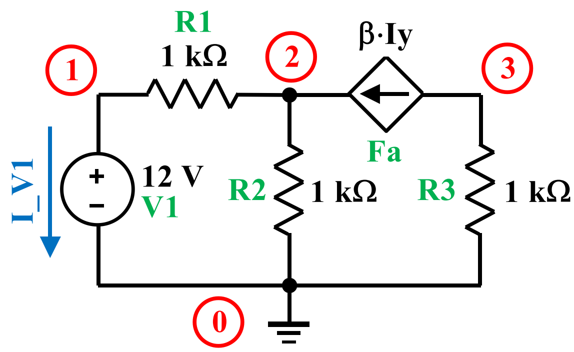
```
 v_2
 v_3
I_V1
I_Ea


The z vector:
 0
 0
 0
V1
 0


The matrix equation:
           I_V1 + v_1/R1 - v_2/R1 == 0
v_2*(1/R1 + 1/R2) - v_1/R1 - I_Ea == 0
                    I_Ea + v_3/R3 == 0
                             v_1 == V1
      v_3 - Ea*v_1 + v_2*(Ea - 1) == 0


The solution:
                                                     v_1 == V1
v_2 == (R2*V1*(R3 - Ea*R1))/(R1*R2 + R1*R3 + R2*R3 - Ea*R1*R2)
v_3 == (R3*V1*(R2 + Ea*R1))/(R1*R2 + R1*R3 + R2*R3 - Ea*R1*R2)
    I_V1 == -(V1*(R2 + R3))/(R1*R2 + R1*R3 + R2*R3 - Ea*R1*R2)
 I_Ea == -(V1*(R2 + Ea*R1))/(R1*R2 + R1*R3 + R2*R3 - Ea*R1*R2)


Elapsed time is 1.6941 seconds.
```

Example 7:

Now consider the following circuit that contains a CCCS:



Circuit



Netlist

```
V1 1 0 12
R1 1 2 1000
R2 2 0 1000
R3 3 0 1000
Fa 3 2 V1 beta
```

MATLAB

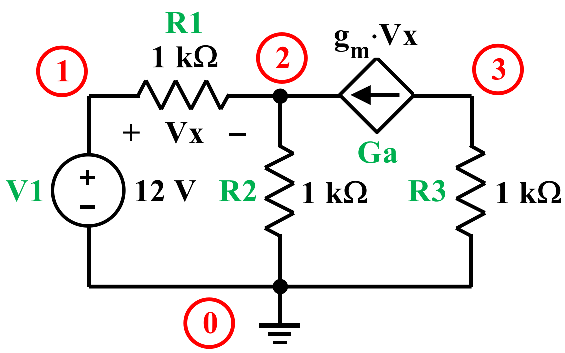
```
>> fname="example7.cir";
>> ISU_scam

Started -- please be patient.

Netlist:
V1 1 0 12
R1 1 2 1000
R2 2 0 1000
R3 3 0 1000
Fa 3 2 V1 beta
```

```
The A matrix:
[ 1/R1,       -1/R1,    0,   1]
[-1/R1, 1/R1 + 1/R2,    0, -Fa]
[    0,           0, 1/R3,  Fa]
[    1,           0,    0,   0]


The x vector:
 v_1
 v_2
 v_3
I_V1


The z vector:
 0
 0
 0
V1


The matrix equation:
              I_V1 + v_1/R1 - v_2/R1 == 0
v_2*(1/R1 + 1/R2) - v_1/R1 - Fa*I_V1 == 0
                    v_3/R3 + Fa*I_V1 == 0
                                v_1 == V1


The solution:
                                 v_1 == V1
v_2 == -(R2*V1*(Fa - 1))/(R1 + R2 - Fa*R2)
       v_3 == (Fa*R3*V1)/(R1 + R2 - Fa*R2)
             I_V1 == -V1/(R1 + R2 - Fa*R2)


Elapsed time is 0.251362 seconds.
```

Example 8:

Now consider the following circuit that contains a VCCS:

Circuit



Netlist

```
V1 1 0 12
R1 1 2 1000
R2 2 0 1000
R3 3 0 1000
Ga 3 2 1 2 gm
```

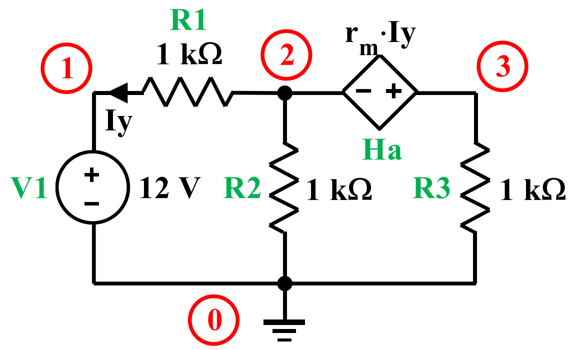
MATLAB

```
>> fname="example8.cir";
>> ISU_scam

Started -- please be patient.

Netlist:
V1 1 0 12
R1 1 2 1000
R2 2 0 1000
R3 3 0 1000
Ga 3 2 1 2 g

The A matrix:
[        1/R1,              -1/R1,    0, 1]
[- Ga - 1/R1, Ga + 1/R1 + 1/R2,    0, 0]
[          Ga,                -Ga, 1/R3, 0]
[           1,                  0,    0, 0]


The x vector:
 v_1
 v_2
```

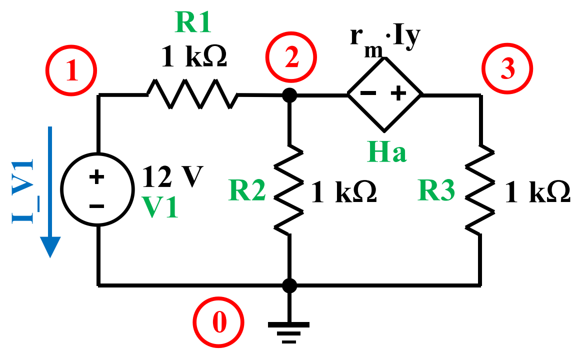
```
 v_3
I_V1


The z vector:
 0
 0
 0
V1


The matrix equation:
                  I_V1 + v_1/R1 - v_2/R1 == 0
v_2*(Ga + 1/R1 + 1/R2) - v_1*(Ga + 1/R1) == 0
                Ga*v_1 - Ga*v_2 + v_3/R3 == 0
                                  v_1 == V1


The solution:
                                       v_1 == V1
v_2 == (R2*V1*(Ga*R1 + 1))/(R1 + R2 + Ga*R1*R2)
     v_3 == -(Ga*R1*R3*V1)/(R1 + R2 + Ga*R1*R2)
               I_V1 == -V1/(R1 + R2 + Ga*R1*R2)


Elapsed time is 0.20817 seconds.
```

Example 9:

Now consider the following circuit that contains a CCVS:



Circuit



Netlist

```
V1 1 0 12
R1 1 2 1000
R2 2 0 1000
R3 3 0 1000
Ha 3 2 V1 r
```

MATLAB

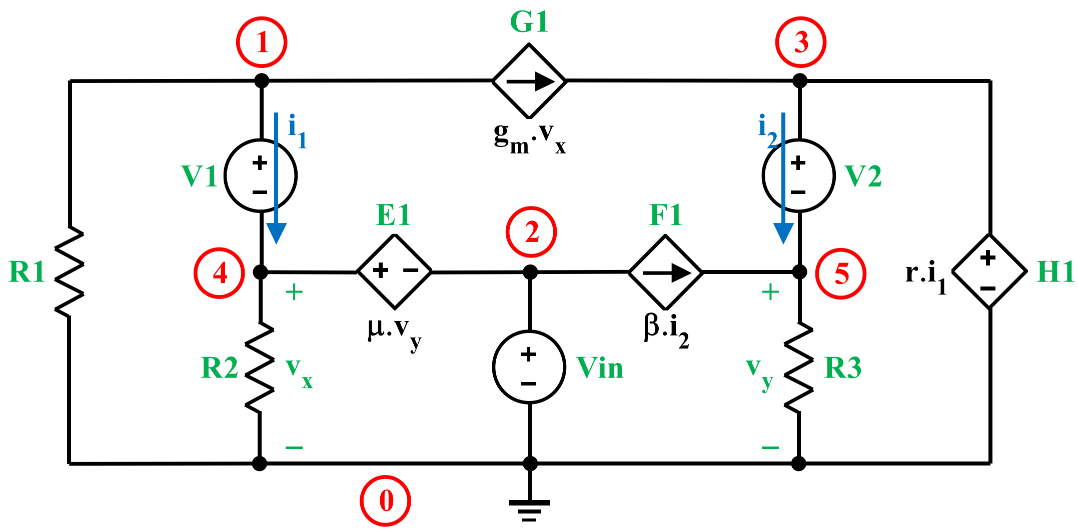
```
>> fname="example9.cir";
>> ISU_scam

Started -- please be patient.

Netlist:
V1 1 0 12
R1 1 2 1000
R2 2 0 1000
R3 3 0 1000
```

```
Ha 3 2 V1 r

The A matrix:
[ 1/R1,       -1/R1,    0,   1,  0]
[-1/R1, 1/R1 + 1/R2,    0,   0, -1]
[    0,           0, 1/R3,   0,  1]
[    1,           0,    0,   0,  0]
[    0,          -1,    1, -Ha,  0]


The x vector:
 v_1
 v_2
 v_3
I_V1
I_Ha


The z vector:
 0
 0
 0
V1
 0


The matrix equation:
           I_V1 + v_1/R1 - v_2/R1 == 0
v_2*(1/R1 + 1/R2) - v_1/R1 - I_Ha == 0
                    I_Ha + v_3/R3 == 0
                             v_1 == V1
             v_3 - v_2 - Ha*I_V1 == 0


The solution:
                                                v_1 == V1
 v_2 == (R2*V1*(Ha + R3))/(R1*R2 + R1*R3 + R2*R3 + Ha*R2)
v_3 == -(R3*V1*(Ha - R2))/(R1*R2 + R1*R3 + R2*R3 + Ha*R2)
  I_V1 == -(V1*(R2 + R3))/(R1*R2 + R1*R3 + R2*R3 + Ha*R2)
   I_Ha == (V1*(Ha - R2))/(R1*R2 + R1*R3 + R2*R3 + Ha*R2)


Elapsed time is 0.273394 seconds.
```

Example 10:

Now consider the circuit shown below. It includes all four types of controlled source.

Circuit



Netlist

```
R3 1 0 Symbolic
R1 2 0 Symbolic
R2 3 0 Symbolic
G1 1 4 2 0 Symbolic
F1 5 3 V2 Symbolic
E1 2 5 3 0 Symbolic
V2 4 3 0
V1 1 2 0
H1 4 0 V1 Symbolic
Vin 5 0 Symbolic
```

MATLAB

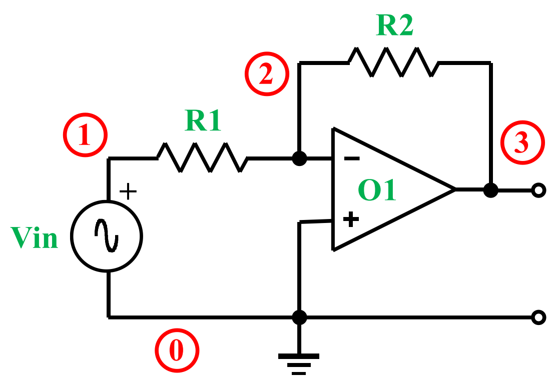
```
>> fname="example10.cir";
>> ISU_scam

Started -- please be patient.
```

Note: If a value is not given for an element, it can be declared symbolic (as above). However, this declaration is not really necessary – the user must simply ensure that the value is not a number. So, for example, instead of "Symbolic", the value of R1 could have been "zyx" or even left blank.

Example 11:

The next example shows an OpAmp with 2 resistors in the standard inverting configuration along with its netlist (*example11.cir)*.

Circuit



Netlist

```
Vin 1 0 Symbolic
R1 1 2 Symbolic
R2 2 3 Symbolic
O1 0 2 3
```

Since we don't have values for the components in the circuit, they are declared to be "Symbolic". We can now solve this circuit to determine the gain between Vin and node 3.

MATLAB

```
>> fname="example11.cir"
>> ISU_scam

Started -- please be patient.

Netlist:
Vin 1 0 Symbolic
R1 1 2 Symbolic
R2 2 3 Symbolic
O1 0 2 3

The A matrix:
[ 1/R1,       -1/R1,     0, 1, 0]
[-1/R1, 1/R1 + 1/R2, -1/R2, 0, 0]
[    0,       -1/R2,  1/R2, 0, 1]
[    1,           0,     0, 0, 0]
[    0,          -1,     0, 0, 0]
```

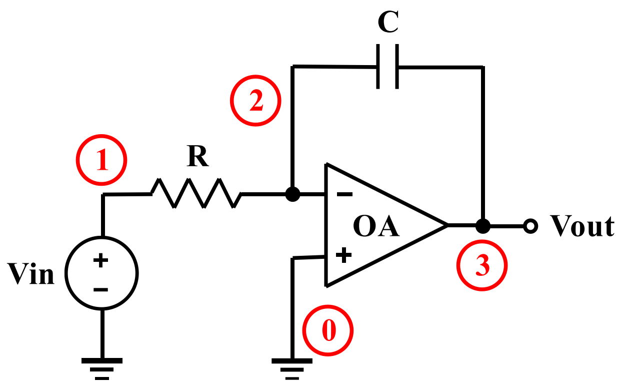
```
The x vector:
  v_1
  v_2
  v_3
I_Vin
 I_O1


The z vector:
  0
  0
  0
Vin
  0


The matrix equation:
            I_Vin + v_1/R1 - v_2/R1 == 0
v_2*(1/R1 + 1/R2) - v_3/R2 - v_1/R1 == 0
             I_O1 - v_2/R2 + v_3/R2 == 0
                             v_1 == Vin
                              -v_2 == 0


The solution:
         v_1 == Vin
           v_2 == 0
v_3 == -(R2*Vin)/R1
   I_Vin == -Vin/R1
     I_O1 == Vin/R1


Elapsed time is 0.202455 seconds.
>> v_3/Vin

ans =

-R2/R1
```

Note: Because of the way ideal OpAmps are handled in MNA, the circuit above would give exactly the same results even if the input terminals were switched. In practice this would not work; it is the responsibility of the designer to make sure negative feedback exists.

Example 12:

Now consider the circuit shown below. It is easily recognized as a simple integrator. Let's see if SCAM will verify this.

Circuit



Netlist

```
Vin 1 0
R 1 2
C 2 3
OA 0 2 3
```

MATLAB

```
>> fname="example12.cir";
>> ISU_scam

Started -- please be patient.

Netlist:
Vin 1 0
R 1 2
C 2 3
OA 0 2 3

The A matrix:
[ 1/R,      -1/R,    0, 1, 0]
[-1/R, C*s + 1/R, -C*s, 0, 0]
[   0,      -C*s,  C*s, 0, 1]
[   1,         0,    0, 0, 0]
[   0,        -1,    0, 0, 0]


The x vector:
  v_1
  v_2
  v_3
I_Vin
```

```
 I_OA


The z vector:
  0
  0
  0
Vin
  0


The matrix equation:
            I_Vin + v_1/R - v_2/R == 0
v_2*(C*s + 1/R) - v_1/R - C*s*v_3 == 0
         I_OA - C*s*v_2 + C*s*v_3 == 0
                         v_1 == Vin
                          -v_2 == 0


The solution:
         v_1 == Vin
           v_2 == 0
v_3 == -Vin/(C*R*s)
    I_Vin == -Vin/R
      I_OA == Vin/R


Elapsed time is 0.243157 seconds.
>> simplify(v_3)

ans =

-Vin/(C*R*s)
```

Hence,

$$Vout = -\frac{Vin}{CRs}$$

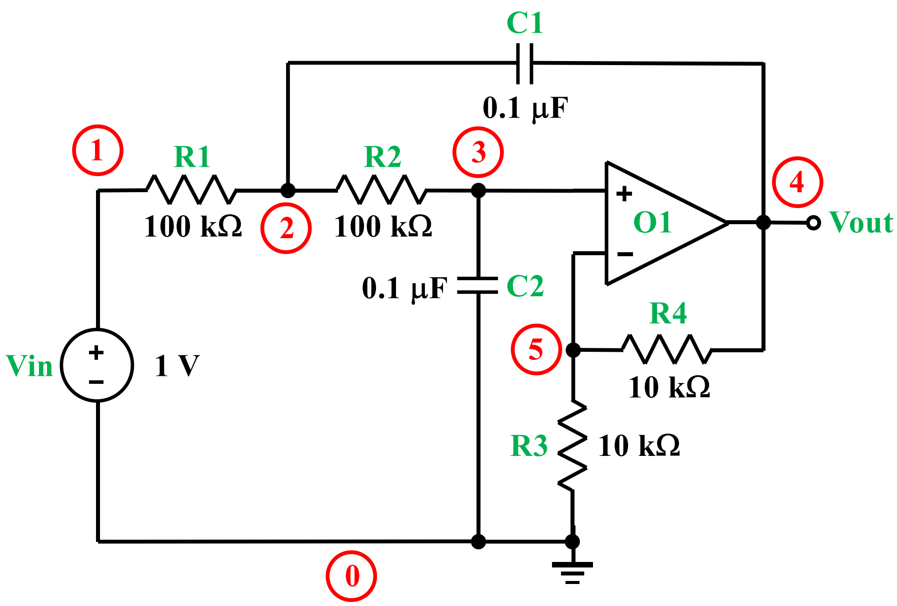
Or, in the time domain,

$$v_{out}(t) = -\frac{1}{RC}\int v_{in}(t)\,dt$$

Example 13:

Now consider the circuit shown below. This is a two-pole, Sallen-Key, Butterworth lowpass filter. Let's use SCAM to investigate its frequency response.

Circuit



Netlist

```
Vin 1 0 1
R1 1 2 100E3
R2 2 3 100E3
C2 3 0 0.1E-6
R3 5 0 10E3
R4 5 4 10E3
C1 2 4 0.1E-6
O1 3 5 4
```

MATLAB

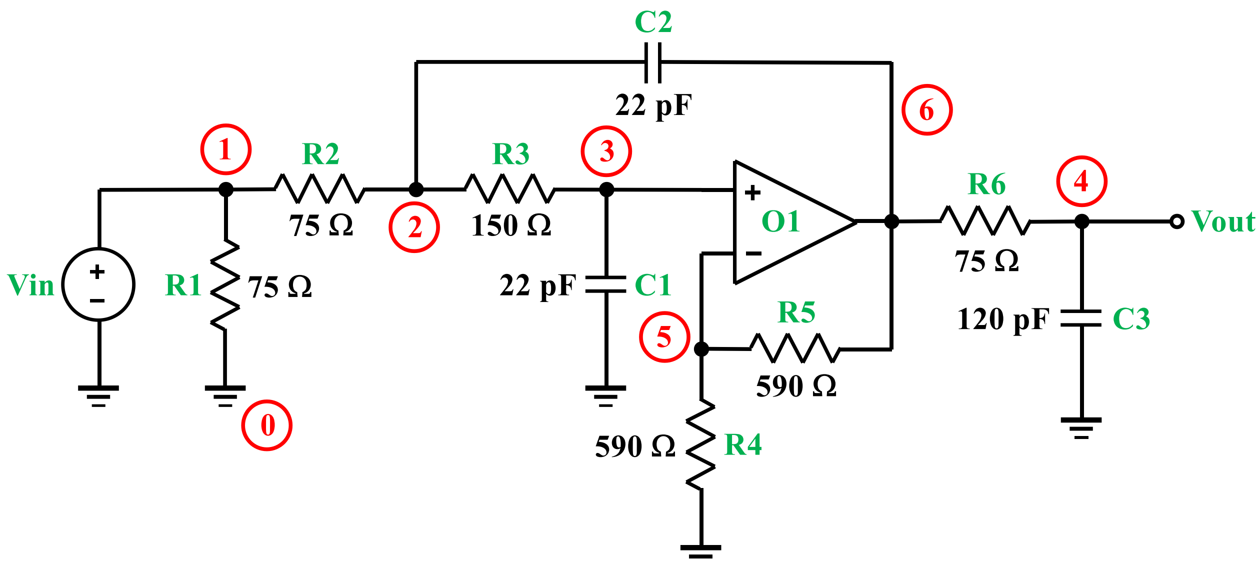
```
>> fname="example13.cir";
>> ISU_scam

Started -- please be patient.
```

$$H = \frac{20000}{s^2 + 100s + 10000}$$

Example 14:

Now consider the circuit shown below. This is a three-pole, Sallen-Key, Butterworth lowpass filter. Let's use SCAM to investigate its frequency response.

Circuit



Netlist

```
Vin 1 0
R1 1 0 75
R2 1 2 75
R3 2 3 150
C1 3 0 22E-12
O1 3 5 6
C2 2 6 22E-12
R4 5 0 590
R5 6 5 590
R6 6 4 75
C3 4 0 120E-12
```

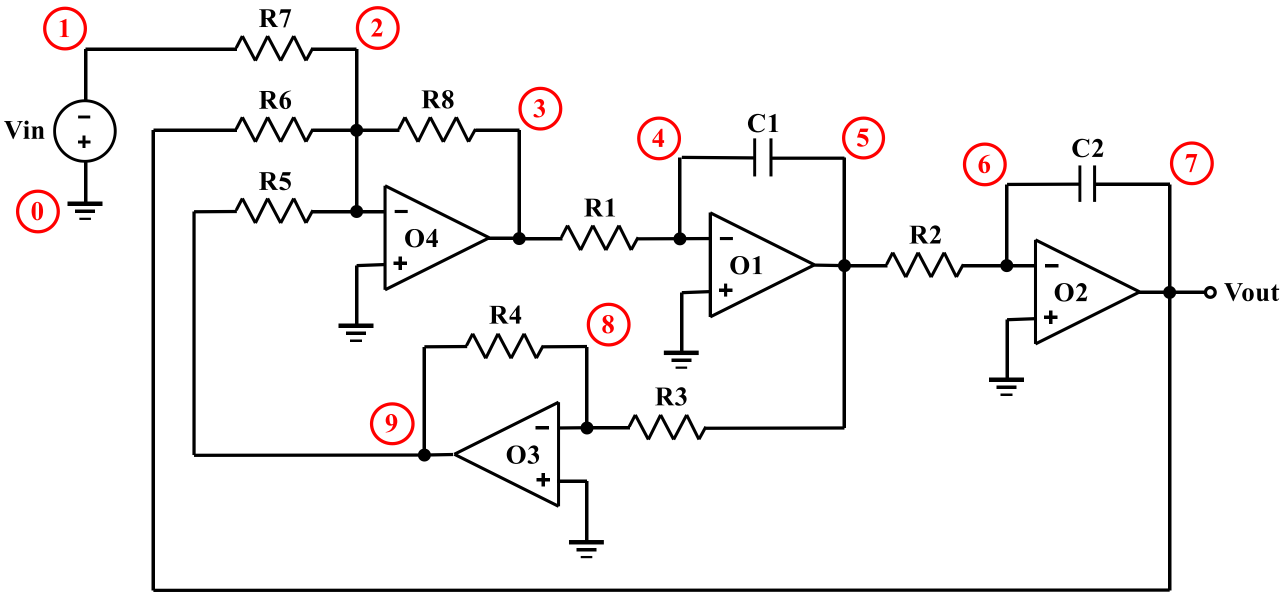
MATLAB

```
>> fname="example14.cir";
>> ISU_scam

Started -- please be patient.
```

Example 15:

Now consider the circuit shown below. This is an analog computer intended to solve the second-order ODE: $\ddot{x}+3\dot{x}+2x=f(t)$. Let's see if we can verify that with SCAM.

Circuit



Netlist

```
Vin 0 1
R7 1 2 60E3
R6 7 2 30E3
R5 9 2 20E3
R8 2 3 60E3
O4 0 2 3
R1 3 4 10E3
C1 4 5 100E-6
O1 0 4 5
R2 5 6 10E3
C2 6 7 100E-6
O2 0 6 7
R3 5 8 10E3
R4 8 9 10E3
O3 0 8 9
```

MATLAB

```
>> fname="example15.cir";
>> ISU_scam

Started -- please be patient.
```

```
Netlist:
Vin 0 1
R7 1 2 60E3
R6 7 2 30E3
R5 9 2 20E3
R8 2 3 60E3
O4 0 2 3
R1 3 4 10E3
C1 4 5 100E-6
O1 0 4 5
R2 5 6 10E3
C2 6 7 100E-6
O2 0 6 7
R3 5 8 10E3
R4 8 9 10E3
O3 0 8 9

The A matrix:
[ 1/R7,                          -1/R7,            0,              0,
0,            0,            0,            0,            0, -1, 0, 0, 0, 0]
[-1/R7, 1/R5 + 1/R6 + 1/R7 + 1/R8,          -1/R8,                 0,
0,            0,        -1/R6,            0,        -1/R5,  0, 0, 0, 0, 0]
[     0,                             -1/R8, 1/R1 + 1/R8,          -1/R1,
0,            0,            0,            0,            0,  0, 1, 0, 0, 0]
[    0,                         0,       -1/R1, C1*s + 1/R1,            -
C1*s,            0,            0,            0,            0,  0, 0, 0, 0,
0]
[    0,                          0,             0,         -C1*s, C1*s +
1/R2 + 1/R3,       -1/R2,            0,       -1/R3,            0,  0, 0,
1, 0, 0]
[    0,                         0,            0,            0,            -
1/R2, C2*s + 1/R2,       -C2*s,            0,            0,  0, 0, 0, 0,
0]
[     0,                          -1/R6,            0,              0,
0,       -C2*s, C2*s + 1/R6,            0,            0,  0, 0, 0, 1, 0]
[    0,                         0,            0,            0,            -
1/R3,            0,            0, 1/R3 + 1/R4,       -1/R4,  0, 0, 0, 0,
0]
[     0,                          -1/R5,            0,              0,
0,            0,            0,       -1/R4, 1/R4 + 1/R5,  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,  0, 0, 0, 0, 0]
[     0,                             0,             0,            -1,
0,            0,            0,            0,            0,  0, 0, 0, 0, 0]
[     0,                             0,             0,              0,
0,           -1,            0,            0,            0,  0, 0, 0, 0, 0]
[     0,                             0,             0,              0,
0,            0,            0,           -1,            0,  0, 0, 0, 0, 0]
```

```
The x vector:
  v_1
  v_2
  v_3
  v_4
  v_5
  v_6
  v_7
  v_8
  v_9
I_Vin
 I_O4
 I_O1
 I_O2
 I_O3


The z vector:
  0
  0
  0
  0
  0
  0
  0
  0
  0
Vin
  0
  0
  0
  0


The matrix equation:
                                                    v_1/R7 - I_Vin - v_2/R7 ==
0
v_2*(1/R5 + 1/R6 + 1/R7 + 1/R8) - v_3/R8 - v_7/R6 - v_9/R5 - v_1/R7 ==
0
                         I_O4 - v_4/R1 - v_2/R8 + v_3*(1/R1 + 1/R8) ==
0
                              v_4*(C1*s + 1/R1) - v_3/R1 - C1*s*v_5 ==
0
       I_O1 - v_6/R2 - v_8/R3 + v_5*(C1*s + 1/R2 + 1/R3) - C1*s*v_4 ==
0
                              v_6*(C2*s + 1/R2) - v_5/R2 - C2*s*v_7 ==
0
                       I_O2 - v_2/R6 + v_7*(C2*s + 1/R6) - C2*s*v_6 ==
0
```

```
v_8*(1/R3 + 1/R4) - v_9/R4 - v_5/R3 == 0
I_O3 - v_2/R5 - v_8/R4 + v_9*(1/R4 + 1/R5) == 0
-v_1 == Vin
-v_2 == 0
-v_4 == 0
-v_6 == 0
-v_8 == 0
```

```
The solution:

v_1 == -Vin

v_2 == 0
v_3 == (C1*C2*R1*R2*R3*R5*R6*R8*Vin*s^2)/(R7*(C1*C2*R1*R2*R3*R5*R6*s^2 + C2*R2*R4*R6*R8*s + R3*R5*R8))

v_4 == 0
v_5 == -(C2*R2*R3*R5*R6*R8*Vin*s)/(R7*(C1*C2*R1*R2*R3*R5*R6*s^2 + C2*R2*R4*R6*R8*s + R3*R5*R8))

v_6 == 0
v_7 == (R3*R5*R6*R8*Vin)/(R7*(C1*C2*R1*R2*R3*R5*R6*s^2 + C2*R2*R4*R6*R8*s + R3*R5*R8))

v_8 == 0
v_9 == (C2*R2*R4*R5*R6*R8*Vin*s)/(R7*(C1*C2*R1*R2*R3*R5*R6*s^2 + C2*R2*R4*R6*R8*s + R3*R5*R8))

I_Vin == -Vin/R7
I_O4 == -(C1*C2*R2*R3*R5*R6*Vin*s^2*(R1 + R8))/(R7*(C1*C2*R1*R2*R3*R5*R6*s^2 + C2*R2*R4*R6*R8*s + R3*R5*R8))
I_O1 == (C2*R5*R6*R8*Vin*s*(R2 + R3 + C1*R2*R3*s))/(R7*(C1*C2*R1*R2*R3*R5*R6*s^2 + C2*R2*R4*R6*R8*s + R3*R5*R8))
I_O2 == -(R3*R5*R8*Vin*(C2*R6*s + 1))/(R7*(C1*C2*R1*R2*R3*R5*R6*s^2 + C2*R2*R4*R6*R8*s + R3*R5*R8))
I_O3 == -(C2*R2*R6*R8*Vin*s*(R4 + R5))/(R7*(C1*C2*R1*R2*R3*R5*R6*s^2 + C2*R2*R4*R6*R8*s + R3*R5*R8))
```

```
Elapsed time is 1.02594 seconds.
>> eval(v_7)

ans =

(36000000000000000*Vin)/(36000000000000000*s^2                    +
108000000000000000*s + 72000000000000000)

>> simplify(ans)

ans =

Vin/(s^2 + 3*s + 2)
```

Hence,

$$Vout = \frac{Vin}{s^2 + 3s + 2},$$

or, in the time domain

$$\ddot{V}out + 3\dot{V}out + 2Vout = Vin,$$

In a more-recognizable form

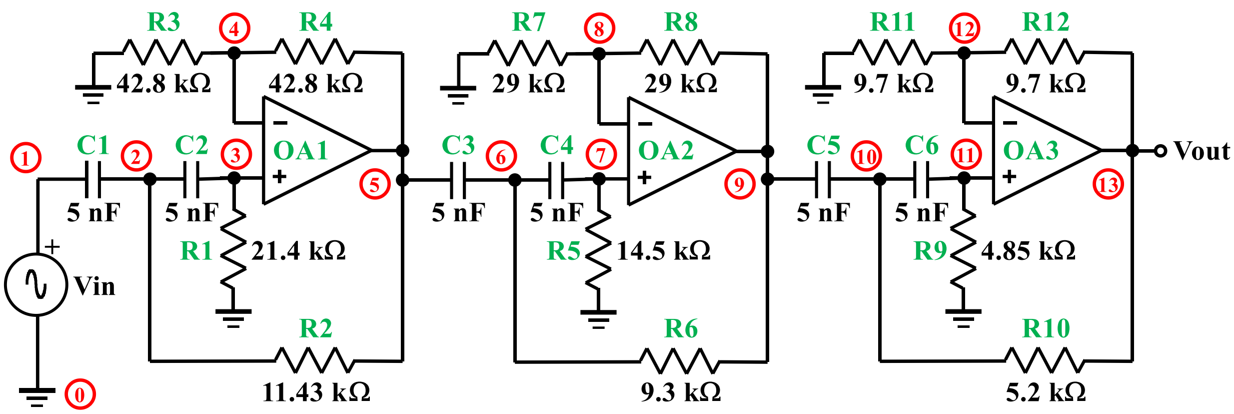
$$\ddot{x} + 3\dot{x} + 2x = f(t)$$

thus verifying the function of the circuit.

Example 16:

The circuit shown below is a 6^{th}-Order Chebyshev 2dB RW High Pass Filter.

Circuit



Netlist

```
Vin 1 0
C1 1 2 5E-9
C2 2 3 5E-9
R1 3 0 21.4E3
R2 2 5 11.43E3
R3 4 0 42.8E3
R4 4 5 42.8E3
OA1 3 4 5
C3 5 6 5E-9
C4 6 7 5E-9
R5 7 0 14.5E3
R6 6 9 9.3E3
R7 8 0 29E3
R8 8 9 29E3
OA2 7 8 9
C5 9 10 5E-9
C6 10 11 5E-9
R9 11 0 4.85E3
R10 10 13 5.2E3
R11 12 0 9.7E3
R12 12 13 9.7E3
OA3 11 12 13
```

MATLAB

```
>> fname="example16.cir"
```

```
>> ISU_scam

Started -- please be patient.

Netlist:
      Vin 1 0
      C1 1 2 5E-9
      C2 2 3 5E-9
      R1 3 0 21.4E3
      R2 2 5 11.43E3
      R3 4 0 42.8E3
      R4 4 5 42.8E3
      OA1 3 4 5
      C3 5 6 5E-9
      C4 6 7 5E-9
      R5 7 0 14.5E3
      R6 6 9 9.3E3
      R7 8 0 29E3
      R8 8 9 29E3
      OA2 7 8 9
      C5 9 10 5E-9
      C6 10 11 5E-9
      R9 11 0 4.85E3
      R10 10 13 5.2E3
      R11 12 0 9.7E3
      R12 12 13 9.7E3
      OA3 11 12 13


The A matrix:
[ C1*s,            -C1*s,          0,           0,                  0,
0,            0,            0,                    0,                  0,
0,              0,              0, 1, 0, 0, 0]
[-C1*s, C1*s + C2*s + 1/R2,      -C2*s,            0,                  -
1/R2,                   0,            0,            0,                  0,
0,            0,              0,              0, 0, 0, 0, 0]
[    0,           -C2*s, C2*s + 1/R1,          0,                  0,
0,            0,            0,                    0,                  0,
0,              0,              0, 0, 0, 0, 0]
[    0,                0,            0, 1/R3 + 1/R4,               -
1/R4,                   0,            0,            0,                  0,
0,            0,              0,              0, 0, 0, 0, 0]
[    0,            -1/R2,            0,       -1/R4, C3*s + 1/R2 +
1/R4,              -C3*s,            0,            0,                  0,
0,            0,              0,              0, 0, 1, 0, 0]
```

```
  [    0,                   0,            0,            0,                 -
C3*s, C3*s + C4*s + 1/R6,       -C4*s,            0,                 -1/R6,
0,            0,              0,              0, 0, 0, 0, 0]
  [    0,                  0,            0,            0,                 0,
-C4*s, C4*s + 1/R5,          0,                  0,                     0,
0,              0,              0, 0, 0, 0, 0]
  [    0,                  0,            0,            0,                 0,
0,            0, 1/R7 + 1/R8,              -1/R8,                    0,
0,              0,              0, 0, 0, 0, 0]
  [    0,                  0,            0,            0,                 0,
-1/R6,             0,        -1/R8, C5*s + 1/R6 + 1/R8,                 -
C5*s,             0,              0,              0, 0, 0, 1, 0]
  [    0,                  0,            0,            0,                 0,
0,            0,             0,              -C5*s, C5*s + C6*s + 1/R10,
-C6*s,                0,         -1/R10, 0, 0, 0, 0]
  [    0,                  0,            0,            0,                 0,
0,            0,             0,                   0,                 -C6*s,
C6*s + 1/R9,             0,              0, 0, 0, 0, 0]
  [    0,                  0,            0,            0,                 0,
0,            0,             0,                   0,                     0,
0, 1/R11 + 1/R12,        -1/R12, 0, 0, 0, 0]
  [    0,                  0,            0,            0,                 0,
0,            0,             0,                   0,                -1/R10,
0,        -1/R12, 1/R10 + 1/R12, 0, 0, 0, 1]
  [    1,                  0,            0,            0,                 0,
0,            0,             0,                   0,                     0,
0,              0,              0, 0, 0, 0, 0]
  [    0,                  0,            1,           -1,                 0,
0,            0,             0,                   0,                     0,
0,              0,              0, 0, 0, 0, 0]
  [    0,                  0,            0,            0,                 0,
0,            1,            -1,                   0,                     0,
0,              0,              0, 0, 0, 0, 0]
  [    0,                  0,            0,            0,                 0,
0,            0,             0,                   0,                     0,
1,             -1,              0, 0, 0, 0, 0]
```

```
The x vector:
  v_1
  v_2
  v_3
  v_4
  v_5
  v_6
  v_7
  v_8
  v_9
 v_10
 v_11
 v_12
 v_13
I_Vin
I_OA1
I_OA2
I_OA3
```

```
The z vector:
  0
  0
  0
  0
  0
  0
  0
  0
  0
  0
  0
  0
  0
Vin
  0
  0
  0
```

The matrix equation:

```
I_Vin + C1*s*v_1 - C1*s*v_2 == 0
v_2*(C1*s + C2*s + 1/R2) - v_5/R2 - C1*s*v_1 - C2*s*v_3 == 0
v_3*(C2*s + 1/R1) - C2*s*v_2 == 0
v_4*(1/R3 + 1/R4) - v_5/R4 == 0
I_OA1 - v_2/R2 - v_4/R4 + v_5*(C3*s + 1/R2 + 1/R4) - C3*s*v_6 == 0
v_6*(C3*s + C4*s + 1/R6) - v_9/R6 - C3*s*v_5 - C4*s*v_7 == 0
v_7*(C4*s + 1/R5) - C4*s*v_6 == 0
v_8*(1/R7 + 1/R8) - v_9/R8 == 0
I_OA2 - v_6/R6 - v_8/R8 + v_9*(C5*s + 1/R6 + 1/R8) - C5*s*v_10 == 0
v_10*(C5*s + C6*s + 1/R10) - v_13/R10 - C5*s*v_9 - C6*s*v_11 == 0
v_11*(C6*s + 1/R9) - C6*s*v_10 == 0
v_12*(1/R11 + 1/R12) - v_13/R12 == 0
I_OA3 - v_10/R10 - v_12/R12 + v_13*(1/R10 + 1/R12) == 0
v_1 == Vin
v_3 - v_4 == 0
v_7 - v_8 == 0
v_11 - v_12 == 0
```

The solution:

```
v_1 == Vin

v_2 == (C1*R2*R3*Vin*s*(C2*R1*s + 1))/(R3 + C1*R2*R3*s - C2*R1*R4*s + C2*R2*R3*s + C1*C2*R1*R2*R3*s^2)

v_3 == (C1*C2*R1*R2*R3*Vin*s^2)/(R3 + C1*R2*R3*s - C2*R1*R4*s + C2*R2*R3*s + C1*C2*R1*R2*R3*s^2)

v_4 == (C1*C2*R1*R2*R3*Vin*s^2)/(R3 + C1*R2*R3*s - C2*R1*R4*s + C2*R2*R3*s + C1*C2*R1*R2*R3*s^2)

v_5 == (C1*C2*R1*R2*Vin*s^2*(R3 + R4))/(R3 + C1*R2*R3*s - C2*R1*R4*s + C2*R2*R3*s + C1*C2*R1*R2*R3*s^2)
```

```
v_6 == (C1*C2*C3*R1*R2*R6*R7*Vin*s^3*(R3 + R4)*(C4*R5*s + 1))/((R3 +
C1*R2*R3*s - C2*R1*R4*s + C2*R2*R3*s + C1*C2*R1*R2*R3*s^2)*(R7 +
C3*R6*R7*s - C4*R5*R8*s + C4*R6*R7*s + C3*C4*R5*R6*R7*s^2))

v_7 == (C1*C2*C3*C4*R1*R2*R5*R6*R7*Vin*s^4*(R3 + R4))/((R3 + C1*R2*R3*s
- C2*R1*R4*s + C2*R2*R3*s + C1*C2*R1*R2*R3*s^2)*(R7 + C3*R6*R7*s -
C4*R5*R8*s + C4*R6*R7*s + C3*C4*R5*R6*R7*s^2))

v_8 == (C1*C2*C3*C4*R1*R2*R5*R6*R7*Vin*s^4*(R3 + R4))/((R3 + C1*R2*R3*s
- C2*R1*R4*s + C2*R2*R3*s + C1*C2*R1*R2*R3*s^2)*(R7 + C3*R6*R7*s -
C4*R5*R8*s + C4*R6*R7*s + C3*C4*R5*R6*R7*s^2))

v_9 == (C1*C2*C3*C4*R1*R2*R5*R6*Vin*s^4*(R3 + R4)*(R7 + R8))/((R3 +
C1*R2*R3*s - C2*R1*R4*s + C2*R2*R3*s + C1*C2*R1*R2*R3*s^2)*(R7 +
C3*R6*R7*s - C4*R5*R8*s + C4*R6*R7*s + C3*C4*R5*R6*R7*s^2))

v_10 == (C1*C2*C3*C4*C5*R1*R2*R5*R6*R10*R11*Vin*s^5*(R3 + R4)*(R7 +
R8)*(C6*R9*s + 1))/((R3 + C1*R2*R3*s - C2*R1*R4*s + C2*R2*R3*s +
C1*C2*R1*R2*R3*s^2)*(R7 + C3*R6*R7*s - C4*R5*R8*s + C4*R6*R7*s +
C3*C4*R5*R6*R7*s^2)*(R11 + C5*R10*R11*s - C6*R9*R12*s + C6*R10*R11*s +
C5*C6*R9*R10*R11*s^2))

v_11 == (C1*C2*C3*C4*C5*C6*R1*R2*R5*R6*R9*R10*R11*Vin*s^6*(R3 + R4)*(R7
+ R8))/((R3 + C1*R2*R3*s - C2*R1*R4*s + C2*R2*R3*s +
C1*C2*R1*R2*R3*s^2)*(R7 + C3*R6*R7*s - C4*R5*R8*s + C4*R6*R7*s +
C3*C4*R5*R6*R7*s^2)*(R11 + C5*R10*R11*s - C6*R9*R12*s + C6*R10*R11*s +
C5*C6*R9*R10*R11*s^2))

v_12 == (C1*C2*C3*C4*C5*C6*R1*R2*R5*R6*R9*R10*R11*Vin*s^6*(R3 + R4)*(R7
+ R8))/((R3 + C1*R2*R3*s - C2*R1*R4*s + C2*R2*R3*s +
C1*C2*R1*R2*R3*s^2)*(R7 + C3*R6*R7*s - C4*R5*R8*s + C4*R6*R7*s +
C3*C4*R5*R6*R7*s^2)*(R11 + C5*R10*R11*s - C6*R9*R12*s + C6*R10*R11*s +
C5*C6*R9*R10*R11*s^2))

v_13 == (C1*C2*C3*C4*C5*C6*R1*R2*R5*R6*R9*R10*Vin*s^6*(R3 + R4)*(R7 +
R8)*(R11 + R12))/((R3 + C1*R2*R3*s - C2*R1*R4*s + C2*R2*R3*s +
C1*C2*R1*R2*R3*s^2)*(R7 + C3*R6*R7*s - C4*R5*R8*s + C4*R6*R7*s +
```

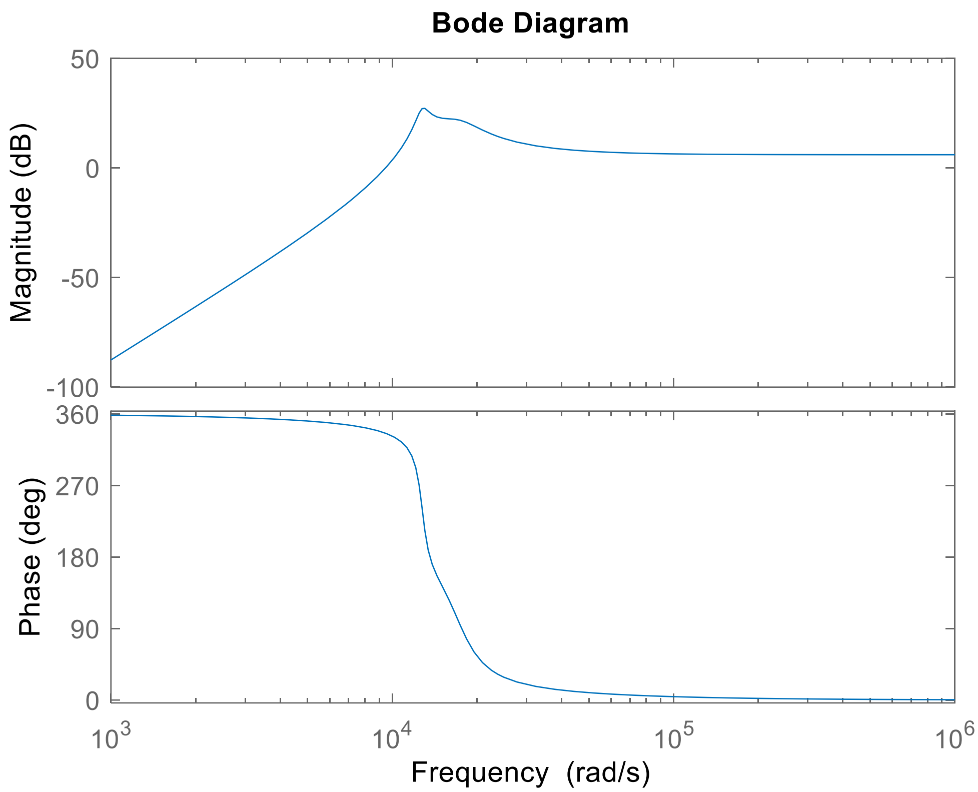
```
C3*C4*R5*R6*R7*s^2)*(R11 + C5*R10*R11*s - C6*R9*R12*s + C6*R10*R11*s +
C5*C6*R9*R10*R11*s^2))

I_Vin == -(C1*Vin*s*(R3 - C2*R1*R4*s + C2*R2*R3*s))/(R3 + C1*R2*R3*s -
C2*R1*R4*s + C2*R2*R3*s + C1*C2*R1*R2*R3*s^2)

I_OA1 == -(C1*Vin*s*(C2*R1*R2*R7*s - R3*R7 + C2*R1*R4*R7*s -
C3*R3*R6*R7*s + C4*R3*R5*R8*s - C4*R3*R6*R7*s + C2*C3*R1*R2*R3*R7*s^2
+ C2*C3*R1*R2*R4*R7*s^2 + C2*C3*R1*R2*R6*R7*s^2 - C2*C4*R1*R2*R5*R8*s^2
+ C2*C4*R1*R2*R6*R7*s^2 + C2*C3*R1*R4*R6*R7*s^2 - C2*C4*R1*R4*R5*R8*s^2
+ C2*C4*R1*R4*R6*R7*s^2 - C3*C4*R3*R5*R6*R7*s^2 -
C2*C3*C4*R1*R2*R3*R5*R8*s^3 + C2*C3*C4*R1*R2*R3*R6*R7*s^3 -
C2*C3*C4*R1*R2*R4*R5*R8*s^3 + C2*C3*C4*R1*R2*R4*R6*R7*s^3 +
C2*C3*C4*R1*R2*R5*R6*R7*s^3 + C2*C3*C4*R1*R4*R5*R6*R7*s^3))/((R3 +
C1*R2*R3*s - C2*R1*R4*s + C2*R2*R3*s + C1*C2*R1*R2*R3*s^2)*(R7 +
C3*R6*R7*s - C4*R5*R8*s + C4*R6*R7*s + C3*C4*R5*R6*R7*s^2))
  I_OA2 == -(C1*C2*C3*R1*R2*Vin*s^3*(R3 + R4)*(C4*R5*R6*R11*s - R7*R11
+ C4*R5*R8*R11*s - C5*R7*R10*R11*s + C6*R7*R9*R12*s - C6*R7*R10*R11*s
+ C4*C5*R5*R6*R7*R11*s^2 + C4*C5*R5*R6*R8*R11*s^2 +
C4*C5*R5*R6*R10*R11*s^2 - C4*C6*R5*R6*R9*R12*s^2 +
C4*C6*R5*R6*R10*R11*s^2 + C4*C5*R5*R8*R10*R11*s^2 -
C4*C6*R5*R8*R9*R12*s^2 + C4*C6*R5*R8*R10*R11*s^2 -
C5*C6*R7*R9*R10*R11*s^2 - C4*C5*C6*R5*R6*R7*R9*R12*s^3 +
C4*C5*C6*R5*R6*R7*R10*R11*s^3 - C4*C5*C6*R5*R6*R8*R9*R12*s^3 +
C4*C5*C6*R5*R6*R8*R10*R11*s^3 + C4*C5*C6*R5*R6*R9*R10*R11*s^3 +
C4*C5*C6*R5*R8*R9*R10*R11*s^3))/((R3 + C1*R2*R3*s - C2*R1*R4*s +
C2*R2*R3*s + C1*C2*R1*R2*R3*s^2)*(R7 + C3*R6*R7*s - C4*R5*R8*s +
C4*R6*R7*s + C3*C4*R5*R6*R7*s^2)*(R11 + C5*R10*R11*s - C6*R9*R12*s +
C6*R10*R11*s + C5*C6*R9*R10*R11*s^2))

I_OA3 == -(C1*C2*C3*C4*C5*R1*R2*R5*R6*Vin*s^5*(R3 + R4)*(R7 +
R8)*(C6*R9*R10*s - R11 + C6*R9*R12*s))/((R3 + C1*R2*R3*s - C2*R1*R4*s
+ C2*R2*R3*s + C1*C2*R1*R2*R3*s^2)*(R7 + C3*R6*R7*s - C4*R5*R8*s +
C4*R6*R7*s + C3*C4*R5*R6*R7*s^2)*(R11 + C5*R10*R11*s - C6*R9*R12*s +
C6*R10*R11*s + C5*C6*R9*R10*R11*s^2))


  Elapsed time is 273.274 seconds.
```

```
>> v_13/Vin

ans =

(C1*C2*C3*C4*C5*C6*R1*R2*R5*R6*R9*R10*s^6*(R3 + R4)*(R7 + R8)*(R11 + R12))/((R3 + C1*R2*R3*s - C2*R1*R4*s + C2*R2*R3*s + C1*C2*R1*R2*R3*s^2)*(R7 + C3*R6*R7*s - C4*R5*R8*s + C4*R6*R7*s + C3*C4*R5*R6*R7*s^2)*(R11 + C5*R10*R11*s - C6*R9*R12*s + C6*R10*R11*s + C5*C6*R9*R10*R11*s^2))

>> eval(ans)

ans =

(106502304368873692267109375*s^6)/(850705917302346158658436518579420 52864*((1805080315891145*s^2)/295147905179352825856 + (10767*s)/40000 + 9700)*((2413979114245865*s^2)/9223372036854775808 + (10993005215434 6603*s)/351843720888320000 + 42800)*((7213875971185227*s^2)/73786976294838206464 + (1189*s)/2000 + 29000))

>> simplify(ans)

ans =

(106502304368873692267109375*s^6)/(850705917302346158658436518579420 52864*((1805080315891145*s^2)/295147905179352825856 + (10767*s)/40000 + 9700)*((2413979114245865*s^2)/9223372036854775808 + (10993005215434 6603*s)/351843720888320000 + 42800)*((7213875971185227*s^2)/73786976294838206464 + (1189*s)/2000 + 29000))

>>  H=v_7/Vin

H =
```

```
(C1*C2*C3*C4*R1*R2*R5*R6*R7*s^4*(R3 + R4))/((R3 + C1*R2*R3*s - C2*R1*R4*s + C2*R2*R3*s + C1*C2*R1*R2*R3*s^2)*(R7 + C3*R6*R7*s - C4*R5*R8*s + C4*R6*R7*s + C3*C4*R5*R6*R7*s^2))

>> H=collect(H)

H =

((C1*C2*C3*C4*R1*R2*R3*R5*R6*R7 + C1*C2*C3*C4*R1*R2*R4*R5*R6*R7)*s^4)/(C1*C2*C3*C4*R1*R2*R3*R5*R6*R7*s^4 + (C1*C2*C3*R1*R2*R3*R6*R7 - C1*C2*C4*R1*R2*R3*R5*R8 + C1*C2*C4*R1*R2*R3*R6*R7 + C1*C3*C4*R2*R3*R5*R6*R7 - C2*C3*C4*R1*R4*R5*R6*R7 + C2*C3*C4*R2*R3*R5*R6*R7)*s^3 + (C1*C2*R1*R2*R3*R7 + C1*C3*R2*R3*R6*R7 - C1*C4*R2*R3*R5*R8 + C1*C4*R2*R3*R6*R7 - C2*C3*R1*R4*R6*R7 + C2*C3*R2*R3*R6*R7 + C2*C4*R1*R4*R5*R8 - C2*C4*R1*R4*R6*R7 - C2*C4*R2*R3*R5*R8 + C2*C4*R2*R3*R6*R7 + C3*C4*R3*R5*R6*R7)*s^2 + (C1*R2*R3*R7 - C2*R1*R4*R7 + C2*R2*R3*R7 + C3*R3*R6*R7 - C4*R3*R5*R8 + C4*R3*R6*R7)*s + R3*R7)

>> pretty(H)
                                                                              4
4
((C1 C2 C3 C4 R1 R2 R3 R5 R6 R7 + C1 C2 C3 C4 R1 R2 R4 R5 R6 R7) s
)/(C1 C2 C3 C4 R1 R2 R3 R5 R6 R7 s

   + (C1 C2 C3 R1 R2 R3 R6 R7 - C1 C2 C4 R1 R2 R3 R5 R8 + C1 C2 C4
R1 R2 R3 R6 R7

   + C1 C3 C4 R2 R3 R5 R6 R7 - C2 C3 C4 R1 R4 R5 R6 R7 + C2 C3 C4 R2
R3 R5 R6 R7)

    3
   s  + (C1 C2 R1 R2 R3 R7 + C1 C3 R2 R3 R6 R7 - C1 C4 R2 R3 R5 R8 +
C1 C4 R2 R3 R6 R7 - C2 C3 R1 R4 R6 R7

   + C2 C3 R2 R3 R6 R7 + C2 C4 R1 R4 R5 R8 - C2 C4 R1 R4 R6 R7 - C2
C4 R2 R3 R5 R8 + C2 C4 R2 R3 R6 R7
```

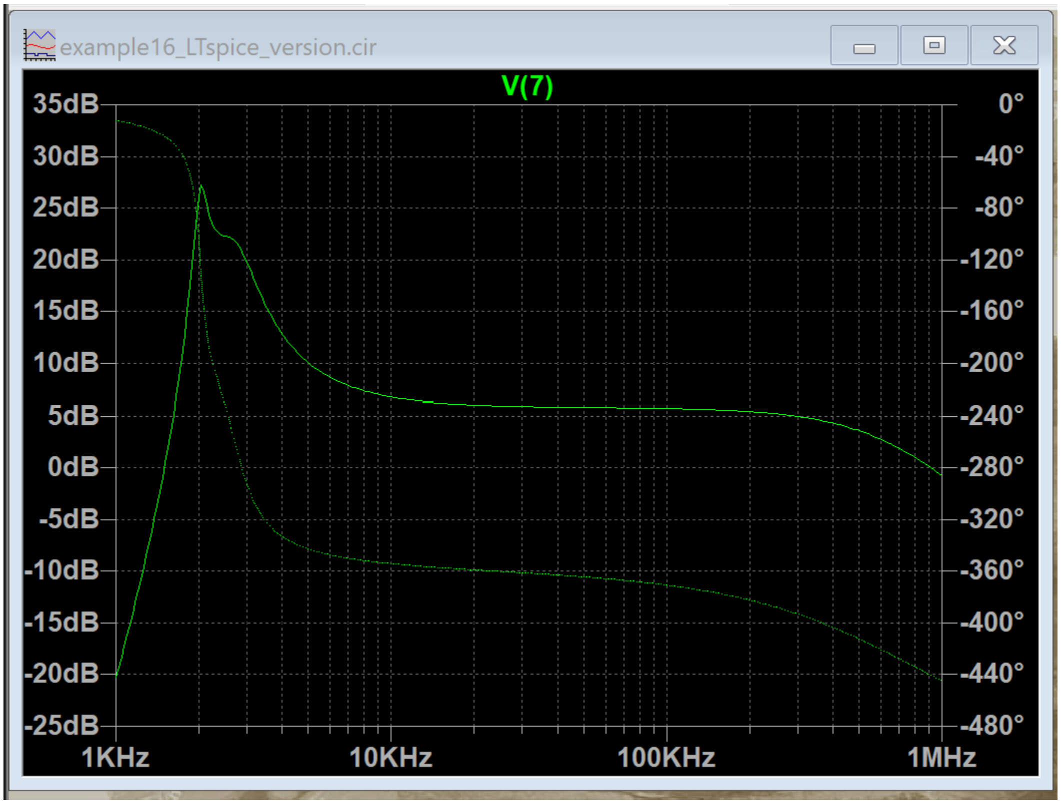
```
                                      2
    + C3 C4 R3 R5 R6 R7) s  + (C1 R2 R3 R7 - C2 R1 R4 R7 + C2 R2 R3
R7 + C3 R3 R6 R7 - C4 R3 R5 R8

    + C4 R3 R6 R7) s + R3 R7)

  >> Hnumbers=eval(H)

  Hnumbers =

  (7733434305024483*s^4)/(151115727451828646838272*((7733434305024483
*s^4)/302231454903657293676544                                       +
(6867394037895219*s^3)/36893488147419103232                          +
(6732960897675319*s^2)/562949953421312                               +
(4742380567574609*s)/137438953472 + 1241200000))

  >> [n,d]=numden(Hnumbers)

  n =

  15466868610048966*s^4



  d =

  7733434305024483*s^4      +      56257691958437634048*s^3        +
3614730857595287191420928*s^2   +   10428605154774458211833479168*s   +
375129681826419432911326412800000

  >> mySys=tf(sym2poly(n),sym2poly(d))

  mySys =

                             1.547e16 s^4
    ----------------------------------------------------------------
-
    7.733e15 s^4 + 5.626e19 s^3 + 3.615e24 s^2 + 1.043e28 s + 3.751e32
```

```
  Continuous-time transfer function.

>> bode(mySys)
```



LTspice Version:

```
* Example16.cir
*
Vin 1 0 AC 1 0
*
* Stage 1
C1 1 2 5E-9
C2 2 3 5E-9
R1 3 0 21.4E3
R2 2 5 11.43E3
R3 4 0 42.8E3
R4 4 5 42.8E3
XOA1 3 4 50 51 5 LM741/NS
*
* Stage 2
C3 5 6 5E-9
C4 6 7 5E-9
R5 7 0 14.5E3
R6 6 9 9.3E3
R7 8 0 29E3
R8 8 9 29E3
XOA2 7 8 50 51 9 LM741/NS
*
* Stage 3
C5 9 10 5E-9
C6 10 11 5E-9
R9 11 0 4.85E3
R10 10 13 5.2E3
R11 12 0 9.7E3
R12 12 13 9.7E3
XOA3 11 12 50 51 13 LM741/NS
*
.LIB LM741.MOD
V+ 50 0 DC 18
V- 51 0 DC -15
* ANALYSIS
.AC DEC 100 1k 1Meg
.END
```

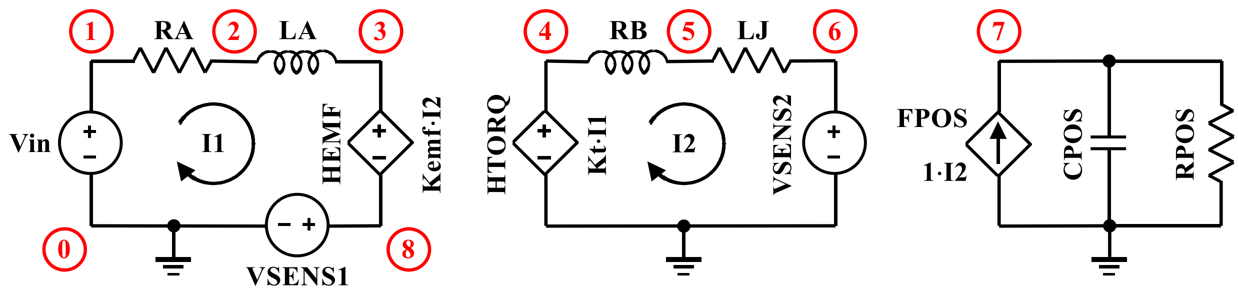


example16_LTspice_version.cir
V(7)
35dB
30dB
25dB
20dB
15dB
10dB
5dB
0dB
-5dB
-10dB
-15dB
-20dB
-25dB
0°
-40°
-80°
-120°
-160°
-200°
-240°
-280°
-320°
-360°
-400°
-440°
-480°
1KHz
10KHz
100KHz
1MHz

Example 17:

Now, as a final example, consider the following circuit. This is a model for a DC motor, adapted from a description at http://ecircuitcenter.com/Circuits/dc_motor_model/DCmotor_model.htm.

Circuit



Electrical Components *Mechanical Components* *Convert Velocity to Position*

Physical Interpretation

V(1)	– Motor Voltage	|	RA	– Terminal Resistance	= 0.5 Ω
I1	– Motor Current	|	LA	– Terminal Inductance	= 1.5 mH
V(3)	– Back emf Voltage	|	Kemf	– Speed Constant	= 0.05 V/rad/s
V(4)	– Motor Torque	|	Kt	– Torque Constant	= 0.05 Nm/A
I2	– Angular Velocity	|	RB	– Friction	= 0.0001 Nm/rad/s
V(7)	– Angular Position	|	LJ	– Motor and Load Inertia	= 0.00025 Nm/rad/s

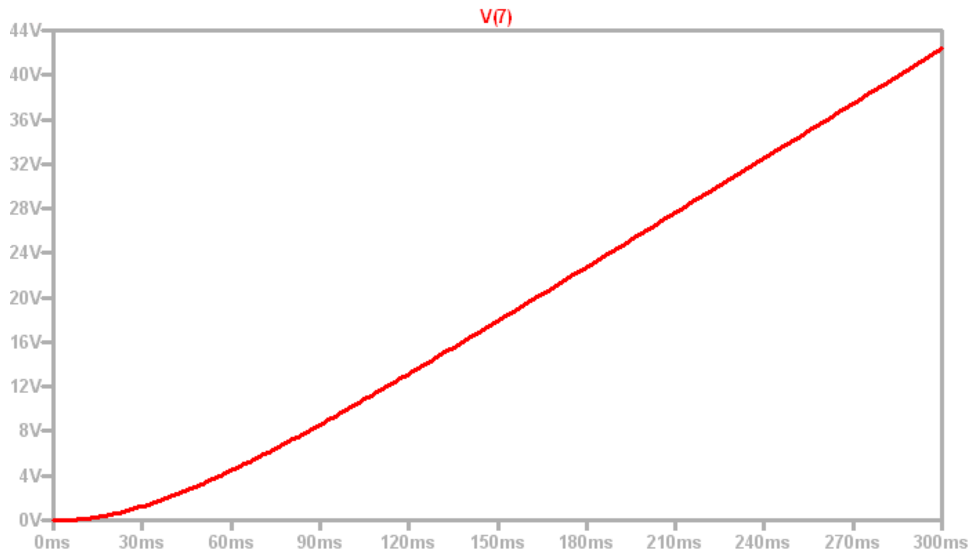
Two analyses will be conducted here for comparison, one with LTspice, and the other with SCAM.

LTspice Netlist

```
* Example17.cir
*
Vin 1 0 10
*
* MOTOR VOLTAGE
RA 1 2 0.5
LA 2 3 0.0015 IC=0
HEMF 3 8 VSENSE2 0.05
VSENSE1 8 0 0
*
* MOTOR TORQUE BASED ON INERTIA AND FRICTION
HTORQ 4 0 VSENSE1 0.05
RB 4 5 0.0001
LJ 5 6 0.00025 IC=0
VSENSE2 6 0 0
*
* MOTOR POSITION
```

```
FPOS 0 7 VSENSE2 1
CPOS 7 0 1 IC=0
RPOS 7 0 1MEG
*
* ANALYSIS
.TRAN 10MS 0.3S UIC
.END
```

LTspice Result



SCAM Netlist

```
Vin 1 0 10
RA 1 2 0.5
LA 2 3 0.0015
HEMF 3 8 VSENSE2 0.05
VSENSE1 8 0 0
HTORQ 4 0 VSENSE1 0.05
RB 4 5 0.0001
LJ 5 6 0.00025
VSENSE2 6 0 0
FPOS 0 7 VSENSE2 1
CPOS 7 0 1
RPOS 7 0 1E6
```

SCAM/MATLAB Result

```
>> fname="example17.cir";
>> ISU_scam
```

```
Started -- please be patient.

Netlist:
Vin 1 0 10
RA 1 2 0.5
LA 2 3 0.0015
HEMF 3 8 VSENSE2 0.05
VSENSE1 8 0 0
HTORQ 4 0 VSENSE1 0.05
RB 4 5 0.0001
LJ 5 6 0.00025
VSENSE2 6 0 0
FPOS 0 7 VSENSE2 1
CPOS 7 0 1
RPOS 7 0 1E6



The A matrix:
[ 1/RA,           -1/RA,         0,     0,                0,          0,
0,  0, 1,  0,      0, 0,     0]
[-1/RA, 1/RA + 1/(LA*s), -1/(LA*s),     0,                0,          0,
0,  0, 0,  0,      0, 0,     0]
[    0,       -1/(LA*s),  1/(LA*s),     0,                0,          0,
0,  0, 0,  1,      0, 0,     0]
[    0,               0,         0,  1/RB,            -1/RB,          0,
0,  0, 0,  0,      0, 1,     0]
[    0,               0,         0, -1/RB, 1/RB + 1/(LJ*s), -1/(LJ*s),
0,  0, 0,  0,      0, 0,     0]
[    0,               0,         0,     0,        -1/(LJ*s),  1/(LJ*s),
0,  0, 0,  0,      0, 0,     1]
[    0,               0,         0,     0,                0,          0,
CPOS*s + 1/RPOS,  0, 0,  0,       0, 0, -FPOS]
[    0,               0,         0,     0,                0,          0,
0,  0, 0, -1,      1, 0,     0]
[    1,               0,         0,     0,                0,          0,
0,  0, 0,  0,      0, 0,     0]
[    0,               0,         1,     0,                0,          0,
0, -1, 0,  0,      0, 0, -HEMF]
[    0,               0,         0,     0,                0,          0,
0,  1, 0,  0,      0, 0,     0]
[    0,               0,         0,     1,                0,          0,
0,  0, 0,  0, -HTORQ, 0,     0]
[    0,               0,         0,     0,                0,          1,
0,  0, 0,  0,      0, 0,     0]


The x vector:
      v_1
      v_2
```

```
       v_3
       v_4
       v_5
       v_6
       v_7
       v_8
     I_Vin
    I_HEMF
I_VSENSE1
  I_HTORQ
I_VSENSE2


The z vector:
      0
      0
      0
      0
      0
      0
      0
      0
    Vin
      0
VSENSE1
      0
VSENSE2


The matrix equation:
                    I_Vin + v_1/RA - v_2/RA == 0
v_2*(1/RA + 1/(LA*s)) - v_1/RA - v_3/(LA*s) == 0
           I_HEMF - v_2/(LA*s) + v_3/(LA*s) == 0
                  I_HTORQ + v_4/RB - v_5/RB == 0
v_5*(1/RB + 1/(LJ*s)) - v_4/RB - v_6/(LJ*s) == 0
        I_VSENSE2 - v_5/(LJ*s) + v_6/(LJ*s) == 0
     v_7*(CPOS*s + 1/RPOS) - FPOS*I_VSENSE2 == 0
                         I_VSENSE1 - I_HEMF == 0
                                 v_1 == Vin
                 v_3 - v_8 - HEMF*I_VSENSE2 == 0
                             v_8 == VSENSE1
                     v_4 - HTORQ*I_VSENSE1 == 0
                             v_6 == VSENSE2


The solution:

v_1 == Vin
                            v_2 == (HEMF*HTORQ*Vin - HEMF*RA*VSENSE2 +
RA*RB*VSENSE1 + LA*LJ*Vin*s^2 + LJ*RA*VSENSE1*s + LA*RB*Vin*s)/(RA*RB +
HEMF*HTORQ + LA*RB*s + LJ*RA*s + LA*LJ*s^2)
```

```
v_3 == (HEMF*HTORQ*Vin - HEMF*RA*VSENSE2 + RA*RB*VSENSE1 +
LA*LJ*VSENSE1*s^2 - HEMF*LA*VSENSE2*s + LA*RB*VSENSE1*s +
LJ*RA*VSENSE1*s)/(RA*RB + HEMF*HTORQ + LA*RB*s + LJ*RA*s + LA*LJ*s^2)
                                                              v_4 ==
(HTORQ*(RB*Vin - RB*VSENSE1 + HEMF*VSENSE2 - LJ*VSENSE1*s +
LJ*Vin*s))/(RA*RB + HEMF*HTORQ + LA*RB*s + LJ*RA*s + LA*LJ*s^2)
                                  v_5 == (HEMF*HTORQ*VSENSE2 +
RA*RB*VSENSE2 - HTORQ*LJ*VSENSE1*s + HTORQ*LJ*Vin*s +
LA*RB*VSENSE2*s)/(RA*RB + HEMF*HTORQ + LA*RB*s + LJ*RA*s + LA*LJ*s^2)

v_6 == VSENSE2
                                      v_7 == -(FPOS*RPOS*(RA*VSENSE2 +
HTORQ*VSENSE1 - HTORQ*Vin + LA*VSENSE2*s))/((CPOS*RPOS*s + 1)*(RA*RB +
HEMF*HTORQ + LA*RB*s + LJ*RA*s + LA*LJ*s^2))

v_8 == VSENSE1
                                                             I_Vin == -
(RB*Vin - RB*VSENSE1 + HEMF*VSENSE2 - LJ*VSENSE1*s + LJ*Vin*s)/(RA*RB +
HEMF*HTORQ + LA*RB*s + LJ*RA*s + LA*LJ*s^2)
                                                             I_HEMF ==
(RB*Vin - RB*VSENSE1 + HEMF*VSENSE2 - LJ*VSENSE1*s + LJ*Vin*s)/(RA*RB +
HEMF*HTORQ + LA*RB*s + LJ*RA*s + LA*LJ*s^2)
                                                          I_VSENSE1 ==
(RB*Vin - RB*VSENSE1 + HEMF*VSENSE2 - LJ*VSENSE1*s + LJ*Vin*s)/(RA*RB +
HEMF*HTORQ + LA*RB*s + LJ*RA*s + LA*LJ*s^2)

I_HTORQ == (RA*VSENSE2 + HTORQ*VSENSE1 - HTORQ*Vin +
LA*VSENSE2*s)/(RA*RB + HEMF*HTORQ + LA*RB*s + LJ*RA*s + LA*LJ*s^2)

I_VSENSE2 == -(RA*VSENSE2 + HTORQ*VSENSE1 - HTORQ*Vin +
LA*VSENSE2*s)/(RA*RB + HEMF*HTORQ + LA*RB*s + LJ*RA*s + LA*LJ*s^2)


Elapsed time is 2.1632 seconds.
>> v_7

v_7 =

-(FPOS*RPOS*(RA*VSENSE2 + HTORQ*VSENSE1 - HTORQ*Vin +
LA*VSENSE2*s))/((CPOS*RPOS*s + 1)*(RA*RB + HEMF*HTORQ + LA*RB*s +
LJ*RA*s + LA*LJ*s^2))

>> eval(v_7)

ans =

500000/((1000000*s + 1)*((1770887431076117*s^2)/4722366482869645213696 +
(591004165331136098587*s)/4722366482869645213696000 + 51/20000))

>> simplify(ans)
```

```
ans =

11805916207174113034240000000000/((1000000*s +
1)*(8854437155380585000*s^2 + 2955020826655680492935*s +
60210172656587976474624))
```