Symbulator

A symbolic simulator of linear electric circuits

Symbulator 8

TI-Nspire CX II CAS

Roberto Perez-Franco

Contents

Introduction **5**

What is Symbulator 5

Why use Symbulator? 5

Download 6

Before you start 7

Direct current analysis **8**

1.1 Run a direct current analysis 8

1.2 How to describe a resistor 9

1.3 What answers do you get 10

1.4 How to describe a voltage source 11

1.5 A numerical DC simulation, step by step 12

1.6 Instructive numerical examples, solved 16

Symbolic circuits and expert mode **28**

2.1 About symbolic problems 28

2.2 Numerical from symbolic, with solve 30

2.3 Numerical from symbolic, using ex 31

2.4 Instructive symbolic examples, solved 32

Current sources, conductances and dependent sources **36**

3.1 How to describe a current source 36

3.2 What about conductances? 37

3.3 How to reduce parallel resistors 38

3.4 How to describe dependent sources 41

3.5 Instructive solved examples 41

Shorts, equivalent resistance and Thévenin/Norton **65**

4.1 How to describe a short circuit 65

4.2 The equivalent resistance script: er 66
4.3 The Thévenin / Norton script: th 68
4.4 Instructive solved examples 72

Operational amplifiers 97

5.1 How to describe an ideal operational amplifier 97
5.2 Instructive Op Amp examples 99

Transient analysis 123

6.1 How to describe a capacitor 123
6.2 How to describe an inductor 124
6.3 Running a transient analysis 124
6.4 Two useful tools 126
6.5 Instructive solved examples 127

Alternating current analysis 165

7.1 AC analysis in Symbulator 165
7.2 Solved numerical examples 166
7.3 Solved numerical-from-symbolic examples 175
7.4 Solved symbolic examples 177

Power in AC circuits 179

8.1 Average power 179
8.2 RMS and PF 180
8.3 Complex power 182

Three-phase circuits 186

9.1 About solving three-phase circuits 186
9.2 Wye-Wye 186
9.3 Wye-Delta 192
9.4 Delta-Delta 194
9.5 Delta-Wye 198

Mutual inductance and ideal transformers 199

10.1 How to describe mutual inductance 199

10.2 Instructive m problems 200
10.3 How to describe ideal transformers 202
10.4 Instructive t problems 202

Bode plots and resonance **205**

The complex frequency domain **206**

12.1 s-domain analysis 206
12.2 The t2s and s2t shortcuts 207
12.3 Instructive FD problems 208
12.4 Transfer function problems 209

Two-ports **210**

13.1 Find two-port equivalents 210
13.2 Use two-ports as elements 211
13.3 Instructive two-port problems 212
13.4 The gain tool 216

Roll the credits **218**

Who made Symbulator 218
Acknowledgements 220
Problem credits 221
Praise received 221
Licence 222

Index **223**

Introduction

Learn about Symbulator and how it can help you focus on learning how circuits work, instead of struggling with math. Get the latest version of Symbulator and get ready to roll.

What is Symbulator

Symbulator is a program to solve linear electric circuits symbolically and numerically in a calculator. It takes numerical and symbolic values, and returns numerical and symbolic results. The odd-sounding name is a portmanteau of *"**symb**olic sim**ulator**"*. Symbulator is widely regarded as the best symbolic simulator of linear electric circuits ever made for a calculator.

What analyses can it do?

Symbulator can solve circuits using four types of analysis: direct current (DC), alternating current (AC), transient time-domain (TR), and complex frequency-domain (FD). It analyses numerical and, more importantly, symbolic linear circuits, returning the voltages in all nodes and the currents, voltage drops and power in all elements. It also finds the passive, Thévenin / Norton and two-port equivalent of a circuit, and draws time domain plots.

What elements does it accept?

Symbulator accepts elements such as resistors, inductors, mutual inductance, capacitors, independent and dependent current and voltage sources, ideal operational amplifiers, ideal transformers and six types of two-ports.

Where does it run?

Symbulator 8, released in 2023, is adapted to run on the TI-Nspire CX II CAS calculator by Texas Instruments. There is also a version for the TI-89 Titanium called version 7, and a newer Python/SymPy port called version 9.

Why use Symbulator?

Symbulator is extremely useful on the circuit theory problems engineering students face in Circuits I and II. It lets the student focus on understanding circuit analysis rather than on the mathematics of solving it.

Strengths

- A fantastic tool to solve circuits composed of ideal, linear elements.
- Shines in solving circuits with symbolic values, and symbolic results.
- Offers the absolute simplest way to define dependent sources.
- Includes ideal transformers and six different two-ports.

Limitations

- Does not simulate non-linear elements, such as diodes and transistors.
- Will not answer under-defined problems that rely on applying formulas selectively rather than solving a circuit completely.
- Is not a replacement for your brain or an excuse to not study. You must understand circuit theory to use it.

Symbulator is free!

Symbulator has always been free of charge, and always will be. *(You are welcome!)* Since 2026, Symbulator is also open source under the MIT licence. You are free to use it, change it and port it to other platforms; all the licence asks is that the copyright notice and the attribution to the original author travel with it.

Download

Symbulator

The version of Symbulator for the TI-Nspire CX II CAS is called version 8 (date stamp **08 July 2023**). As bugs are found and fixed, we will revise and reupload, updating the date stamp.

To get Symbulator 8, download this file: **s.tns**, transfer it to your calculator, and refresh the libraries.

Laplace Functions

Since the CAS in the TI-Nspire CX II CAS does not do Laplace transforms, Symbulator relies for this on another software, which I call Laplace Functions.

This software was made, around the same time Symbulator was being created, by my friend Lars Frederiksen, one of the most gifted programmers for TI-Basic ever. Lars made two programs for the TI-92+/Voyage and the TI-89: one he called DiffEq, because it also solves differential equations, and the other Advanced Laplace, which packed improved versions of his Laplace functions.

The latter was automatically adapted for the TI-Nspire by Philippe Fortin, who renamed the group of functions ETS_specfunc. Not being a fan of that name, and given my friendship and collaboration with Lars, I have taken the liberty of renaming the file here to **LF**,

which in my eyes stands both for Laplace Functions and for Lars Frederiksen.

Download **lf.tns**, transfer it to your calculator, and refresh the libraries.

Solved problems

I have solved all the problems in this documentation, each in a separate TI-Nspire document. This zip file, **problems.zip**, contains three hundred solved problems.

> CAUTION
>
> **Use at your own risk!**
>
> Every effort has gone into Symbulator, and it has no known bugs. Even so, it is provided “as is”, without warranty of any kind, express or implied, including but not limited to the warranty of fitness for a particular purpose. You use it at your own risk.

Please report any problems to help [at] symbulator [dot] com

Before you start

Clean slate

Before running a simulation in the Nspire, either create a new document or delete from the current one every variable unrelated to your simulation.

And if a simulation fails or aborts and leaves variables behind, move to a new document or delete them before running it again.

> TIP
>
> **Pro Tip**
>
> Start each simulation from a clean document, and clear out anything a failed simulation left behind before trying again.

LESSON 1

Direct current analysis

Learn to describe a circuit in Symbulator, and how to run a *direct current* analysis using **dc**. Learn how to describe a *voltage source* using **e**, and a *resistor* using **r**.

I firmly believe one learns best by *doing*, so I have written this documentation as a tutorial, where you learn Symbulator by working through a series of progressively more diverse and complex examples. In a moment, I'll teach you how to swim. But for now, let me throw you in the deep end!

1.1 Run a direct current analysis

I want you to create a new document in your calculator, or to delete all variables you don't need from your current one (ideally, the current document should be empty), and type this:

```
type
s\dc("e1,1,0,36:r1,1,2,1'k:r2,2,3,3'k:r3,3,0,2'k")
```

Now I want you to press ENTER.

If you typed everything correctly and your calculator is properly set up with Symbulator, you should have seen several lines of text appear, with a series of cryptic messages. Then, after a few seconds, it should have displayed a short Done.

Congratulations! You have just run your first simulation in Symbulator.

How does it work

Let's now go over what we just did, one piece at a time.

To analyse a direct current circuit in Symbulator, we use a program called **s\dc()**. The program itself is called **dc**, but since it is found in the Symbulator library **s**, we have to refer to it as **s\dc**, so the calculator knows exactly what we want. And because it is a program, it carries the parenthesis **()** after its name.

The dc program takes one argument as its input: the description of the circuit you want to analyse. That input goes inside the parenthesis.

In Symbulator, a circuit description takes the form of a string of text, describing a series of elements, separated by colons.

The circuit description can be passed to the program directly as an argument, as we did

above, or it can be stored into a variable whose name is then fed in, like this:

```
type
"e1,1,0,36:r1,1,2,1'k:r2,2,3,3'k:r3,3,0,2'k"→cir
s\dc(cir)
```

The results are exactly the same as doing it in a single step. I like to use **cir** as the name of the variable where I store my circuits, and that is what you will see in this tutorial, though any valid variable name will do. You will see both ways used throughout.

Look at the elements

Now let's examine the contents of the circuit description. Study it and you will see three colons. In Symbulator, colons separate elements in a circuit description, so putting them aside, this description has four elements:

- The first one is `e1,1,0,36`
- the second one is `r1,1,2,1'k`
- the third one is `r2,2,3,3'k`
- the fourth one is `r3,3,0,2'k`

At the moment they may look cryptic. But by the end of this lesson you will read them with ease.

1.2 How to describe a resistor

Notice that the second, third and fourth elements in the list all start with the letter **r**. These are resistors.

> ⓘ NOTE
>
> **Describing resistors**
>
> An ideal resistor is described with four pieces of information, separated by commas: a unique name (which must start with the letter **r**), the names of its first and second node, and its value in ohms (Ω).
> For example, an ideal resistor called **r1**, between nodes **a** and **b**, of **300** Ω, is described `r1,a,b,300`

How to name resistors

Every resistor must have a name. As long as it starts with the letter **r** and is unique, it can be whatever you want. *Unique* means no other element or node has the same name.

Two exceptions I know of so far: `r` and `rr` should not name resistors, because some of the answer variables they generate collide with reserved ones.

> ⚠ WARNING
>
> **Can't touch r or rr**
>
> The names `r` and `rr` cannot be used, because their answers conflict with reserved variables in the calculator.

How to name nodes

Naming the nodes is the first thing we do to solve a circuit. Every node must have a name, and node names must be unique: no other node or element may have the same one. Otherwise they can be whatever you want, with two constraints. The first one is that: at least one node should be called **0** (zero). This will be considered the *ground node* of your circuit and will have, by definition, a voltage of 0 volts.

> ⚠ WARNING
>
> **You need a ground node**
>
> Every circuit must have a node called **0** to serve as reference or ground.

The second restriction is that you cannot use variables that are reserved in the calculator as names. These are listed in the User's Manual of the calculator.

Values can use SI prefixes

The values of circuit elements are often given with prefixes of the International System (SI) — kilo, milli, micro. So Symbulator has a shorthand for them: an SI prefix in an element's value, preceded by an apostrophe, multiplies it by the corresponding factor, exactly.

> 💡 TIP
>
> **Prefixes welcome!**
>
> You can use SI prefixes, preceded by an apostrophe, in the values of elements.

For example, an 8 kΩ resistor can be entered in many ways: `8000`, `8'k`, `8000.` and `8E3` are all equivalent, except that the first two are treated as exact and the other two as approximate.

1.3 What answers do you get

After the simulation in DC is complete, Symbulator stores a series of answers in the calculator's memory, labelled with easy to remember names for your convenience.

Answer for each node

Symbulator calculates for each node a voltage with reference to ground, which is stored in a variable called v plus the name of the node. For example, for a node called **1**, its voltage is stored in a variable called v1.

Answers for each resistor

Symbulator calculates for each resistor the following answers:

- The voltage drop in the resistor, defined as the voltage in the first node minus the voltage in the second node, in volts. For a resistor called **r5**, this is stored in `vr5`.
- The current through the resistor, flowing from the first node towards the second, in amperes. For a resistor called **rx**, this is stored in `irx`.
- The power consumed by the resistor, in watts. For a resistor called **r12**, this is stored in `pr12`.

By now you should understand the description of the second, third and fourth elements in our example.

1.4 How to describe a voltage source

The first element in our example, on the other hand, started with the letter **e**. This element is a voltage source.

> ⓘ NOTE
>
> **Describing voltage sources**
>
> In Symbulator, a voltage source is described with four pieces of information, separated by commas: a unique name (which must start with the letter **e**), the names of its positive and negative node, and its value in volts (V).
> For example, an ideal voltage source called **e1**, with its positive terminal in node **3**, its negative terminal in the ground node **0**, and a value of 12 volts, is described `e1,3,0,12`

How to name voltage sources

Every voltage source in your circuit must have a unique name that starts with the letter **e**, and it can be whatever you want.

Values can use SI prefixes

A voltage source's value can use SI prefixes too — they work in the values of every element.

Answers for a voltage source

For each voltage source, the following answers are calculated:

- The voltage drop in the source, defined as the voltage in the first node minus the voltage in the second, in volts. For a source called **e5**, `ve5`.
- The current through the source, flowing from the first node towards the second, in amperes. For a source called **ex**, `iex`.

- The power consumed — attention: not delivered, but consumed — by the source, in watts. For a source called **e12**, `pe12`. If we want the power delivered, we ask for the negative of this value.
- The equivalent resistance of the rest of the circuit, as seen by the source. For a source called **e2**, `re2`.

> NOTE
> **For sources only**
> The equivalent resistance is a property of the *view from a source*, not for every element. It exists only for sources.
> There is no `rr1` for a resistor — a resistor's resistance is the value you gave it. The same applies to the impedance answers in AC analysis, which you will meet in Lesson 7 (page 165).

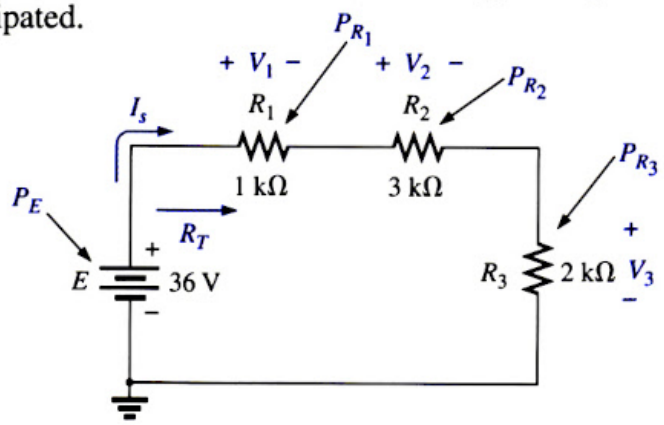
1.5 A numerical DC simulation, step by step

Let's go back now to the simulation we ran earlier. That simulation corresponds to the circuit given in Example 5.7 in Boylestad's *Introductory Circuit Analysis* (11ed). Moving forward, I will refer to that textbook as **B11**.

The problem statement and the circuit schematic are reproduced exactly as they appear in the textbook, as they will be for every other problem in this tutorial. Since this is for educational purposes, it is my understanding that it falls squarely within the "fair use" doctrine of copyright law. In any event, no copyright infringement is meant.

B11's Example 5.7

a. Determine the total resistance R_T.
b. Calculate the current I_s.
c. Determine the voltage across each resistor.
d. Find the power supplied by the battery.
e. Determine the power dissipated by each resistor.
f. Comment on whether the total power supplied equals the total power dissipated.



B11's Example 5.7

SOLUTION

All the values in this circuit are numbers: no element value is unknown. This is a *numerical circuit* — one where we know the numerical value of every element in it.

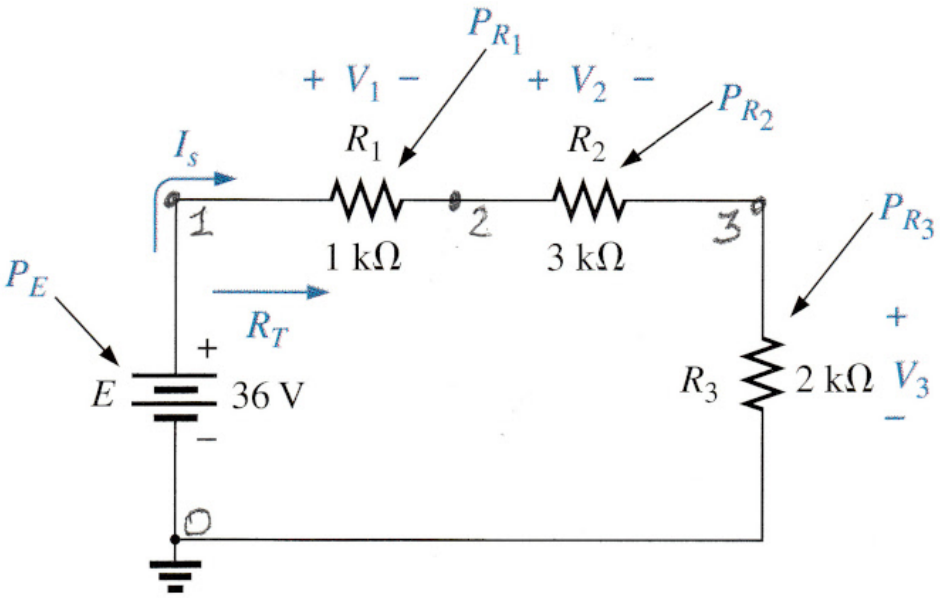
I will now walk you step by step through the solution. This process applies to most numerical simulations in Symbulator: (1) describe the circuit, (2) run the simulation, and (3) get the answers.

Step 1: Describe the circuit

Description starts with **naming the nodes**. As we said, you can call them anything you want, number or letter, as long as the name is unique — but one node must always be called **0** (zero), the ground node, with a voltage of 0 V. In this circuit the ground node is marked with the ground symbol. That is not always so; when it is not marked, you pick a node to serve as zero.

I labelled the nodes in this circuit, starting in the ground and moving clockwise, as 0, 1, 2 and 3. It helps me to pencil the names in the schematic itself.

> TIP
>
> **Write 'em down!**
>
> Write the names of your nodes and elements onto the circuit schematic itself, so you don't forget who's who.



B11's Example 5.7, with the node names pencilled in

After naming the nodes, I am ready to describe the elements of the circuit in Symbulator notation.

Let's start with the source: when I only have one voltage source, like here, I enjoy naming it with a single letter: **e**. So the voltage source is `e,1,0,36`: its name is **e**, its positive node is **1**, its negative node is **0**, and its value is 36 volts between them.

Now the resistors. I named the first **r1** and described it `r1,1,2,1'k`: its name is **r1**, its first node **1**, its second node **2**, and its value 1 kΩ. The second is `r2,2,3,3'k`, and the third `r3,3,0,2'k`.

> NOTE
>
> **An important point about SI prefixes**
>
> In the case of kilo, we can write it as either `'k` or `'K`. But that's not the case for every other SI prefix: their case matters. For example, `'m` means milli, while `'M` means mega. Symbulator's custom menu includes the spelling of all SI prefixes.

We pass the description to Symbulator as a string: open with a quotation mark, enter each element's description separated by colons, and close with a quotation mark. We can store the string in a variable:

```
type
"e,1,0,36:r1,1,2,1'k:r2,2,3,3'k:r3,3,0,2'k"→cir
```

Step 2: Run the simulation

We can now ask Symbulator to simulate this circuit in direct current:

```
type
s\dc(cir)
```

Symbulator says Done when a simulation completes. It took my calculator under 2 seconds to solve this circuit, in which time it found 16 answers and stored each in a memorably named variable in the current document.

For a DC analysis, these answers are as follows:

- The **voltage of each node**, so the voltages of nodes **1**, **2** and **3** are in v1, v2 and v3.
- The **current through each element**. The direction of the current is defined as going from the first node in the element's description to the second.
- The **voltage drop in each element**, defined as the difference in voltage between the first node in the element's description and the second, in that order.
- The **power consumed in each element**. An important point is that this is the power consumed, not the power delivered.
- Finally, **for each source**, **the equivalent resistance** of the rest of the circuit **as seen by that source**.

> NOTE
>
> **The sign is in the details**
>
> To read the signs of your answers correctly, mind the details:
>
> - The current in the elements is defined as going through the element from the first node to the second node.
> - The voltage drop is defined as the voltage of the first node minus the voltage of the second node.
> - The power given is the power consumed. So, if you want power delivered, evaluate the negative of that power.

Step 3: Get the answers

We can now answer the six questions in the problem.

Answer to question (a). The equivalent resistance as seen by the source e:

```
type
re
```

```
returns
6000
```

That is 6 kΩ. Correct.

Answer to question (b). Current I_s is defined in the schematic as the current flowing through the source, in the direction that goes from node **0** to node **1**. One way to find this value is evaluating the negative of the current through the source, which as you know flows in the opposite direction:

```
type
-ie
```

```
returns
.006
```

That is 6 mA. Another way, since this is a series circuit where every element carries the same current, is to evaluate the current through any of the resistors.

Answer to question (c). The voltage drop in resistor R_1 — since its polarity is defined in the schematic in the same way it is defined in our circuit description — is as follows.

It is found by evaluating `vr1`: the calculator returns `6`, that is 6 V. For R_2, `vr2` gives **18** V. And for R_3, `vr3` gives **12** V. These are all the right answers.

Answer to question (d). Since the problem asks for the power supplied by the source, and we know that `pe` has the power consumed by it, we need to evaluate the negative of it, and we get `.216`, that is 216 mW delivered.

Answer to question (e). The power consumed by the resistors is found evaluating `pr1`, `pr2` and `pr3`. We get `.036`, `.108` and `.072`, that is 36 mW, 108 mW and 72 mW consumed, respectively.

Answer to question (f). Let's ask the calculator whether the sum of the consumed power in the resistors equals the power supplied by the source:

```
type
pr1+pr2+pr3=-pe
```

```
returns
true
```

This is the right answer, and concludes the solution to this, your first ever problem in Symbulator.

1.6 Instructive numerical examples, solved

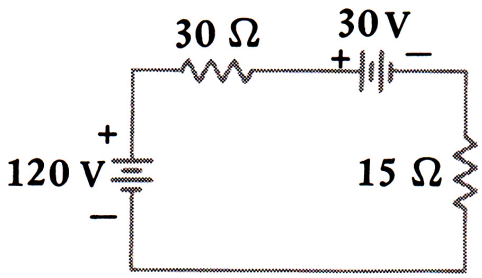
Numerical problems in DC using e and r

These practice problems are taken from several textbooks, chosen because they apply only the concepts you have learned so far. They let you practise and reinforce them before Lesson 2.

The problem below comes from Figure 1-26 (a) in Hyatt and Kemmerly's *Engineering Circuit Analysis* (5ed). Moving forward, we will refer to that textbook as **HK5**.

HK5's Figure 1-26

We are asked for the current, voltage drop and power consumed in each resistor, the power *delivered* by each voltage source, and a check that the powers in the circuit add up to zero.



Here is my solution.

I named the nodes thus: the bottom node **0**, and the top nodes, from left to right, **1**, **2** and **3**. My description of the circuit, as the argument of the DC simulation command:

type
```
s\dc("e1,1,0,120:r1,1,2,30:e2,2,3,30:r2,3,0,15")
```

When the simulation is done, you can ask the calculator for the answers you need:

- Evaluating `ir1` or `ir2` gets the current in the resistors: **2** A
- Evaluating `vr1` gets the voltage drop in the 30Ω resistor: **60** V
- Evaluating `vr2` gets the voltage drop in the 15Ω resistor: **30** V
- Evaluating `pr1` gets the power consumed in the 30Ω resistor: **120** W
- Evaluating `pr2` gets the power consumed in the 15Ω resistor: **60** W
- Evaluating `-pe1` gets the power delivered by the 120V source: **240** W
- Evaluating `-pe2` gets the power delivered by the 30V source: **-60** W. The source consumes 60W, so it delivers *negative* 60W.
- Evaluating `pr1+pr2+pe1+pe2` gets the sum of powers: **0** W. As expected.

Wasn't that easy?

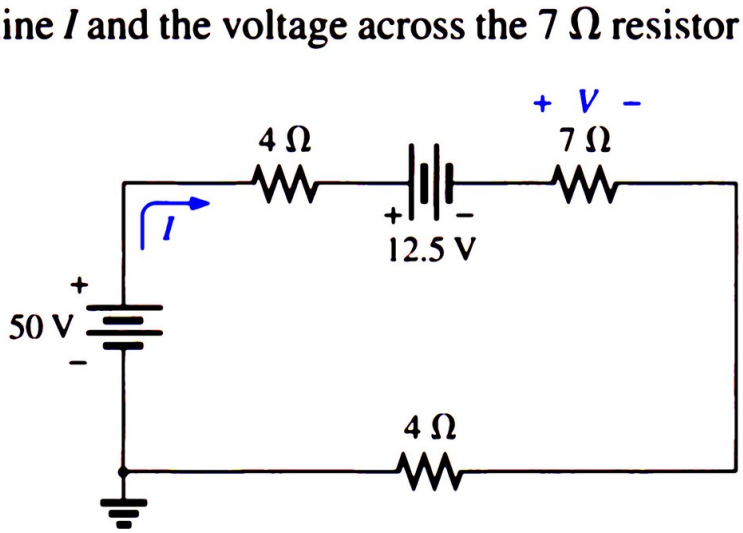
We could also have asked for all the answers with one array:

type

```
{ir1,vr1,vr2,pr1,pr2,-pe1,-pe2,pr1+pr2+pe1+pe2}
```

B11's Example 5.20

The practice problems get progressively more complicated, so you build up your *'symbulating'* skills with confidence.



I named the nodes clockwise from the ground: **0**, **1**, **2**, **3** and **4**. My circuit description, as an argument of the DC simulation command:

type

```
s\dc("e1,1,0,50:r1,1,2,4:e2,2,3,12.5:r2,3,4,7:r3,4,0,4")
```

When it's done, ask for the answers we need. Evaluating `ir1` gets the current I: **2.5** A. Evaluating `vr2` gets the voltage drop in the 7Ω resistor: **17.5** V

B11's Example 6.13

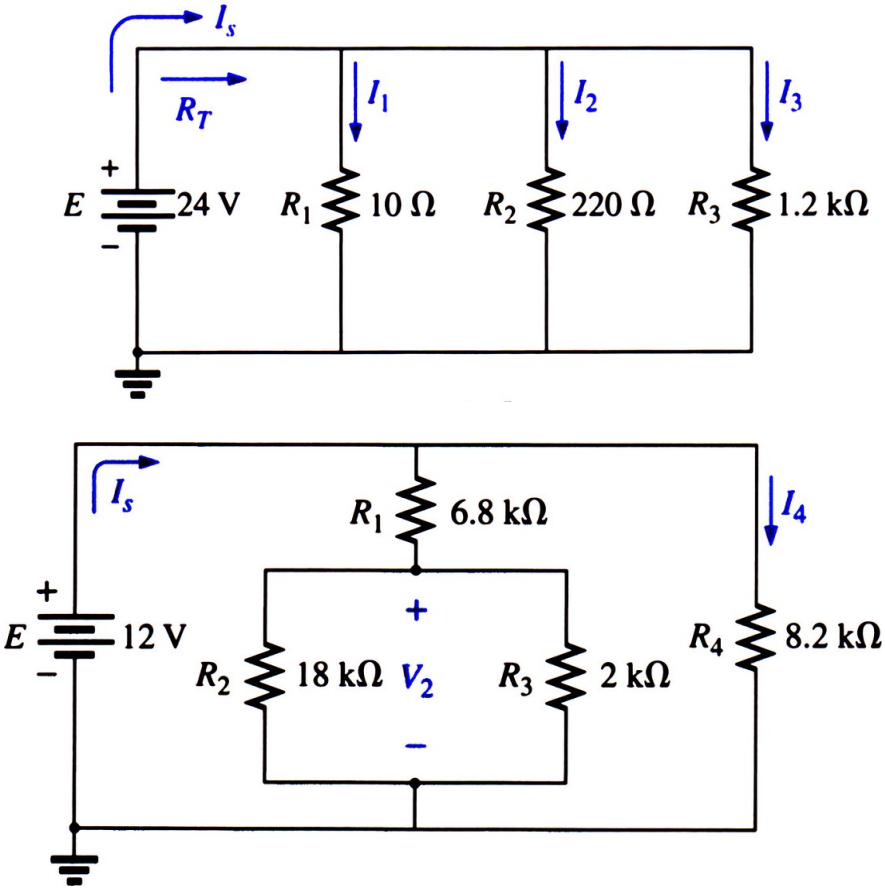
Since the ground is the bottom node, I named it **0**. I named the top node **1**.

type

```
s\dc("e,1,0,24:r1,1,0,10:r2,1,0,220:r3,1,0,1.2'k")
```

- Evaluating `re` gets the total resistance: **9.49** Ω
- Evaluating `-ie` gets us the source current: **2.53** A
- Evaluating `ir1` gets I_1: **2.4** A, `ir2` gets I_2: **109** mA, and `ir3` gets I_3: **20** mA.

a. Find the total resistance.
b. Calculate the source current.
c. Determine the current through each branch.



B11's Example 7.2

Determine I_4, I_S and V_2.

My solution: I named the top node **1**, and the other **2**:

```
type
s\dc("e,1,0,12:r1,1,2,6.8'k:r2,2,0,18'k:r3,2,0,2'k:r4,1,0,8.2'k")
```

Answers: `v2` is **2.51** V, `-ie` (i.e. I_S) is **2.86** mA and `ir4` is **1.46** mA.

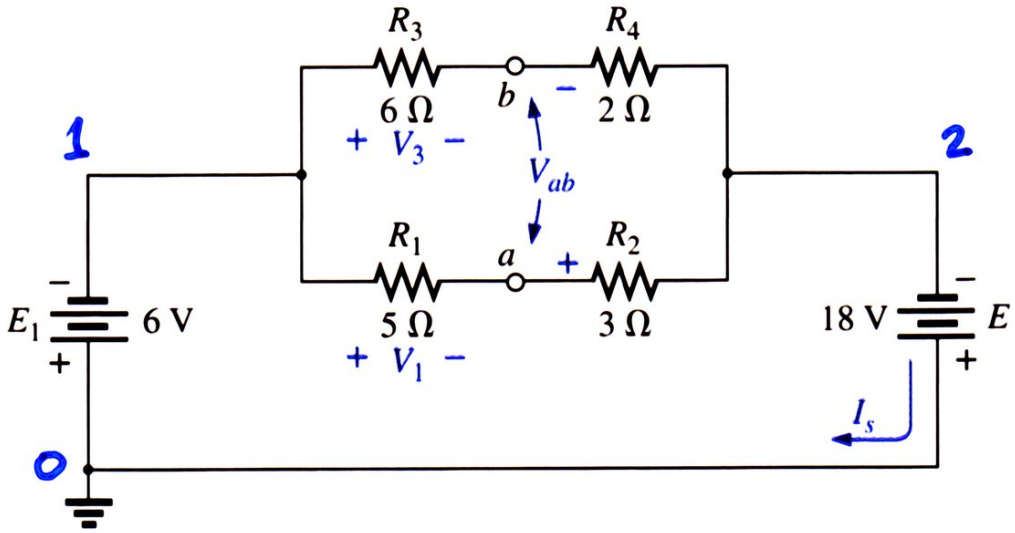
B11's Example 7.7

I have labelled the node names I used. My solution:

```
type
s\dc("e1,0,1,6:e2,0,2,18:r1,1,a,5:r2,a,2,3:r3,1,b,6:r4,b,2,2")
```

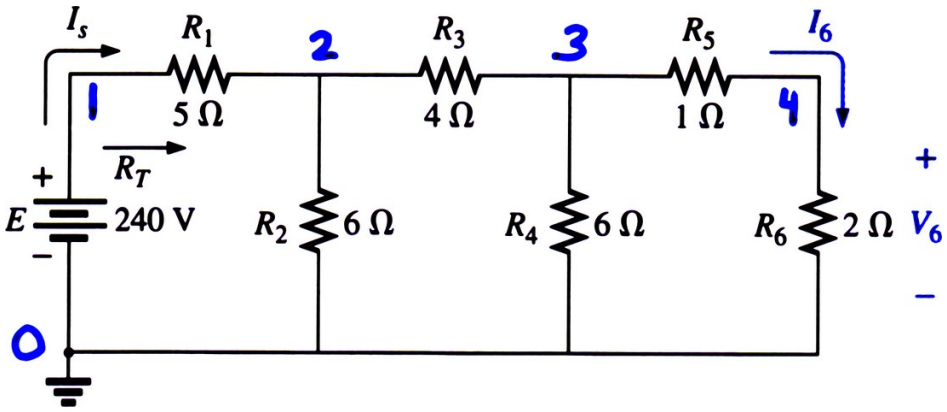
Answers: `vr1` is **7.5** V, `vr3` is **9** V. For V_{ba}, `vb-va` is **-1.5** V. For I_S, `-ie2` is **3** A.

a. Find the voltages V_1, V_3, and V_{ab} for the network
b. Calculate the source current I_s.



B11's Figure 7.32

Determine I_6 and V_6.



My solution:

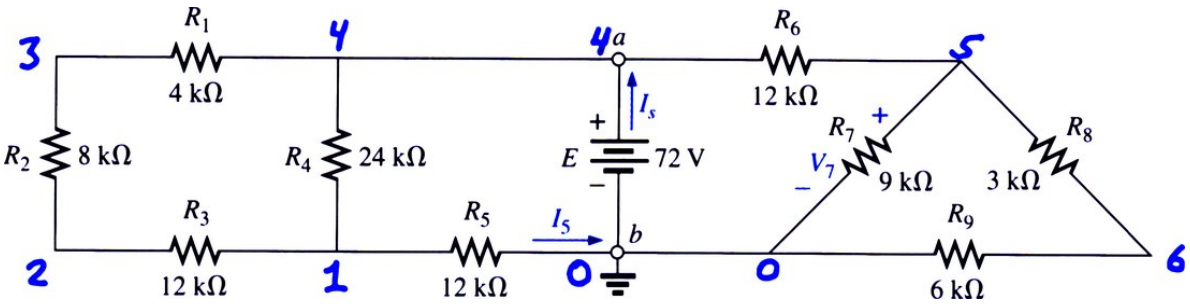
```
type
s\dc("e,1,0,240:r1,1,2,5:r2,2,0,6:r3,2,3,4:r4,3,0,6:r5,3,4,1:r6,4,0,2")
```

Answers: `ir6` is **10** A, and `vr6` is **20** V.

B11's Example 7.10

Calculate the indicated currents and voltages.

My solution:

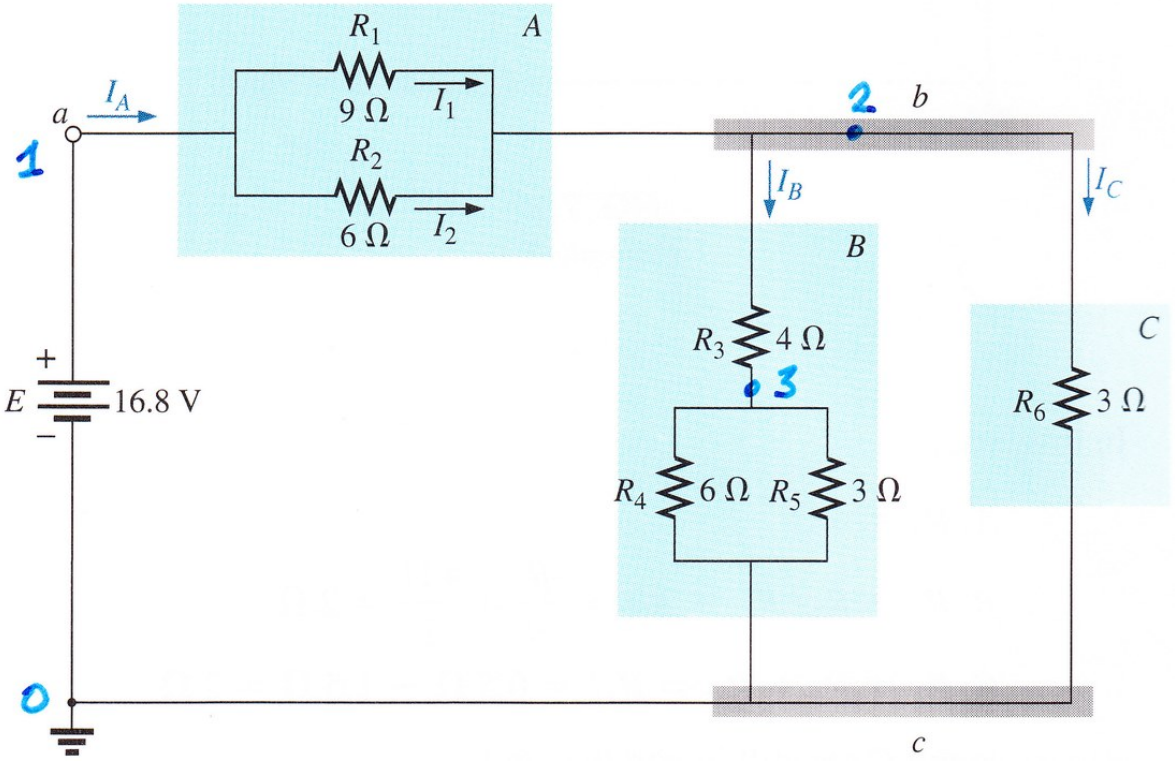


type
```
s\dc("r2,2,3,8'k:r1,3,4,4'k:r3,1,2,12'k:r4,1,4,24'k:
r5,1,0,12'k:e,4,0,72:r6,4,5,12'k:r7,5,0,9'k:r8,5,6,3'k:r9,0,6,6'k")
```

Answers: `ir5` is **3** mA, `-ie` (i.e. I_S) is **7.36** mA, and `vr7` is **19.6** V.

B11's Example 7.4

Determine the currents I_1, I_2, I_A, I_B and I_C, and the voltage drop in areas A, B and C.

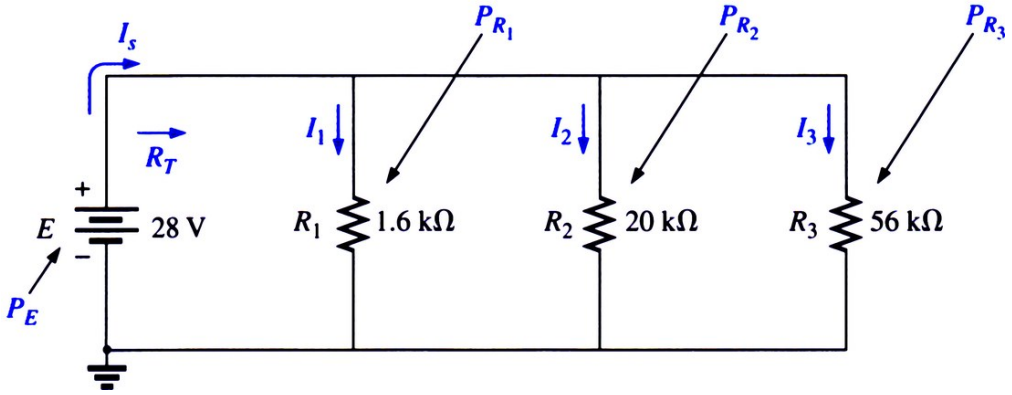


My solution:

type
```
s\dc("e,1,0,16.8:r1,1,2,9:r2,1,2,6:r3,2,3,4:r4,3,0,6:r5,3,0,3:r6,2,0,3")
```

Current I_1 is found via `ir1` = **1.2** A, I_2 via `ir2` = **1.8** A, I_A, via `-ie` = **3** A, I_B via `ir3` = **1** A and I_C via `ir6` = **2** A. The voltage drop in area A is `vr1` = **10.8** V; in both B and C it is `v2` = **6** V.

a. Determine the total resistance R_T.
b. Find the source current and the current through each resistor.
c. Calculate the power delivered by the source.



B11's Example 6.15

My solution:

```
type
s\dc("e,1,0,28:r1,1,0,1.6'k:r2,1,0,20'k:r3,1,0,56'k")
```

- Evaluating `re` gets the total resistance: **1.44** kΩ
- Evaluating `ir1` gets **17.5** mA, `ir2` gets **1.4** mA, and `ir3` gets **0.5** mA
- Evaluating `-pe` gets the power: **543** mW

These are the correct answers.

B11's Figure 7.40

Determine V_b and V_c.

My solution:

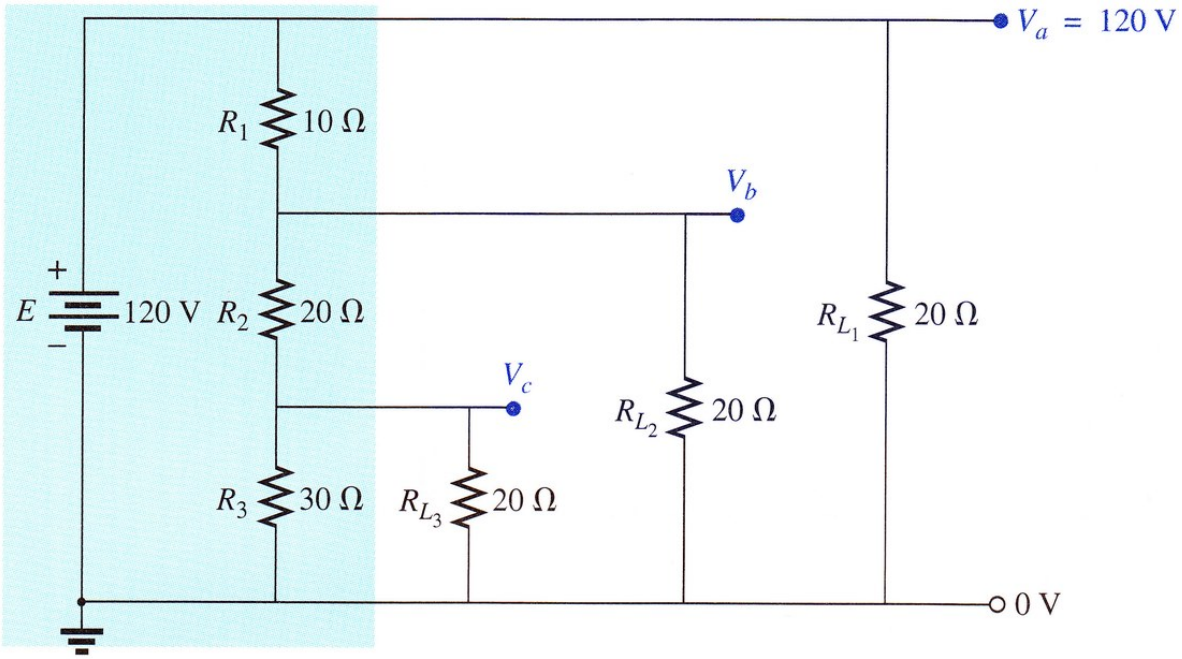
```
type
s\dc("e,a,0,120:r1,a,b,10:r2,b,c,20:r3,c,0,30:
rl1,a,0,20:rl2,b,0,20:rl3,c,0,20")
```

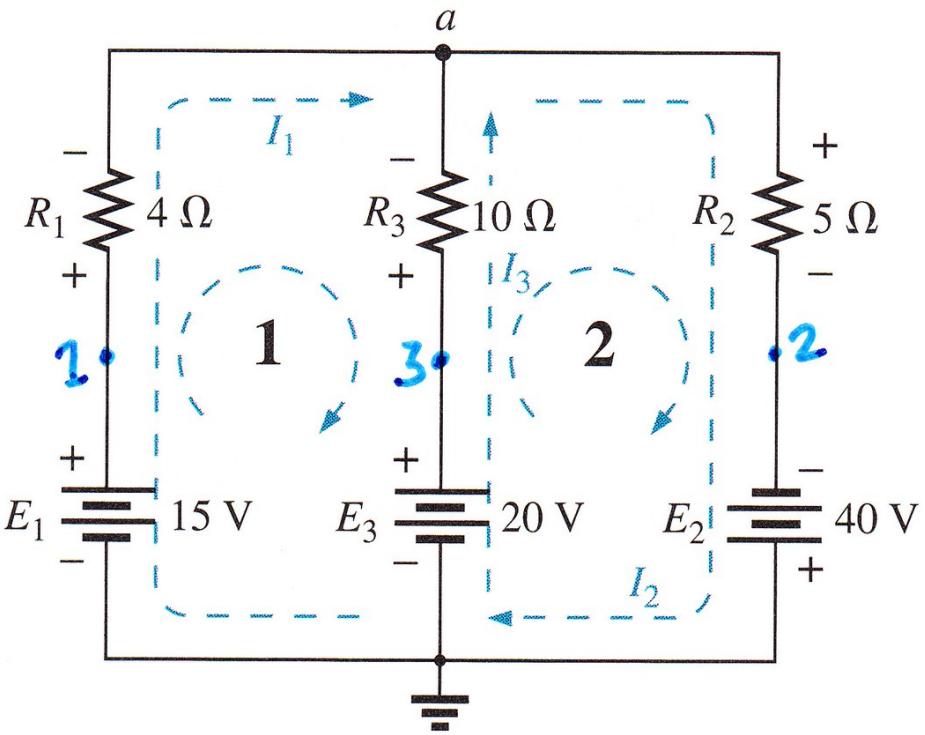
Answers: `vb` is **66.21** V, and `vc` is **24.83** V.

B11's Example 8.10

Determine the current through each resistor.

My solution:






```
type
s\dc("e1,1,0,15:r1,1,a,4:e3,3,0,20:r3,3,a,10:e2,0,2,40:r2,a,2,5")
```

Through an array, and using the `approx` command, we ask for all the three answers:

```
type
approx({ir1,ir2,ir3})
```

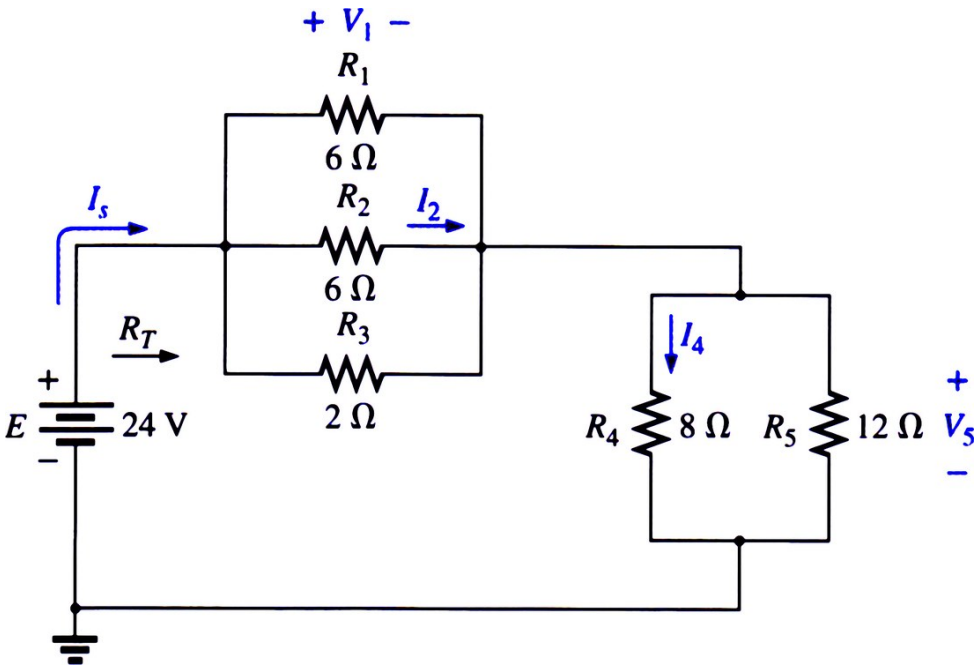
The calculator returns **{4.77,7.18,2.41}** meaning I_{R1} =**4.77**A, I_{R2} =**7.18**A and I_{R3} =**2.41**A.

These are the correct answers. We could get this answer in a single-line command:

```
type
s\dc("e1,1,0,15:r1,1,a,4:e3,3,0,20:r3,3,a,10:e2,0,2,40:r2,a,2,5"):
approx({ir1,ir2,ir3})
```

Moving forward, we will often use this single-line approach for getting our answers.

B11's Example 7.6



My solution:

```
type
s\dc("e,1,0,24:r1,1,2,6:r2,1,2,6:r3,1,2,2:r4,2,0,8:r5,2,0,12")
```

Answer: `-ie` is I_S=**4** A, `ir2` is I_2 = **.8** A, `ir4` is I_4=**2.4** A, `vr1` is V_1=**4.8** V, `vr5` is V_5=**19.2** V.

B11's Example 7.11

My solution:

```
type
s\dc("e1,a,0,20:e2,a,b,5:e3,c,0,8:r1,a,c,10:r2,b,c,4:r3,b,0,5")
```

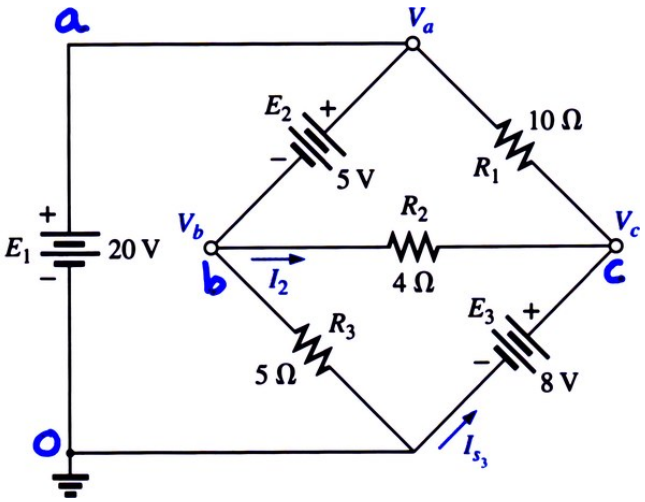
Answers: `va`=**20** V, `vb`=**15** V, `vc`=**8** V, `va-vc`= **12** V, `vb-vc`=**7** V, `ir2`=**1.75** A, I_S via `-ie3`=**-2.95** A

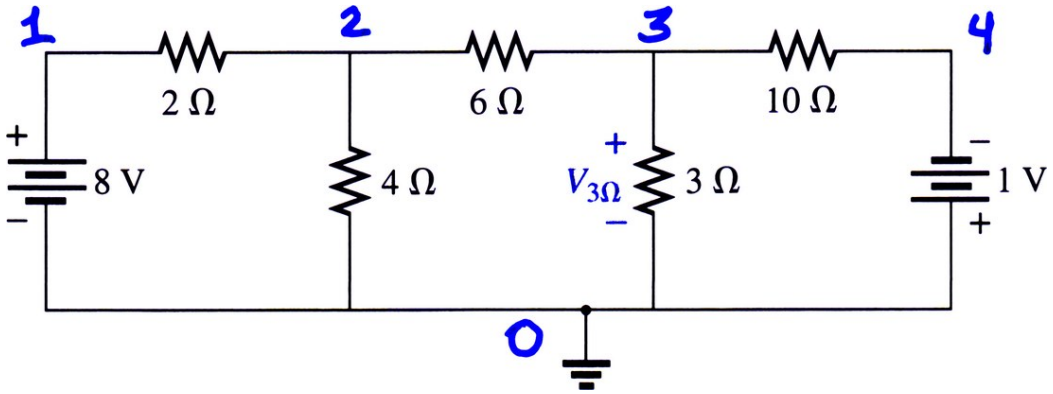
B11's Example 8.24

Find the voltage drop in the 3Ω resistor.

My solution:

a. Determine voltages V_a, V_b, and V_c.
b. Find voltages V_{ac} and V_{bc}.
c. Find current I_2.
d. Find the source current I_{s_3}.





type
```
s\dc("e8,1,0,8:r2,1,2,2:r4,2,0,4:r6,2,3,6:r3,3,0,3:r10,3,4,10:e1,0,4,1")
```

Now, evaluating **vr3** via `approx(vr3)` finds that $V_{3\Omega}$ is **1.1** V. This is correct.

B11's Example 8.18

Find the current through the 10Ω resistor in the network shown.

My solution:

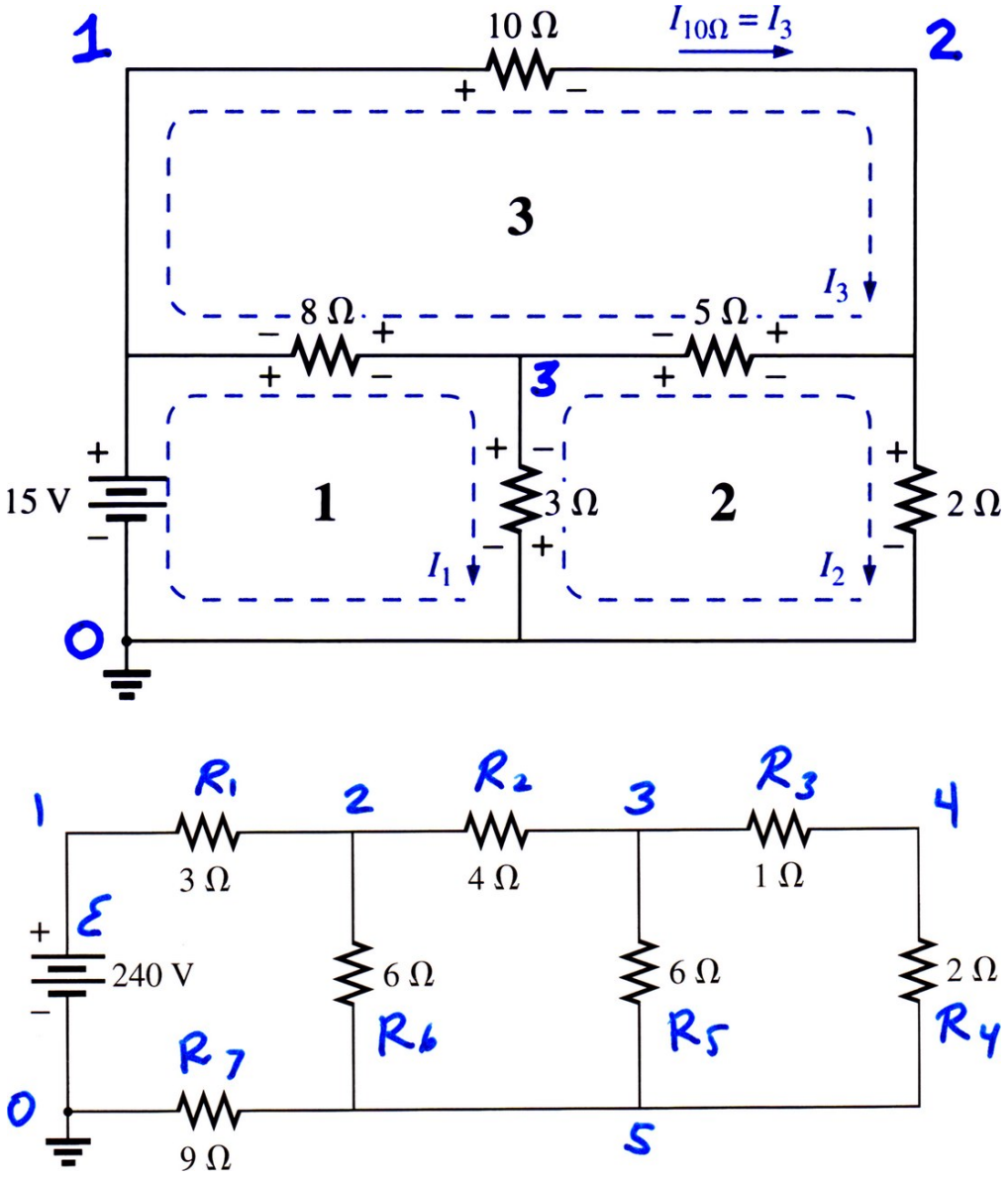
type
```
s\dc("e15,1,0,15:r10,1,2,10:r8,1,3,8:r5,3,2,5:r3,3,0,3:r2,2,0,2"):
approx(ir10)
```

We find that `ir10` = **1.22** A.

B11's Example 8.26

Find the voltage drop in the 2Ω resistor.

My solution is below:

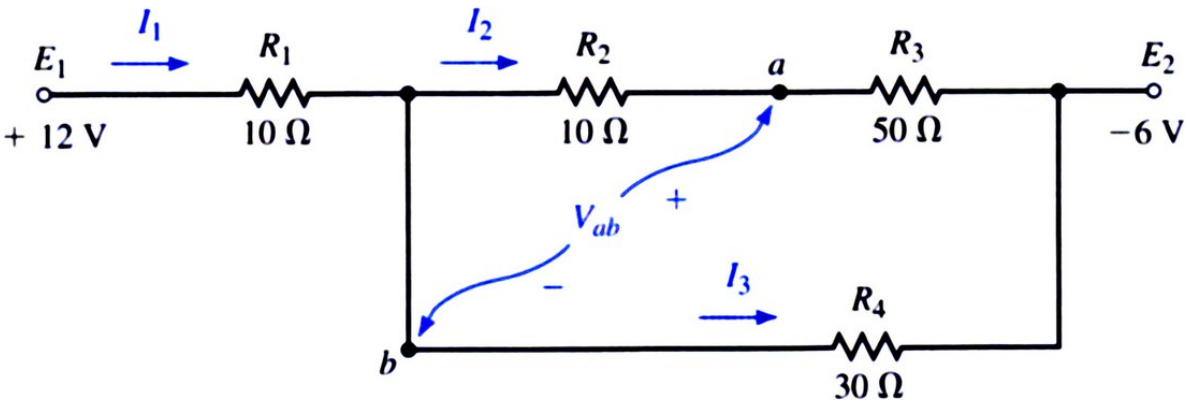



```
type
s\dc("e,1,0,240:r1,1,2,3:r2,2,3,4:r3,3,4,1:
r4,4,5,2:r5,3,5,6:r6,2,5,6:r7,5,0,9")
```

Evaluating `approx(vr4)`, we find the voltage drop in R_4 (the 2Ω resistor): it is **10.67** V

Circuits with 'hidden source'

Sometimes the schematics of circuits are presented in such a way that sources of voltage are not shown explicitly, yet their voltage is provided. These are what I call 'hidden source' problems. Below I offer two examples of these types of problems. Both are taken from the textbook *Circuit Analysis: Theory and Practice* (3ed) by Allan H. Robbins and Wilhelm C. Miller, to which from this point on we will refer as **RM3**.



RM3's Example 7-5 (Hidden source)

Find the indicated currents and voltages.

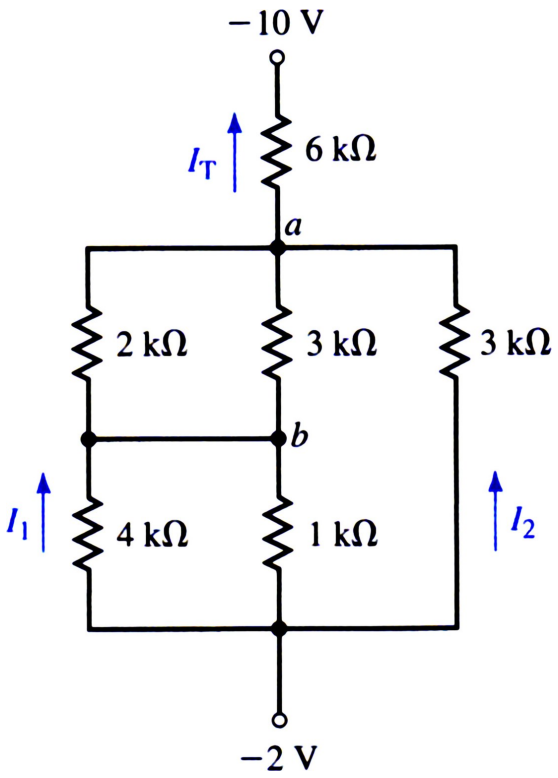
In my solution below, notice I introduced two sources, one for 12V and one for -6V:

```
type
s\dc("e1,1,0,12.:r1,1,b,10:r2,b,a,10:r3,a,2,50:
r4,b,2,30:e2,2,0,-6"):approx({ir1,ir2,ir4,va-vb})
```

The answer, **{.6,.2,.4,-2.}**, indicates I_1=**.6** A, I_2=**.2** A, I_3=**.4** A and V_{ab}=**-2** V. This is correct.

RM3's Figure 7-16 (Hidden source)

Find the total circuit resistance, and the indicated currents and voltages.



My solution:

This is how I named the nodes: the top, **1**; the bottom, **2**, and **a** and **b** as in the figure. In my solution below, notice I introduced two sources, one for -10V and one for -2V:

```
type
s\dc("r1,1,b,4'k:r2,1,a,3'k:r3,b,a,2'k:r4,b,a,3'k:r5,1,b,1'k:
rt,a,2,6'k:e1,1,0,-2:e2,2,0,-10"): approx({(v1-v2)/irt,irt,ir1,ir2,
va-vb})
```

The answer we get indicates that the equivalent resistance, given by

$$\frac{v_1 - v_2}{I_T}$$

is **7.2** kΩ, and that I_T=**1.11** mA, I_1=**.133** mA, I_2=**.444** mA and V_{ab}=**-0.8** V. This is correct.

LESSON 2

Symbolic circuits and expert mode

Learn how to solve different types of *symbolic* circuits in Symbulator. Use the solve command on Symbulator. And explore **ex**, the powerful *expert mode*, for the ambitious user.

Here you will learn about the types of symbolic circuits and how to solve them in Symbulator, using the solve command when needed, and — for the more adventurous — the powerful *expert mode*.

2.1 About symbolic problems

The problem we saw in Lesson 1 (page 8) was a *numerical problem* because it involved a numerical circuit: from the start, we knew the values of all its elements. None were unknown.

A circuit with one or more elements whose value we do not know is a **symbolic circuit**. A *symbolic problem* is one that involves a symbolic circuit. It gives me great joy when I encounter a symbolic problem, because the ability to simulate symbolic circuits is what sets Symbulator apart from other programs.

> ⓘ NOTE
>
> **Variables are case-sensitive; names are not**
>
> Element and node names ignore case — **R1** and **r1** are the same resistor, and so is any reference built from a name: `2*VR1`, `2*vr1` and `2*v_r1` all mean r1's voltage drop. A variable that names nothing in the circuit is different: `e,1,2,c` and `e,1,2,C` are two *different* symbolic sources, and telling Symbulator that `c = 5` says nothing about `C`. Pick one spelling for each unknown and keep it.

I find it useful to distinguish between two types of symbolic problems:

- **Purely symbolic problems** use symbolic circuits whose answers are also symbolic, given as expressions of unknown variables. Solving them in Symbulator is as straightforward as solving numerical problems.
- **Numerical-from-symbolic problems** start with a symbolic circuit but ask for numerical answers. That is possible when the problem gives us more information about the circuit.

Numerical-from-symbolic problems can be solved in two ways. If they are simple — one or two unknown values, one or two numerical answers wanted — it is quicker to simulate the circuit symbolically and then solve for the numerical answers with the **solve** command of the calculator. If they are not simple, it is easier to use **expert mode**.

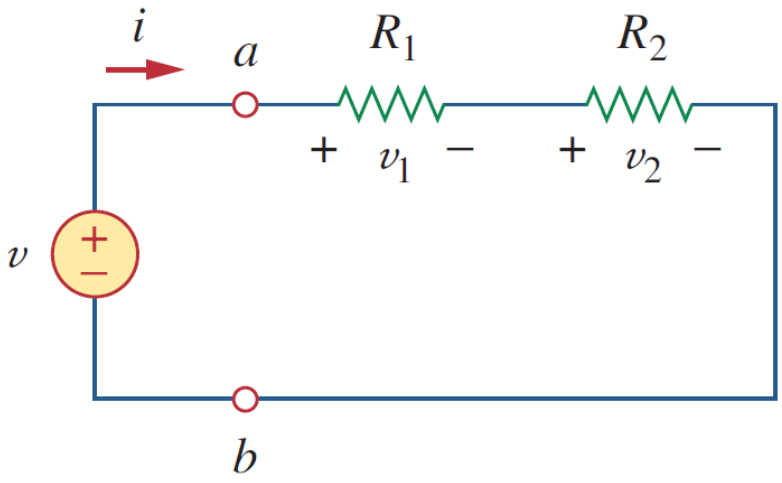
Let's see an example of each.

Solving a purely symbolic problem

Since Symbulator is natively a symbolic simulator, there is nothing special about solving a purely symbolic problem. The only limitation comes from the computing capacity of the calculator: while small circuits can be solved relatively fast, larger circuits may be too slow to solve and may cause a memory error.

Let's see a simple purely symbolic problem, from *Fundamentals of Electric Circuits* by Alexander and Sadiku (5th edition), from now on referred to as **AS5**.

AS5's Figure 2.29



AS5's Figure 2.29

SOLUTION

All the values in this circuit are variables: no element value is known. This is an example of a symbolic circuit. Imagine you are asked to find symbolic expressions for the current ***i***, the voltage drop in R_2, and the power consumed by R_1 and R_2. Since we are being asked to find symbolic answers from a symbolic circuit, this is a purely symbolic problem.

This is how I would describe this circuit. First, the nodes. The node called **a** in the schematic stays node **a**. The node called **b** is my ground, so it becomes node **0**. The node between the two resistors I call **c**.

Then the elements. I name the source **ev** and the resistors **r1** and **r2**, and give them variables for values: **v** for the source, **r1** and **r2** for the resistors. It is no problem that a symbolic value carries the same name as its element.

```
type
"ev,a,0,v:r1,a,c,r1:r2,c,0,r2"→cir
s\dc(cir)
```

This simulation took a couple of seconds in my calculator. To find the current i, we ask for `ir1` or `ir2`. To find the voltage drop on r2, we ask for `vr2`.

Both are expressions rather than numbers:

$$i_{r1} = \frac{v}{r_1 + r_2} \quad \text{and} \quad v_{r2} = \frac{r_2\, v}{r_1 + r_2}$$

which is the voltage divider you would have written by hand.

To find the power consumed in r2, we ask for `pr2`. To find the power delivered by the source, we ask for the negative of `pev`.

$$p_{r2} = \frac{r_2\, v^2}{(r_1 + r_2)^2}$$

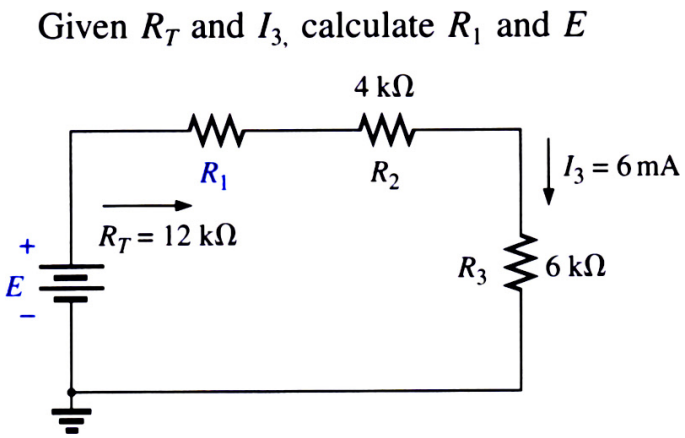
$$-p_{ev} = \frac{v^2}{r_1 + r_2}$$

It is this ability to simulate symbolically, as if it were nothing, that puts Symbulator in a league of its own. Symbolic answers from a symbolic circuit take no extra effort.

2.2 Numerical from symbolic, with solve

Getting numerical answers from a symbolic circuit, on the other hand, takes an extra step: the solve command, as in this next example, or expert mode, as in the one after.

B11's Example 5.6, with solve



B11's Example 5.6

SOLUTION

This is a very nice numerical-from-symbolic problem. We can solve it into numbers because, although it hides two values from us (the source **E** and the resistor R_1), it gives us in exchange two answers (the equivalent resistance R_T and the current I_3) to solve for them with.

Since this circuit is structurally identical to **B11's Example 5.7 (page 12)** solved in Lesson 1 (page 8), we will use the same names for the nodes. The description is identical except for the values. Symbulator accepts numbers, variables or even algebraic expressions as values. Here I use **e** for source **e**, and **r1** for resistor **r1**.

⚠ WARNING

Clean your document

Make sure the variables you will use as symbolic values are empty — that they do not exist in the current document. Empty the whole document, or delete just those variables: `DelVar e,r1`

```
type
"e,1,0,e:r1,1,2,r1:r2,2,3,4'k:r3,3,0,6'k"→cir
s\dc(cir)
```

A moment later Symbulator is done. From its symbolic answers and the answers the problem gives us, we write two equations and solve them for the two unknowns.

The problem says that I_3 is 6 mA. In Symbulator, I_3 is `ir3`, the current through resistor **r3**. If you evaluate it, you will see it produces an algebraic expression in terms of the two unknowns, **e** and **r1**. This is what we call a symbolic answer. The problem also says that R_T is 12 kΩ; the equivalent resistance as seen by the source **e** is given by `re`, which when evaluated gives another algebraic expression in terms of r1. We can then write two new equations and solve them for e and r1:

```
type
solve(re=12000 and ir3=.006,{e,r1})
```

ⓘ NOTE

The apostrophe shorthand does not work here

You cannot use `'k` here as a shorthand for kilo, or any other SI prefix: it works only in the values in the circuit description.

An instant later we get the answers: **e** = **72** V and **r1** = **2000** Ω. These are the right answers. Not many other circuit simulators allow this flexibility.

2.3 Numerical from symbolic, using ex

Symbulator's true strength is seen in numerical-from-symbolic problems like the one we solved above, when we use its expert mode of simulation. Expert mode cracks these problems open even faster, and gives fully numerical answers whenever the problem provides as many answers as it hides values. Learning it pays off handsomely in power and speed.

Let's solve the same circuit again, this time using the expert mode's program **ex**.

B11's Example 5.6, using ex

SOLUTION

We will use the same circuit description as before, with a single change: we will use rx for the value of resistor R_1, instead of the r1 value we used before. Like this:

type
```
"e,1,0,e:r1,1,2,rx:r2,2,3,4'k:r3,3,0,6'k"→cir
```

We use rx instead of r1 because expert mode saves the value it finds into that variable, and r1 — like the other r# variables — cannot store anything: trying to results in an error.

To run an expert mode simulation, type this:

type
```
s\ex(cir)
```

When prompted, select DC and press Enter. Now you will see a prompt asking you to add equations, variables and conditions. You may recall from algebra that solving a set of equations into numbers needs as many equations as unknowns. The statement of the problem gives us the information we need to write the two additional equations.

In "Add equation(s)" type:

type
```
re=12'k and ir3=6'm
```

In "Add unknown(s)" type:

type
```
e,rx
```

Now we have six variables and six equations. Press Enter and wait a few seconds. A few other dialogs appear; in this and every other expert example in this volume, just press OK without changing anything. When Symbulator says "Done", go ahead and retrieve the answers: `rx` gives **2000** and `e` gives **72**. Both are right.

The speed advantage is not obvious in so simple a problem, but the idea is: you halt the simulation in mid-air and give Symbulator extra information. Had the circuit been larger, the saving would be clear.

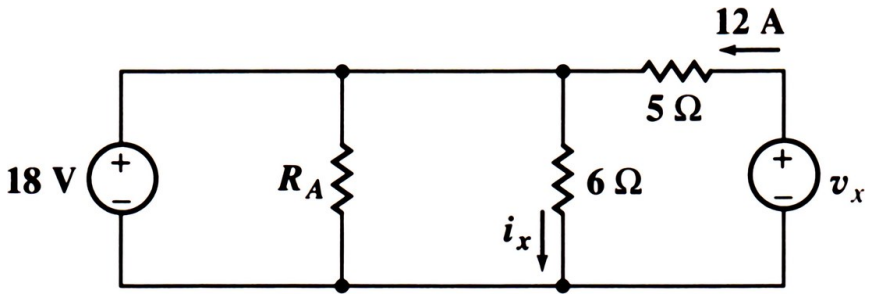
2.4 Instructive symbolic examples, solved

Numeric-from-symbolic, with solve

HK5's Figure 1-24a (*solve*)

Determine i_x and v_x in the following circuit:

I named the nodes thus: bottom is **0**, top left is **1**, top right is **2**. I named the elements according to their value: this facilitates remembering who's who in the circuit. I also defined the resistors' nodes in the direction of the current indicated in the diagram.




```
type
s\dc("e18,1,0,18:ra,1,0,ra:r6,1,0,6:r5,2,1,5:evx,2,0,vx")
```

Explore the answers. Since **ir5** (which we know is 12 A) is in terms of v_x, we can find v_x:

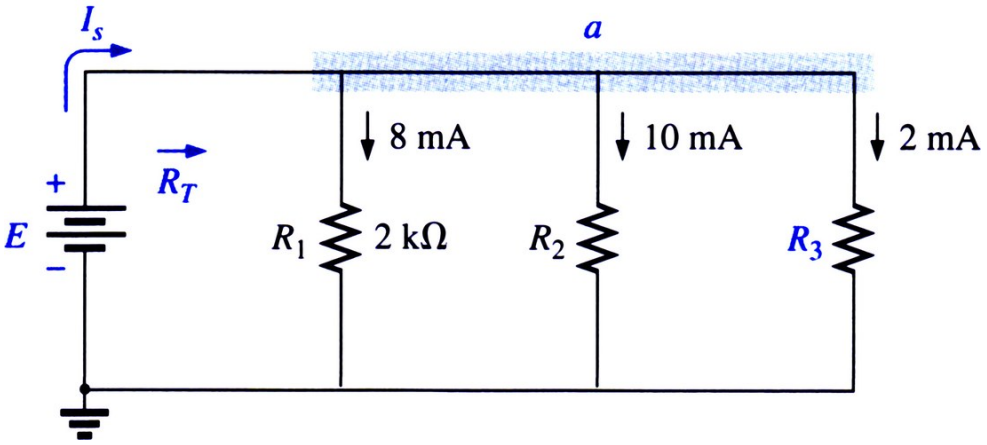
```
type
solve(ir5=12,vx)
```

We get that v_x is **78** V, which is correct. Evaluating `ir6` we find that i_x is **3** A. The fact that we can find numerical answers in this problem can be quite puzzling until one realizes that ignoring the value of R_A doesn't matter: due to the circuit's structure, it is not needed to answer the two questions we have been asked.

Numeric-from-symbolic, with Expert

B11's Example 6.19 (Expert)

a. Determine the source current I_s.
b. Find the source voltage E.



This problem, having three unknown element values and three known answers, is a perfect candidate for the expert mode. Below is my circuit description.

```
type
s\ex("e,1,0,e:r1,1,0,2'k:r2,1,0,r2:r3,1,0,r3")
```

Select DC. Add these three equations:

```
type
ir1=8'm and ir2=10'm and ir3=2'm
```

Add these unknowns:

```
type
e,r2,r3
```

Press Enter and Enter. When Symbulator is *Done*, find I_S and E by asking:

```
type
approx(-ie1)
```

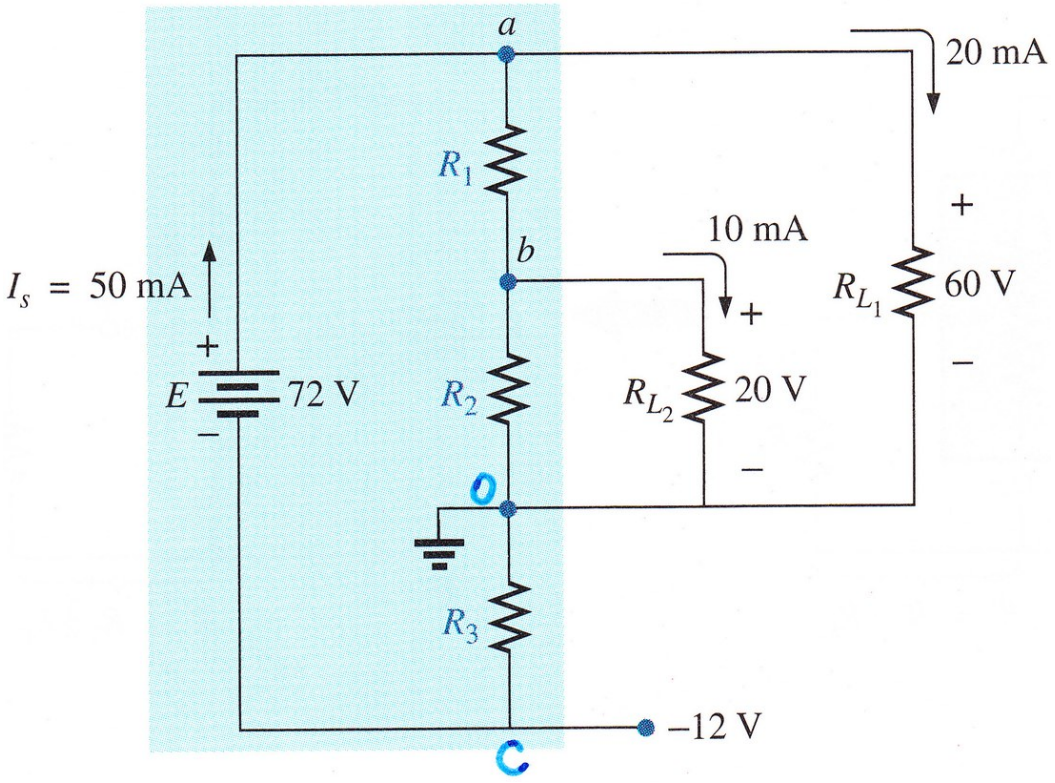
We get that I_S is **0.02** A.

```
type
e
```

We get that E is **16** V. These are correct.

B11's Example 7.12 (Expert)

Determine R_1, R_2 and R_3 for the voltage divider supply. Can 2W resistors be used?



This problem is also perfect for the expert mode, because: (a) the target is to obtain numerical values, and (b) we have N unknown element values, and in turn we are given N numerical answers. Here is how I solved it:

type
```
s\ex("e,a,c,72:r1,a,b,r1:r2,b,0,r2:r3,0,c,r3:rl1,a,0,rl1:rl2,b,0,rl2")
```

Choose DC. Add these five equations:

type
```
irl1=20'm and vrl1=60 and irl2=10'm and vrl2=20 and -ie=50'm
```

Add these five variables to the list of first level variables:

type
```
r1,r2,r3,rl1,rl2
```

Press Enter and Enter. When Done, evaluate these variables to find the answers: `r1` = **1333.3** Ω, `r2` = **1000** Ω, `r3` = **240** Ω. These are correct.

Then we ask for the powers consumed in the resistors, and find that:

`pr1` is **1.2** W, `pr2` is **0.4** W, and `pr3` is **0.6** W

Since all are smaller than 2W, it is possible to use 2W resistors in the design.

LESSON 3

Current sources, conductances and dependent sources

Learn how to describe a *current source* using the **j** element. Simplify *parallel resistors* using the **pr** tool and its shorthand. Learn to describe *conductances* and *dependent sources*.

In this lesson you will learn how to describe a *current source* with the **j** element, and a trick to simplify *parallel resistors* with the **pr** tool or its shorthand. You will also learn to describe *conductance* and *dependent sources* using elements you already know.

3.1 How to describe a current source

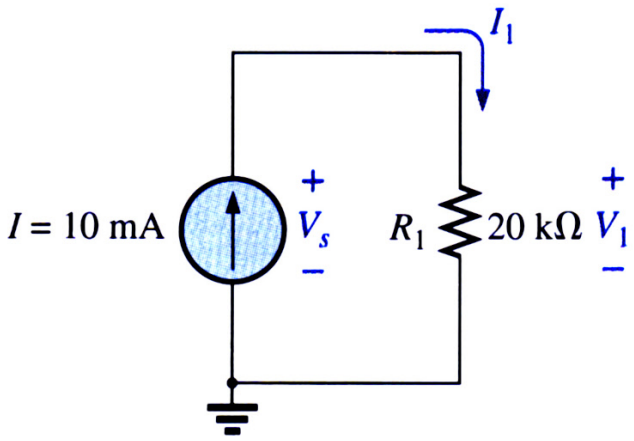
> ⓘ NOTE
>
> **Describing current sources**
>
> In Symbulator, an ideal current source is described with four pieces of information, separated by commas: a unique name (which must start with the letter **j**), the names of its first and second node, and its value in amperes (A). The value is the current flowing through the source from the first node towards the second: how much current leaves by the second node, and equally how much enters by the first. For example, an ideal current source called **j1**, between nodes **0** and **3**, carrying **5** A from the first node to the second, is described: `j1,0,3,5`

You can use SI prefixes here as well. The value of a current source is often given in milliamps; Symbulator will interpret any `'m` given in the value as a division by a thousand.

Answers for a current source

For each current source you get the same answers as for a voltage source, with the same polarity conventions: the voltage drop in it, the current through it, the power consumed by it (for the delivered power, ask for the negative), and the equivalent resistance of the rest of the circuit as seen by that source.

Let's see an example.



B11's Example 8.1

B11's Example 8.1

Given the circuit, determine the current and voltage drop in R_1.

SOLUTION

We ask Symbulator to run a DC simulation of the circuit described between quotations:

```
type
s\dc("j,0,1,10'm:r1,1,0,20'k")
```

We ask for the values of two variables, `ir1` and `vr1`. We get **.01** and **200.**, meaning a 10 mA current and a 200 V voltage drop.

3.2 What about conductances?

Conductances are really resistors by another name. So we describe them as resistors, using the element **r**, and give as the value the inverse of the conductance: one divided by the conductance is the resistor in Ω.

The next example is from *Elementary Linear Circuit Analysis* (2ed) by Leonard S. Bobrow, which from here on I call **Bo2**.

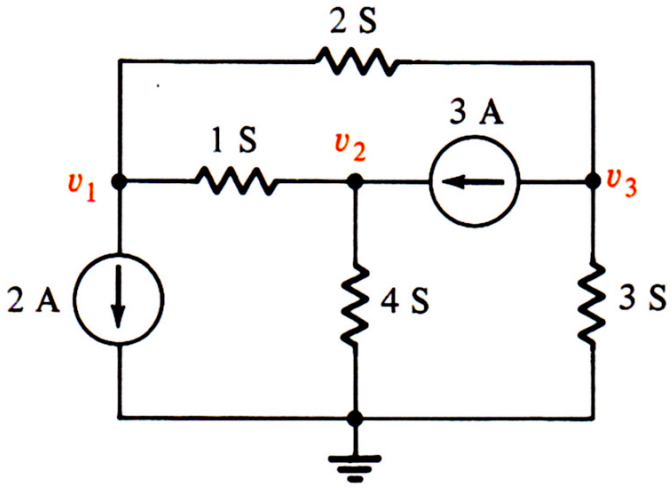
Bo2's Example 2.2

Given the circuit, determine the voltages in the nodes.

SOLUTION

As I explained, all conductances are simulated as resistors: the 4 siemens conductance becomes a 1/4 resistor, and so on. My description:

```
type
s\dc("j10,1,0,2:r12,1,2,1:r20,2,0,1/4:r30,3,0,1/3:r13,1,3,1/2:j32,3,2,3")
```



Bo2's Example 2.2

We ask for these answers:

```
type
approx({v1,v2,v3})
```

```
returns
{-1.3, .34, -1.12}
```

That indicates **v1** = **–1.3** V, **v2** = **.34** V, **v3** = **–1.12** V. This is correct.

3.3 How to reduce parallel resistors

Symbulator can reduce resistors connected in parallel to their equivalent. The function is called **pr**, and you can use it on its own or inside a circuit description. There is a shorthand too, to save you typing.

Using s\pr({r,r,r...})

When you do not need the current through, or the power consumed by, each resistor, you can reduce any number of parallel resistors to one equivalent value: invoke **s\pr** with a comma-separated list of resistance values.

For example, to reduce three resistors in parallel with values of 10 Ω, 20 Ω and 30 Ω:

```
type
s\pr({10,20,30})
```

You get 60/11 if you evaluate exactly, or 5.45 approximately.

You can also give symbolic values. For example, to reduce four resistors in parallel with values **r1**, **r2**, **r3** and **r4**:

type
```
s\pr({r1,r2,r3,r4})
```

You get the answer as a symbolic expression.

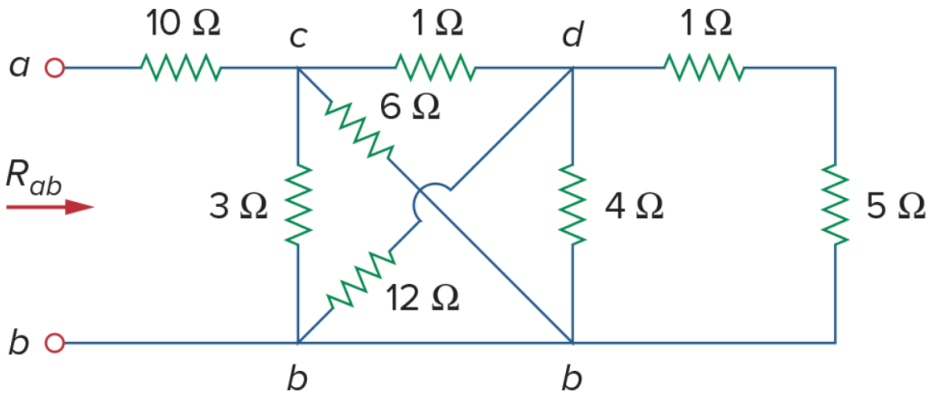
You can use the s\pr tool inside a circuit description or on its own, on the entry line. That is useful because a problem which is really just a resistor reduction may need no simulation at all.

Reducing series and parallel combos

Resistors in series reduce by simple addition, whenever we do not need each resistor's own answers — the voltage drop or power in each, or the voltage in the node between them. The current through series resistors is the same, so an equivalent still gives you the current.

AS7's Example 2.10

Find the equivalent resistance.



AS7's Example 2.10

SOLUTION

We don't need to run a simulation to find this answer. We can find it by typing this on the entry line:

type
```
10+s\pr({3,6,1+s\pr({12,4,1+5})})
```

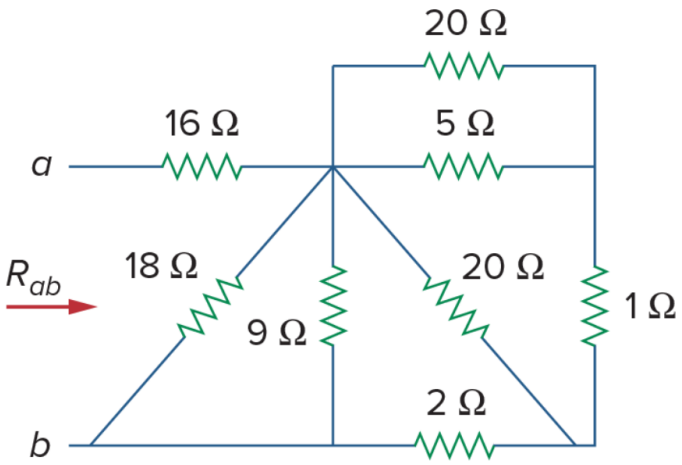
We evaluate approximately and get 11.2 Ω. This is correct.

AS7's Practice Problem 2.10

Find the equivalent resistance.

SOLUTION

Again, no simulation is needed:



AS7's Practice Problem 2.10

```
type
16+s\pr({18,9,2+s\pr({20,1+s\pr({5,20})})})
```

We evaluate and get 19 Ω. This is correct.

The [r,r,r...] shorthand for descriptions

Typing `s\pr({})` while describing a circuit is not very practical, so there is a shorthand: Symbulator reads any values inside square brackets, such as `[10,20,30]` or `[r1,r2,r3,r4]`, as input to s\pr. It works only inside the circuit description passed to Symbulator.

When to reduce resistors

A simulation where it makes sense to use **pr** is **B11's Example 7.4 (page 20)**, which you saw in the practice problems of Lesson 1 (page 8). It makes sense to reduce R_4 and R_5 to an equivalent resistor, since we do not need to know their individual currents or power use:

```
type
s\dc("e,1,0,16.8:r1,1,2,9:r2,1,2,6:r3,2,3,4:re,3,0,[6,3]:r6,2,0,3")
```

An example where using **pr** makes no sense is **B11's Example 6.13 (page 17)**, because you need to know the current through each resistor.

Finally, a simulation where you can reduce part of the resistors is **B11's Example 6.22 (page 50)**. We must leave R_1 alone, because we need the current through it, but we can reduce R_2 and R_3:

```
type
s\dc("jt,0,1,12'm:r1,1,0,1'k:re,1,0,[10'k,22'k]")
```

Moving forward, we will use the `pr({,,})` tool, or its shorthand `[,,]`, in circuit descriptions whenever we feel it is appropriate.

3.4 How to describe dependent sources

One of my favourite scenes in cinema comes from *The Dark Knight*: the Joker, played masterfully by Heath Ledger, is rolling on the floor of a Gotham City prison, taking a bare-knuckle beating from an ever-more-frustrated Batman. Master of the situation and laughing hysterically, the Joker says: *"You have nothing! Nothing to threaten me with!"*

Even though the movie had not been made yet, I remember feeling something similar — if less hysterical — back in 1999, when I realised that a 100% symbolic implementation let me make any element's value depend on any answer of the circuit. I could simulate voltage or current sources dependent on any voltage, current or combination of them as easily as a 12 V source.

Here is what you need to know for simulating dependent sources in Symbulator: nothing. There is nothing special to it at all. Just write the value as a function of the circuit's answers, using the variables you know by now, and run the simulation like it's nobody's business. For example:

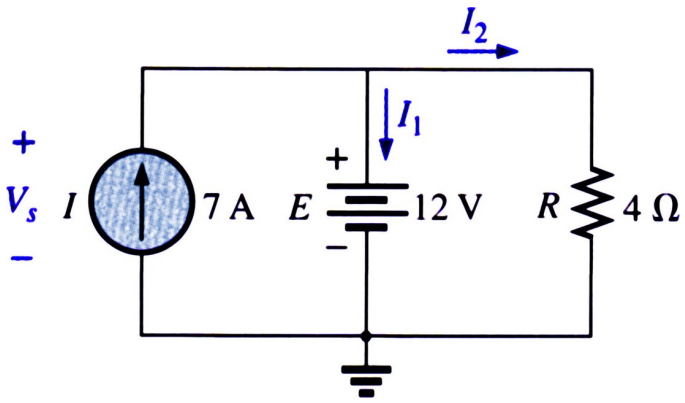
- if the source depends on the current through a resistor called **r1**, you define its value as `ir1`
- if the source depends on the voltage drop in a resistor called **r2**, you define its value as `vr2`
- if the source depends on the current through a short called **s3**, you define its value as `is3`
- if the source depends on the difference between the voltage of two nodes a and b, you define its value as `va-vb`

With Symbulator, instead of fearing them, you will laugh in the face of dependent sources, thinking: *"You have nothing!"* Booyah!

> TIP
>
> **Dependent sources are just sources**
>
> For Symbulator, dependent sources are just sources and need no special notation. State the expression that describes the value, using variables Symbulator knows and minding their polarity, and you are off to the races.

3.5 Instructive solved examples

Circuits with E, J and R



B11's Example 8.2

Determine the values of V_S, I_1 and I_2.

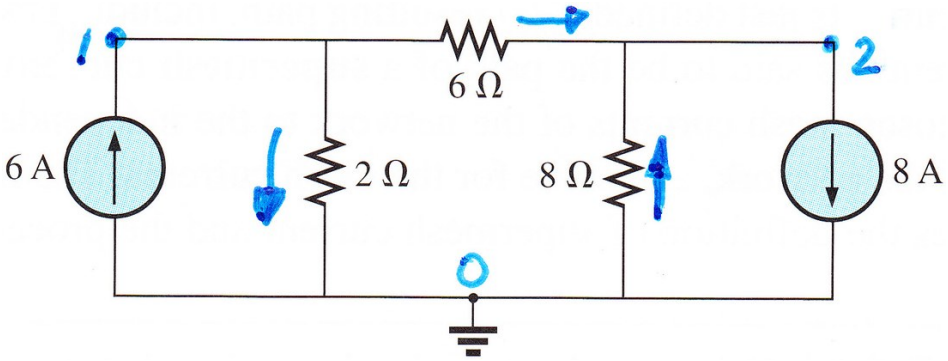
In the line below we join two commands with a colon. The first runs a DC simulation of the circuit described in the string; the second asks the calculator for three variables that answer the questions.

```
type
s\dc("j,0,1,7:e,1,0,12:r1,1,0,4"):
{v1,ie,ir1}
```

The calculator returns **{12,4,3}**, meaning V_S is **12**V, I_1 is **4**A and I_2 is **3**A. These are the correct answers. We will keep using the single-line instruction.

B11's Example 8.15

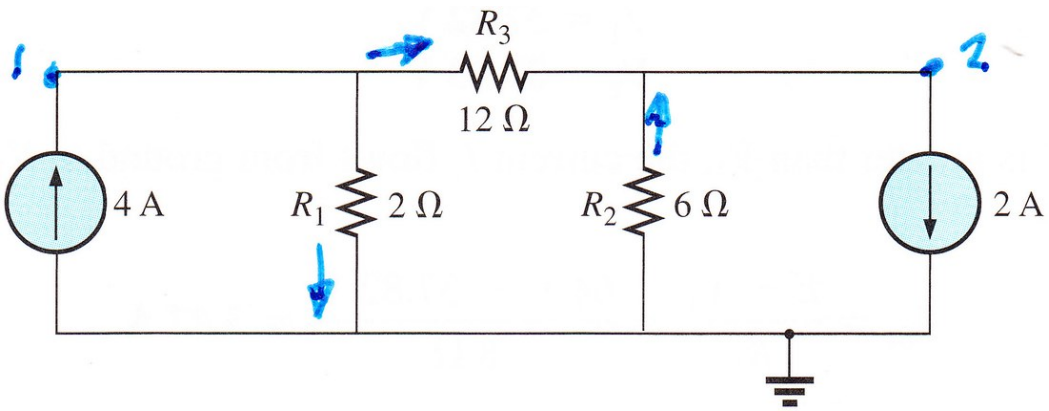
Determine the current through each resistor. My solution, with the direction marked in blue:




```
type
s\dc("j6,0,1,6:r2,1,0,2:r6,1,2,6:r8,0,2,8:j8,2,0,8"):
approx({ir2,ir6,ir8})
```

We get these answers: **{1.25,4.75,3.25}**. So I_{R2} is **1.25**A, I_{R6} is **4.75**A, and I_{R8} is **3.25**A.

B11's Example 8.21



Determine the voltage in each node and the current through each resistor.

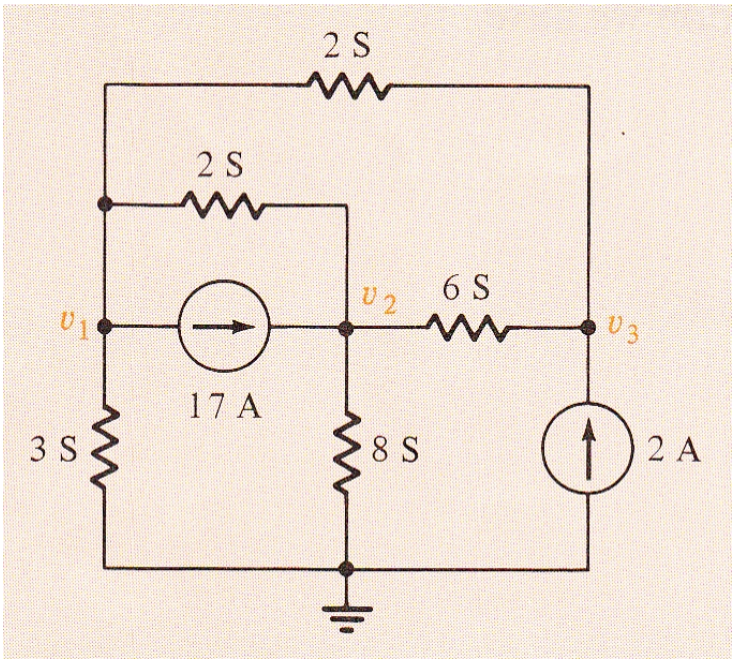
My solution:

```
type
s\dc("j1,0,1,4:r1,1,0,2:r3,1,2,12:r2,0,2,6:j2,2,0,2"):
{v1,v2,ir1,ir2,ir3}
```

We get the following answers: **{6,-6,3,1,1}**. So V_1=**6**V, V_2=**-6**V, I_{R1}=**3**A, and I_{R2}=I_{R3}=**1**A.

Bo2's Drill Exercise 2.2 (Conductances)

Given the circuit, determine the voltages in the nodes.



Below my solution:

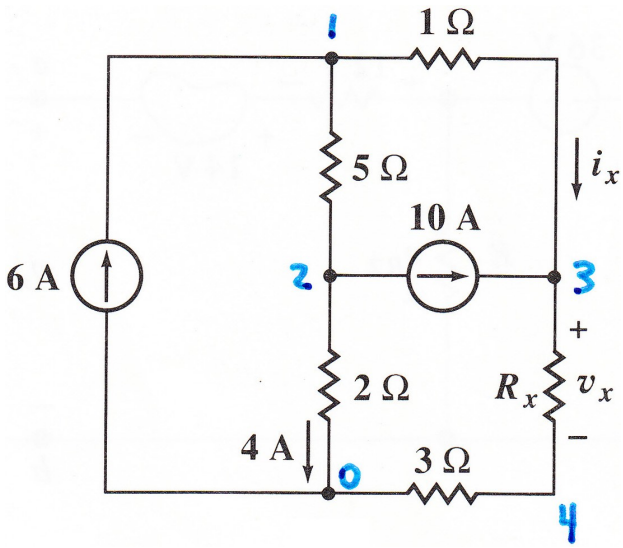
type

```
s\dc("r10,1,0,1/3:r12,1,2,1/2:r13,1,3,1/2:r23,2,3,1/6:
r20,2,0,1/8:j12,1,2,17:j03,0,3,2"):approx({v1,v2,v3})
```

The answer, **{-2.,1.,.5}**, indicates v_1=**-2**V, v_2=**1**V, v_3=**0.5**V. This is correct.

HK5's Figure 1-24b (Expert)

Determine i_x and v_x in the following circuit.



My solution:

type

```
s\ex("j6,0,1,6:r5,1,2,5:r2,2,0,2:r1,1,3,1:
r3,0,4,3:j10,2,3,10:rx,3,4,rx"):{ir1,vrx}
```

Select DC. Add equation `ir2=4`. Add unknown `rx`. Run the simulation. The answer, **{-8,80}**, means that i_x is **-8**A and that v_x is **80**V.

HK5's Example 2.2 (Conductances)

Determine the voltages in the nodes. My solution:

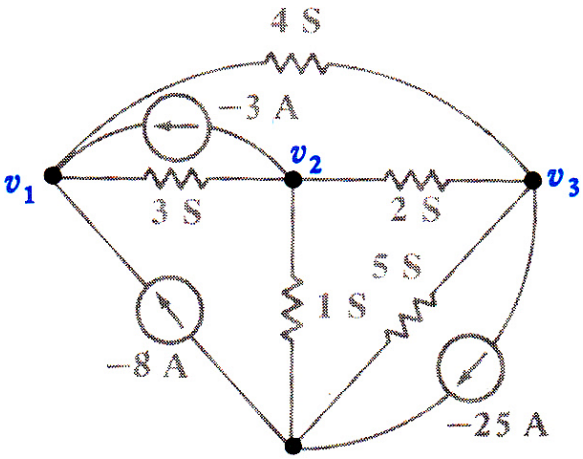
type

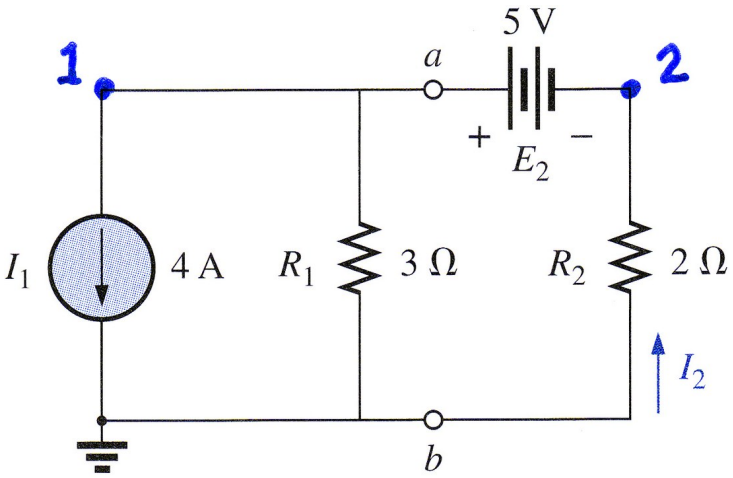
```
s\dc("j01,0,1,-8:j30,3,0,-25:j21,2,1,-3:r12,1,2,1/3:r23,2,3,1/2:r13,1,3,
1/4:r20,2,0,1:r30,3,0,1/5"):{v1,v2,v3}
```

The answer, **{1,2,3}**, is correct: v_1=**1**V, v_2=**2**V, v_3=**3**V.

B11's Example 8.5

Determine the current I_2 in the circuit shown. My solution:

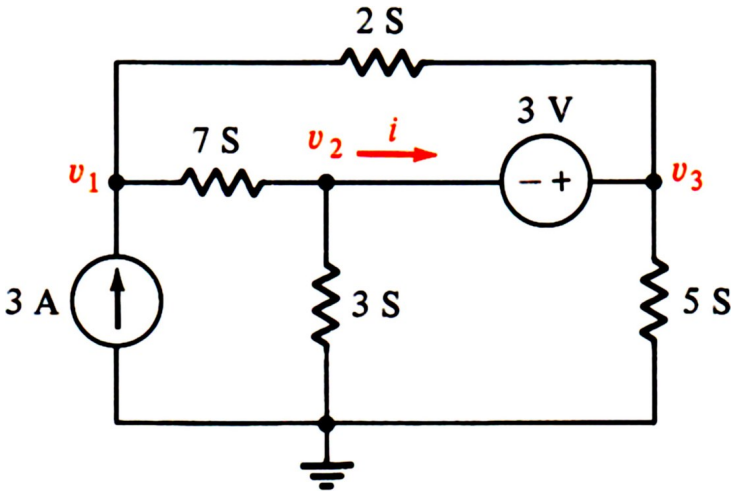





```
type
s\dc("j1,1,0,4:r1,1,0,3:e2,1,2,5:r2,0,2,2"):approx(ir2)
```

This gives us a value for I_2 of **3.4** A. This is correct.

Bo2's Example 2.5 (Conductances)



Determine the voltages in the nodes, and the current through the voltage source.

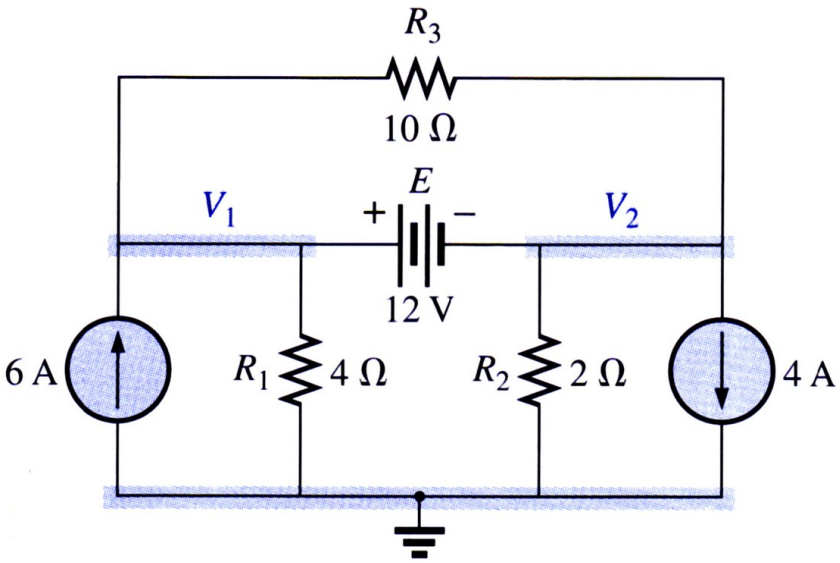
type

```
s\dc("j,0,1,3:e,3,2,3:r12,1,2,1/7:r20,2,0,1/3:r30,3,0,1/5:r13,1,3,1/2"):
approx({v1,v2,v3,-ie})
```

The answer, **{-.5,-1.5,1.5,11.5}**, is correct: v_1=**-.5**, v_2=**-1.5**, v_3=**1.5**, i=**11.5**

B11's Example 8.22

Determine V_1 and V_2. My solution:



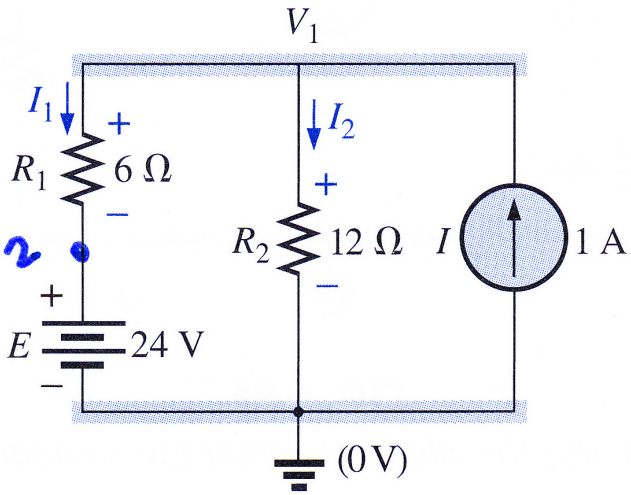
type

```
s\dc("j1,0,1,6:r1,1,0,4:e,1,2,12:r3,1,2,10:r2,2,0,2:j2,2,0,4"):
approx({v1,v2})
```

The answer, **{10.67,-1.33}**, tells us that V_1 is **10.67**V and V_2 is **-1.33**V.

B11's Example 8.19

Determine V_1, I_1 and I_2, in the circuit:

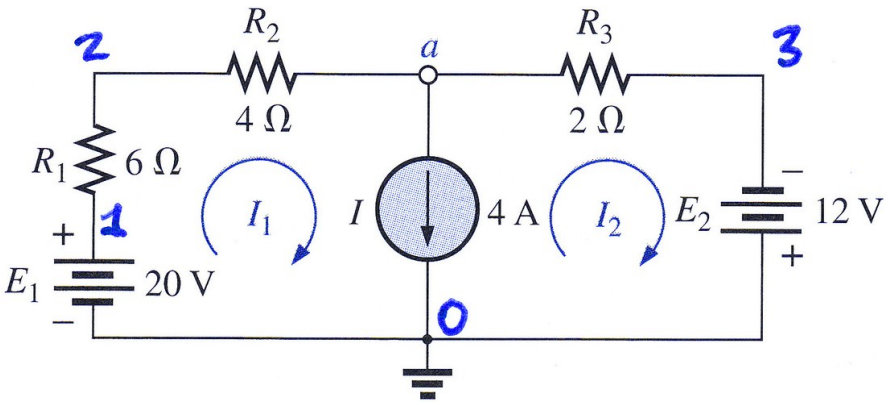


My solution:

```
type
s\dc("e,2,0,24:r1,1,2,6:r2,1,0,12:j,0,1,1"):approx({v1,ir1,ir2})
```

The answer, **{20.,-.667,1.67}**, tells us that V_1 is **20**V, I_1 is **-.667**A and I_2 is **1.67** A.

B11's Example 8.14



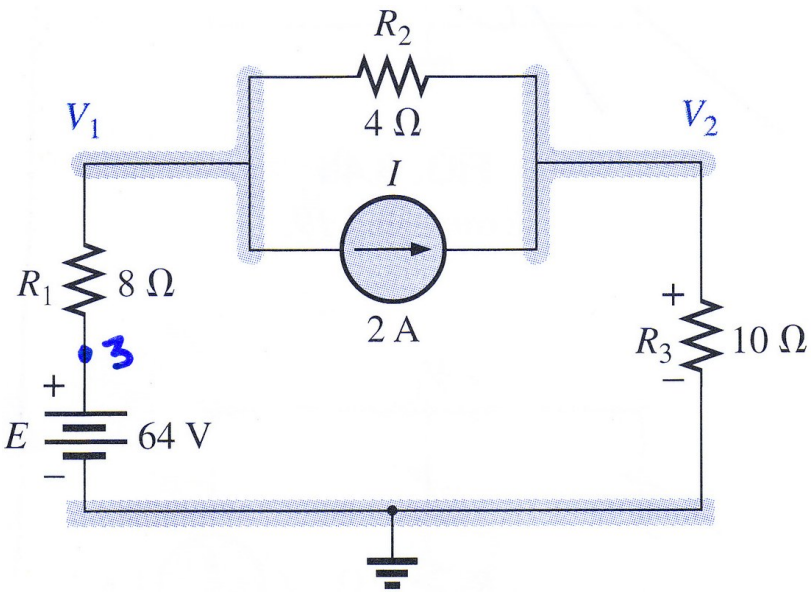
Determine I_2 and I_3.

```
type
s\dc("e1,1,0,20:r1,1,2,6:r2,2,a,4:j,a,0,4:r3,a,3,2:e2,0,3,12"):
approx({ir2,ir3})
```

Answer: **{3.33,-.667}**. This is correct.

B11's Example 8.20

Determine V_1, V_2, I_1, I_2 and I_3. My solution:

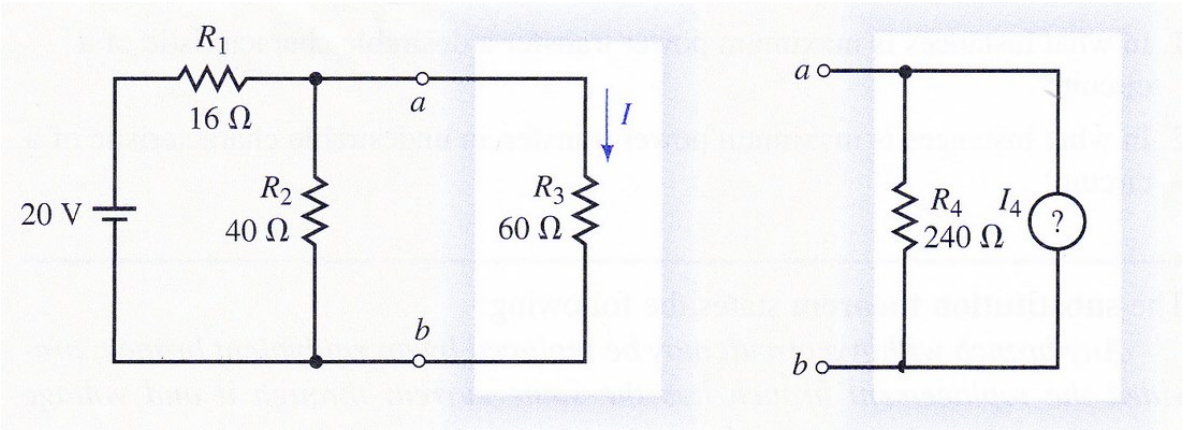


type
```
s\dc("e,3,0,64:r1,3,1,8:r2,1,2,4:j,1,2,2:r3,2,0,10"):
approx({v1,v2,ir1,ir2,ir3})
```

Answer: **{37.82,32.73,3.27,1.27,3.27}**. You should know how to read these by now, but here it is just in case: V_1 = **37.82**V, V_2 = **32.73**V, I_1 = **3.27**A, I_2 = **1.27**A, I_3 = **3.27**A.

RM3's Example 9-12 (*solve*)

If R_3 is to be replaced with R_4 and I_4, determine the value and direction of the source.



So that we can keep the name of node **b**, we will use the top left node as reference.

First, make sure you understand what this problem is asking you to do. The idea is that, despite the change, we keep the same voltage drop and current flow between nodes **a** and **b**. We must first know what they are. So we simulate the original circuit:

type
```
s\dc("e,0,b,20:r1,0,a,16:r2,a,b,40:r3,a,b,60"):{va-vb,ir3}
```

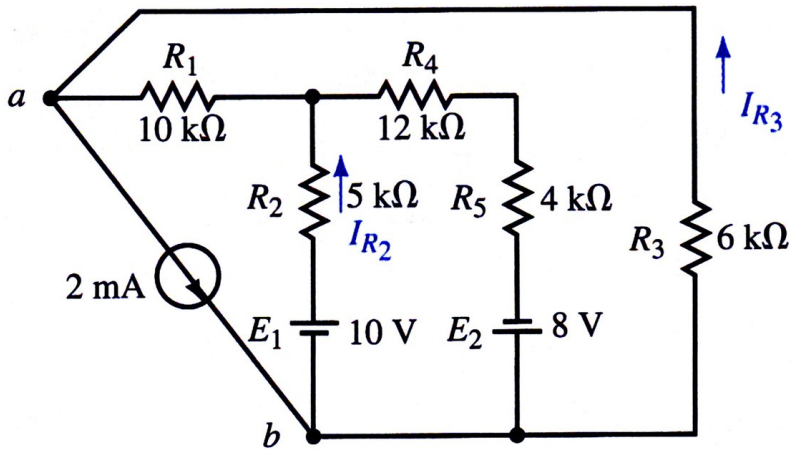
We find that the voltage drop is **12**V and the current is **0.2**A. These are the currents and voltages that we have to keep once we do the replacement. Simulate the circuit now replacing R_3 with a resistor R_4 of 240Ω and a source **j** with value I_4. Run this:

type
```
s\dc("e,0,b,20:r1,0,a,16:r2,a,b,40:r4,a,b,240:j,a,b,i4")
```

Notice that both the voltage drop (given by va-vb) and the current (given by `ir4+ij`) are algebraic functions in terms of i4. Now you can find i4 solving by voltage drop:

type
```
solve(va-vb=12.,i4) …or by current flow… solve(ir4+ij=0.2,i4)
```

The result is the same: i4 = **.15** A. The required current source is .15A from **a** to **b**.



RM3's Example 8-13

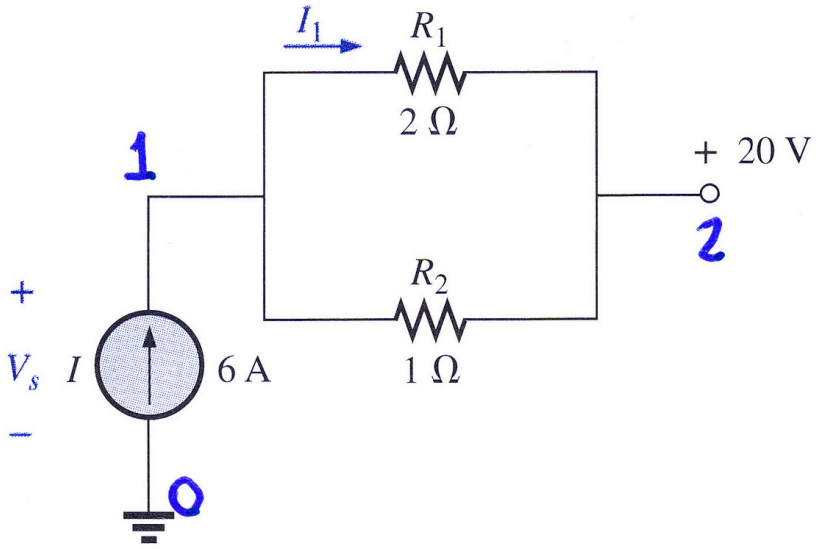
Solve for the currents through R_2 and R_3 in the circuit shown. My solution:

```
type
s\dc("r1,a,0,10'k:r2,1,0,5'k:r3,b,a,6'k:r4,0,2,16'k:
j,a,b,2'm:e1,1,b,10:e2,b,2,8"):approx({ir2,ir3})
```

The answer, **{.00154,.00111}** is correct: I_{R2} = **1.54** mA and I_{R3} = **1.11** mA.

B11's Example 6.3 (Hidden source)

Determine V_S and I_1.



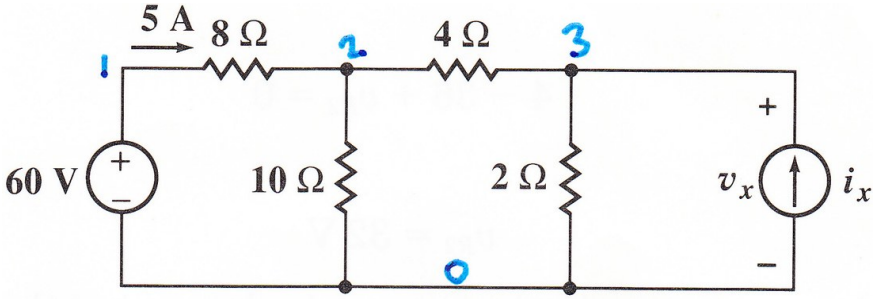
The problem presents a seemingly 'hanging' node with 20V. To simulate this, imagine a 'hidden' 20V voltage source connected to the node.

```
type
s\dc("j,0,1,6:r1,1,2,2:r2,1,2,1:e,2,0,20"):
{v1,ir1}
```

We get the answer: **{24,2}**, which is correct. V_S is **24**V and I_1 is **2**A.

HK5's Figure 1-24c (Expert)

Determine i_x and v_x in the following circuit.

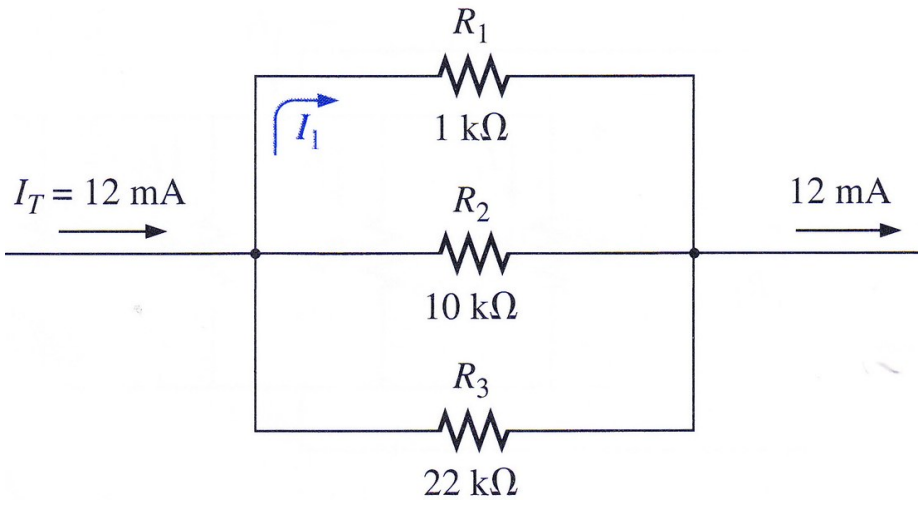



```
type
s\ex("e,1,0,60:r8,1,2,8:r10,2,0,10:r4,2,3,4:r2,3,0,2:j,0,3,ix"):
{ix,v3}
```

Select DC. Add equation `ir8=5`. Add unknown `ix`. Run the simulation, and you will get: **{1,8}**. This is correct: i_x is **1**A and that v_x is **8**V.

B11's Example 6.22 (Hidden source)

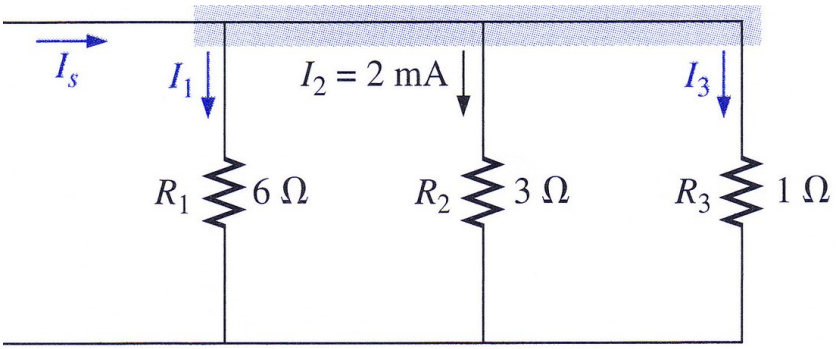
Determine I_1.



Although no source is shown, the circuit has a current. I use a 'hidden' current source.

```
type
s\dc("jt,0,1,12'm:r1,1,0,1'k:r2,1,0,10'k:r3,1,0,22'k"):approx(ir1)
```

Looking at the value of `ir1`, we get I_1 = **10.48** mA, which is correct.



B11's Example 6.21 (Hidden source, Expert)

Determine I_S, I_1 and I_3.

This 'hidden source' problem is perfect for Expert. My solution:

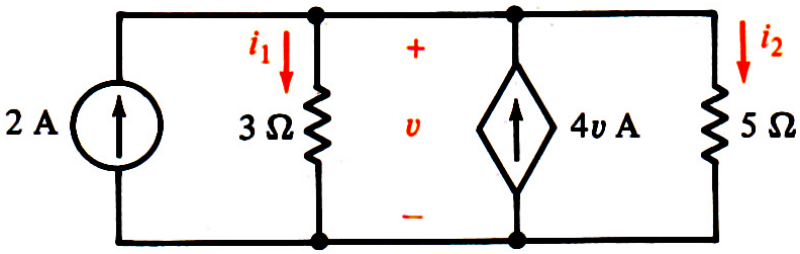
```
type
s\ex("js,0,1,is:r1,1,0,6:r2,1,0,3:r3,1,0,1"):approx({is,ir1,ir3})
```

Select DC, and press Enter. Add equation `ir2=2'm`. Add unknown `is`. Run the simulation.

The answer, **{.009,.001,.006}**, is correct, since the currents are as follows: I_S is **9**mA, I_1 is **1**mA and I_3 is **6**mA.

Bo2's Example 1.9

Determine I_1, I_2 and v.



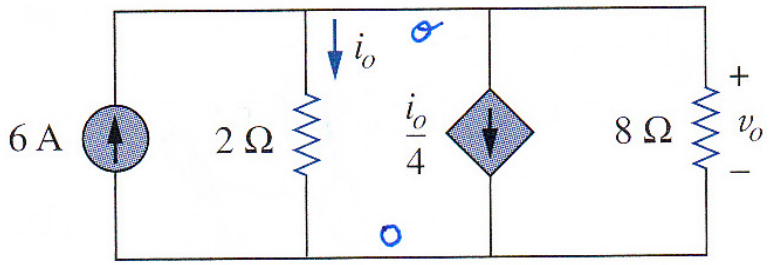
This is my solution.

```
type
s\dc("ji,0,1,2:r1,1,0,3:jd,0,1,4v1:r2,1,0,5"):{ir1,v1,ir2}
```

The answer, **{-5/26,-15/26,-3/26}**, is correct: I_1=**-5/26** A, I_2=**-3/26** A and v=**-15/26** V.

AS2's Practice Problem 2.7

Find v_o and i_o in the circuit. My solution:

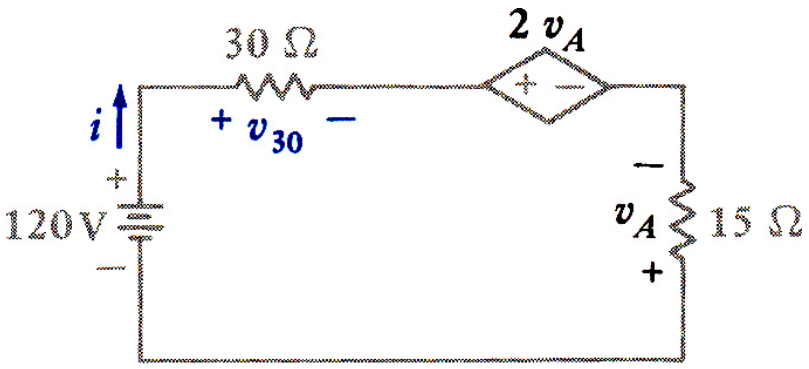


type
```
s\dc("ji,0,o,6:ro,o,0,2:jd,o,0,iro/4:r8,o,0,8"):{vo,iro}
```

The answer, **{8,4}**, is correct: v_o = **8** V and i_o = **4** A.

HK5's Example 1-3

Determine the power delivered by each source and consumed by both resistors.

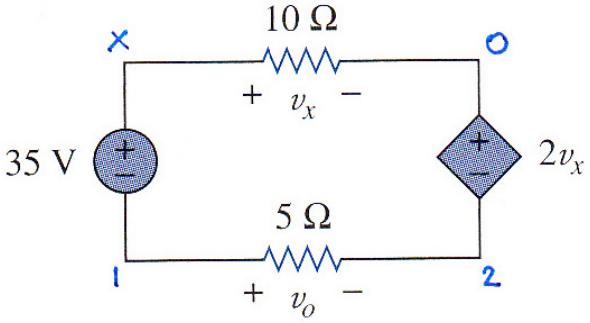


type
```
s\dc("ei,1,0,120:r1,1,2,30:ed,2,3,2vra:ra,0,3,15"):{-pei,-ped,pr1+pra}
```

The answer, **{960,1920,2880}**, is right: the independent source delivers **960**W, the dependent source delivers **1920**W, and the resistors consume **2880**W together.

AS2's Practice Problem 2.6

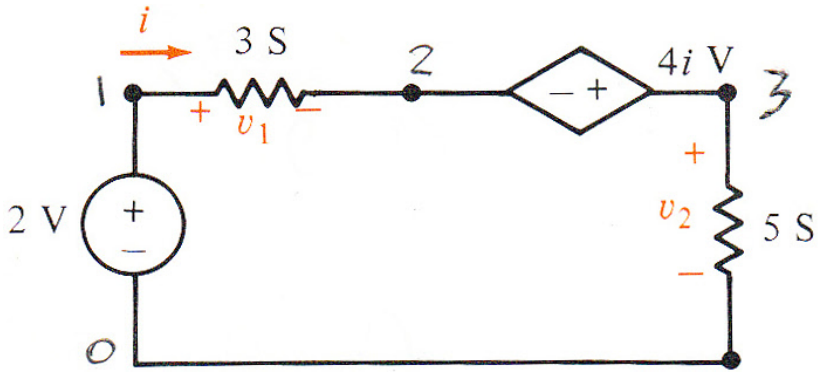
Find v_x and v_o in the circuit. My solution:



type
```
s\dc("ei,x,1,35:rx,x,0,10:ed,0,2,2vx:ro,1,2,5"):{vx,vro}
```

The answer, **{10,-5}**, is correct: v_x =**10** and v_o =**-5**.

Bo2's Example 1.10

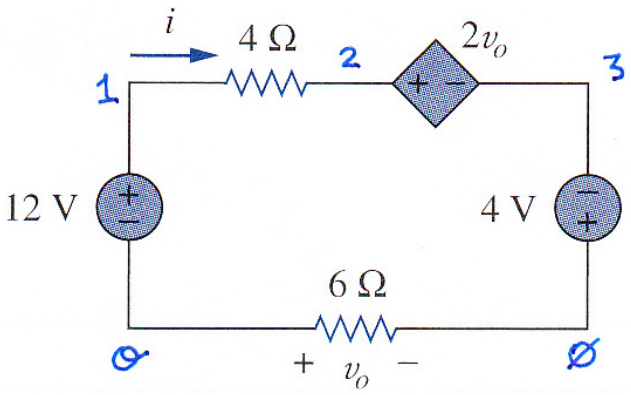


Determine v_1, v_2 and i. My solution: The simulation took 14 seconds.

type
```
s\dc("ei,1,0,2:r1,1,2,1/3:ed,3,2,4*ir1:r2,3,0,1/5"):{ir1,vr1,vr2}
```

The answers, **{-15/26,-5/26,-3/26}**, is correct: v_1=**-5/26** V, v_2=**-3/26** V and i=**-15/26** A.

AS2's Example 2.6



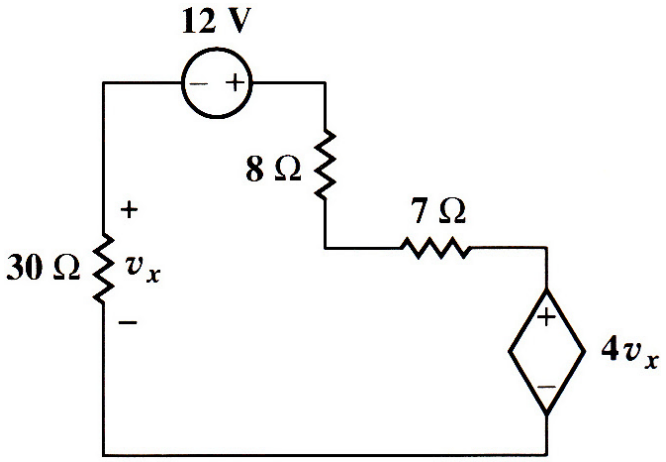
Determine v_o and i in the circuit. My solution is shown below:

type
```
s\dc("e12,1,o,12:ri,1,2,4:ed,2,3,2vo:e4,0,3,4:ro,o,0,6"):{vo,iri}
```

The answer, **{48,-8}**, is correct: v_o =**48** and i = **-8**.

HK5's Drill Problem 1.11

Find the power absorbed by each element in the circuit. My solution:

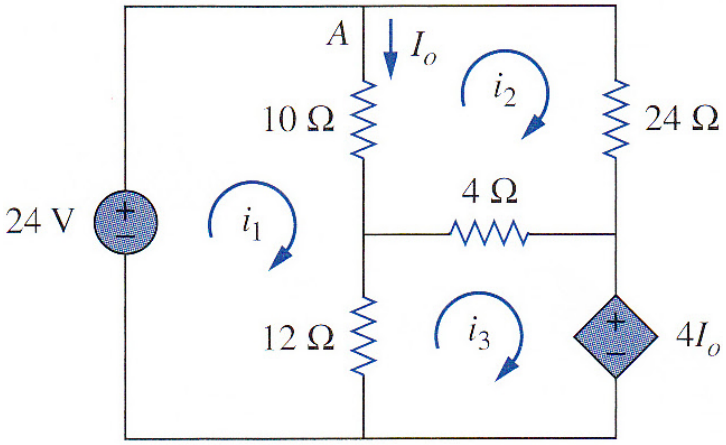



```
type
s\dc("r1,x,0,30:ei,1,x,12:r2,1,2,8:r3,2,3,7:ed,3,0,4vx"):
approx({pr1,pei,pr2,pr3,ped})
```

The answer, **{.768,1.92,.2048,.1792,-3.072}**, is correct.

AS2's Example 3.6

Determine the value of I_o in the circuit. My solution:




```
type
s\dc("ei,a,0,24:ro,a,b,10:r12,b,0,12:r4,b,c,4:r24,a,c,24:ed,c,0,4iro"):
approx(iro)
```

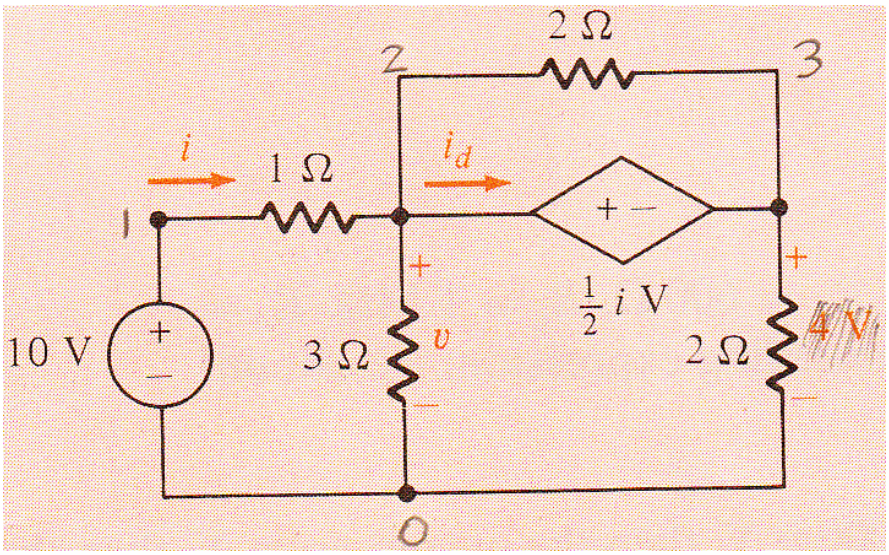
The answer, **1.5** A, is correct.

Bo2's Drill Exercise 1.12

Determine i, v and i_d.

We are given an unnecessary piece of information: the 4V drop in the 2Ω resistor.

My solution:

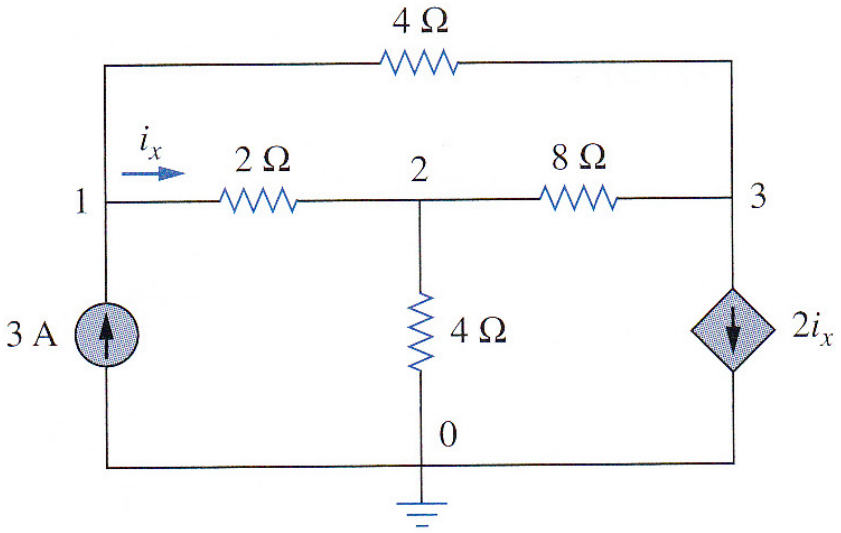



```
type
s\dc("ei,1,0,10:r1,1,2,1:r2,2,3,2:r3,2,0,3:
r4,3,0,2:ed,2,3,ir1/2"):{ir1,vr3,ied}
```

The answer, **{4,6,1}**, is correct: ***i*=4**, ***v*=6** and i_d**=1**.

AS2's Example 3.2

Determine the voltages at the nodes.



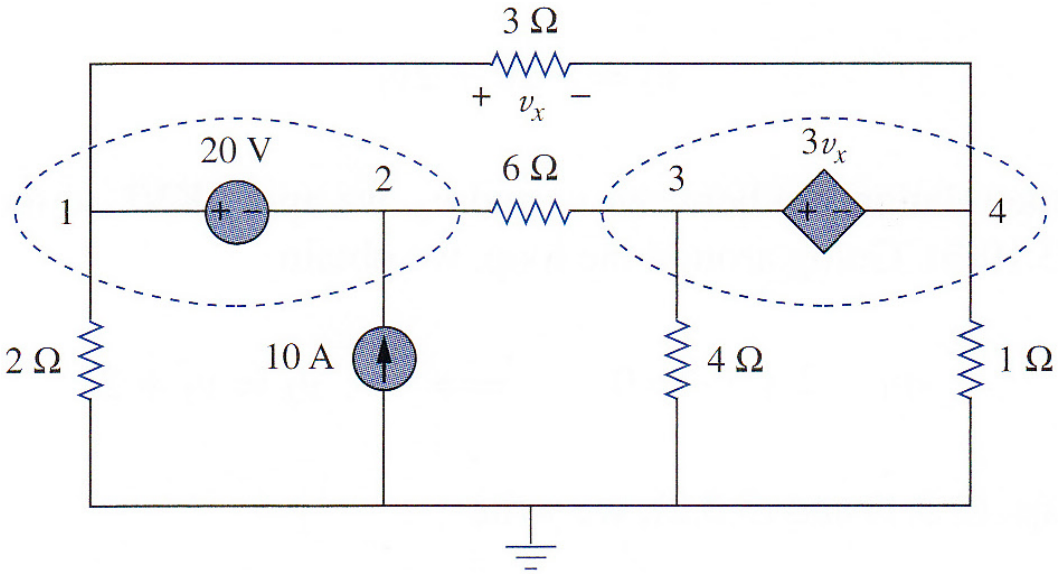
My solution:

```
type
s\dc("ji,0,1,3:jd,3,0,2ir2:r2,1,2,2:r4a,1,3,4:r8,2,3,8:r4b,2,0,4"):
approx({v1,v2,v3})
```

The answer, **{4.8,2.4,-2.4}**, is correct.

AS2's Example 3.4

Find the node voltages in the circuit. My solution:




```
type
s\dc("r2,1,0,2:e,1,2,20:j,0,2,10:r6,2,3,6:rx,1,4,3:r4,3,0,4:
ed,3,4,3vrx:r1,4,0,1"):approx({v1,v2,v3,v4})
```

The answer, **{26.67,6.67,173.33,-46.67}**, is correct.

Bo2's Drill Exercise 2.6

Determine the voltage in each node.

Notice that in the schematic, the resistors' values are given in siemens. Symbulator has to be fed the resistors with values in ohms. This means that, when we simulate them, we have to convert them from siemens to ohms by dividing 1 over the siemens value.

My solution:

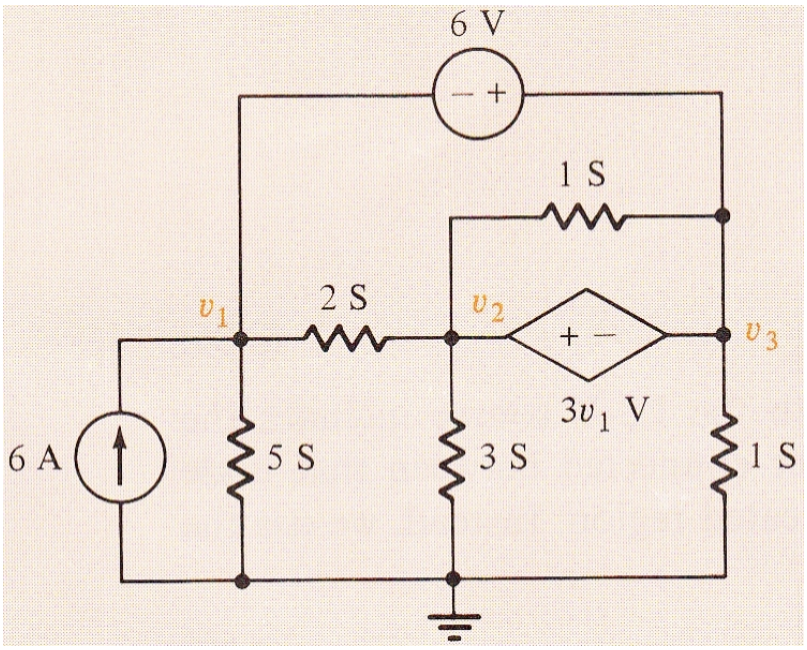
```
type
s\dc("j,0,1,6:ei,3,1,6:ed,2,3,3v1:r5,1,0,1/5:r2,1,2,1/2:
r3,2,0,1/3:r1,2,3,1:r4,3,0,1"):{v1,v2,v3}
```

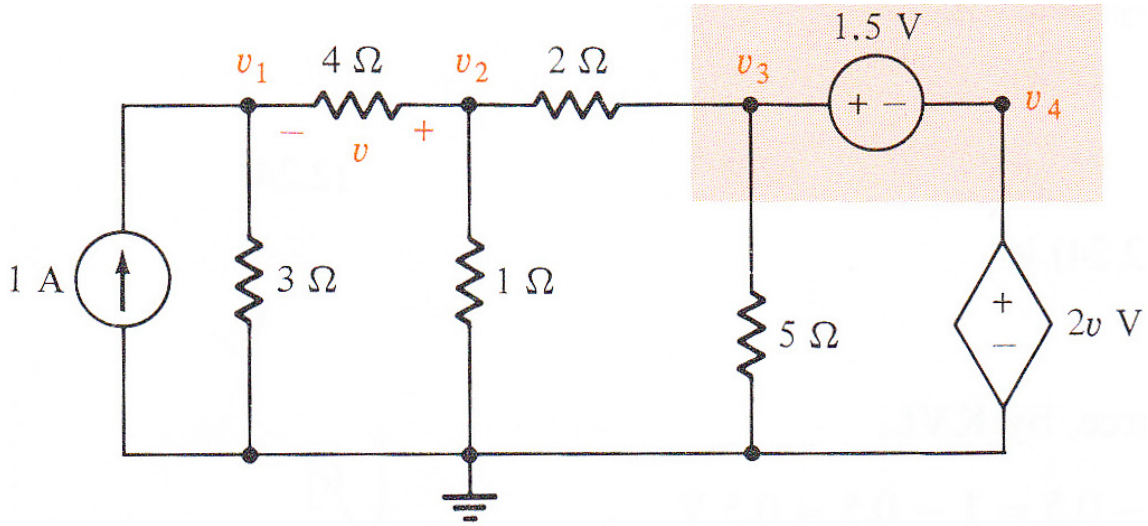
The answer, **{-1,2,5}**, is correct.

Bo2's Example 2.7

Determine the voltages in all nodes. My solution:

```
type
s\dc("j,0,1,1:r3,0,1,3:r4,2,1,4:r1,2,0,1:r2,2,3,2:
r5,3,0,5:ei,3,4,1.5:ed,4,0,2vr4"):{v1,v2,v3,v4}
```





The answer, **{1.5,-.5,-2.5,-4.}**, is correct.

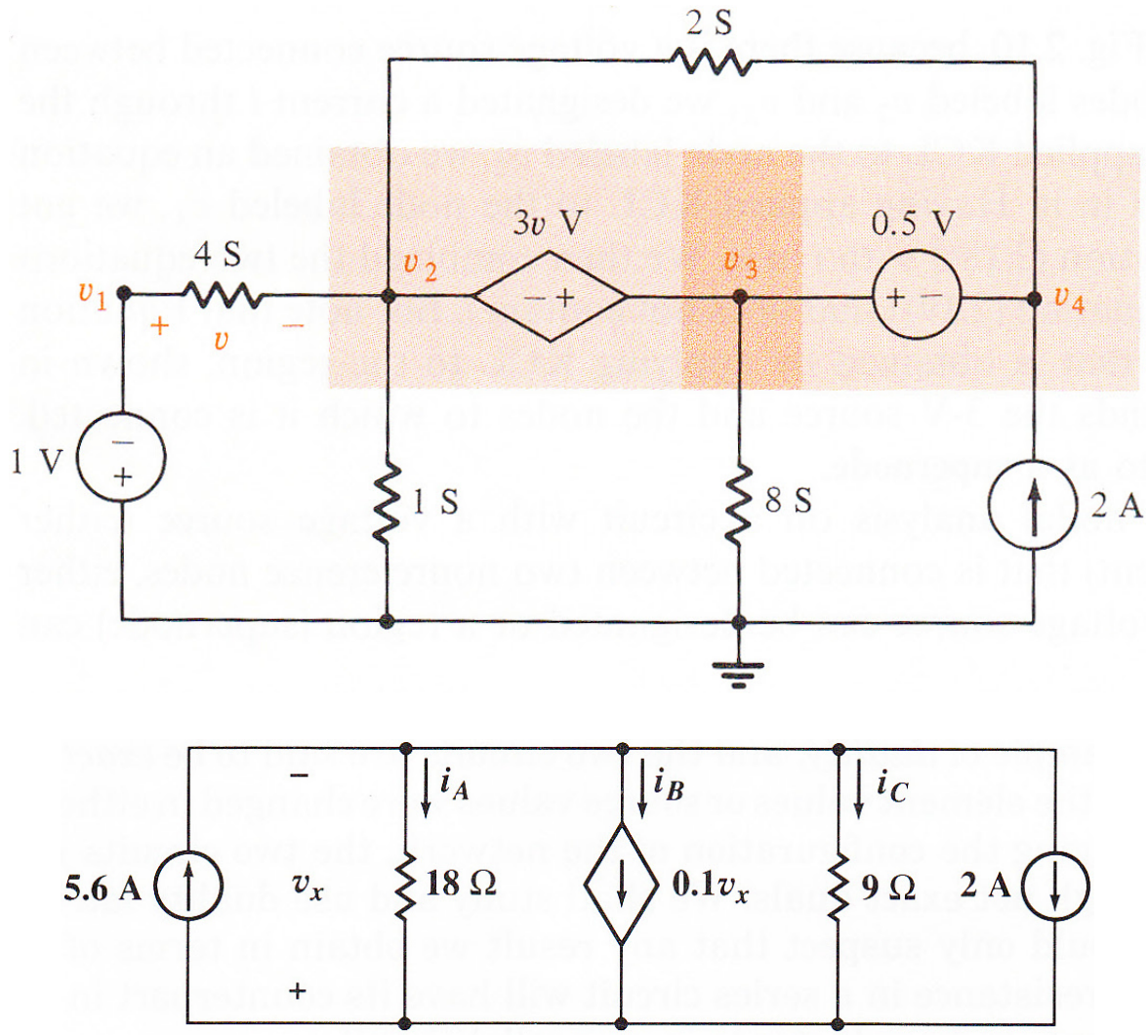
Bo2's Example 2.6

Determine the voltages in all nodes.

My solution:

```
type
s\dc("e1,0,1,1:e2,3,4,.5:ed,3,2,3vr4:j,0,4,2:r4,1,2,1/4:
r1,2,0,1:r8,3,0,1/8:r2,2,4,1/2"):{v1,v2,v3,v4}
```

The answer, **{-1,-2.,1.,.5}**, is correct.



HK5's Drill Problem 1-12

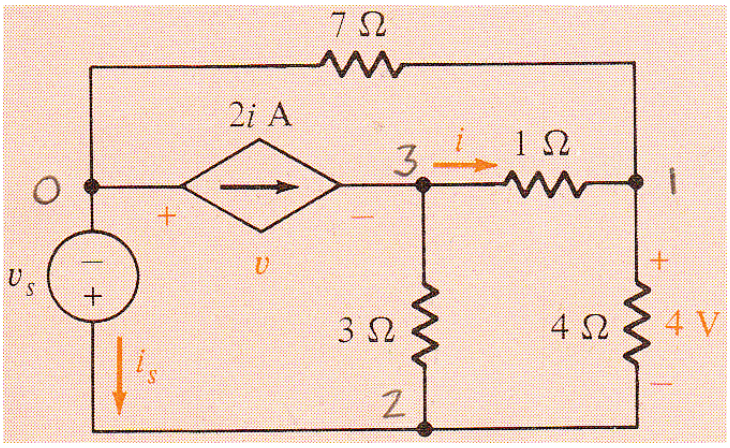
Find i_A, i_B and i_C.

My solution:

```
type
s\dc("jl,x,0,5.6:ra,0,x,18:jb,0,x,.1vx:r9,0,x,9:jr,0,x,2"):
approx({ira,ijb,ir9})
```

The answer, **{3.,-5.4,6.}**, is correct.

Numerical-from-symbolic examples



Bo2's Drill Exercise 1.10

Determine i, v, i_s and v_s.

With one unknown value and one known solution, this problem is a job for Expert.

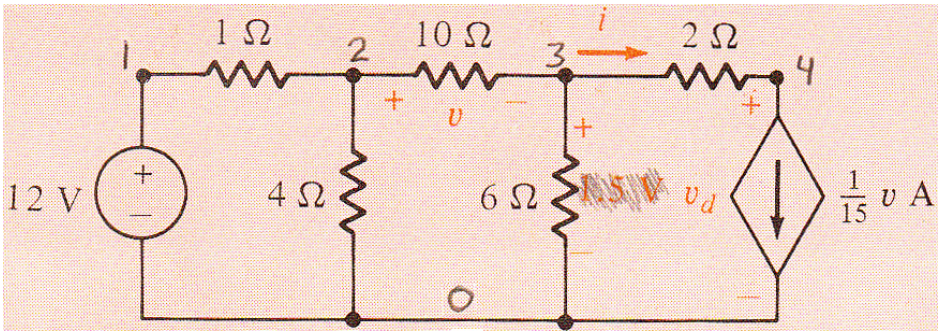
Determine i, v, i_s and v_s.

```
type
s\ex("es,2,0,vs:jd,0,3,2ir1:r7,0,1,7:r1,3,1,1:r3,3,2,3:r4,1,2,4"):
{ir1,vjd,-ies,vs}
```

Select DC, Add `vr4=4` to the equations and `vs` to the unknowns. Run the simulation. The answer, **{2,-9,-3,3}**, is correct: i=**2**, v=**-9**, i_s=**-3** and v_s=**3**.

Bo2's Drill Exercise 1.11

Determine i, v and v_d. (Since all element values are known, the tip we are given by the book – namely, that the voltage drop in the 6Ω resistor is 1.5V – is totally superfluous.)

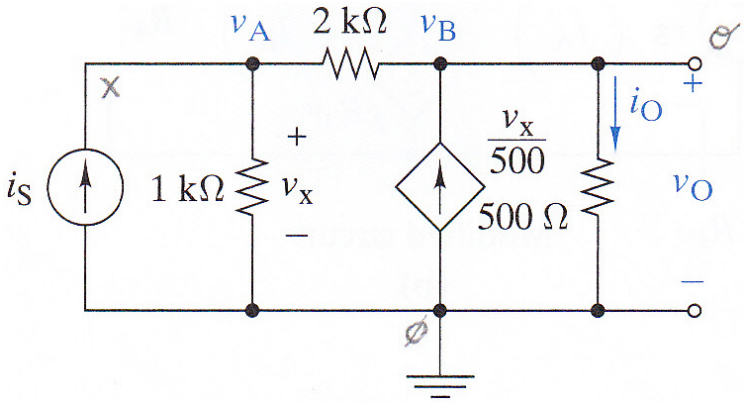



```
type
s\dc("e,1,0,12:r1,1,2,1:r4,2,0,4:r10,2,3,10:r6,3,0,6:
r2,3,4,2:j,4,0,vr10/15"):approx({vr10,ir2,vj})
```

The answer, **{7.5,.5,.5}** is correct: i=**.5**, v=**7.5** and v_d=**.5**.

TR5's Exercise 4.2

Find v_O and i_O in terms of i_S.



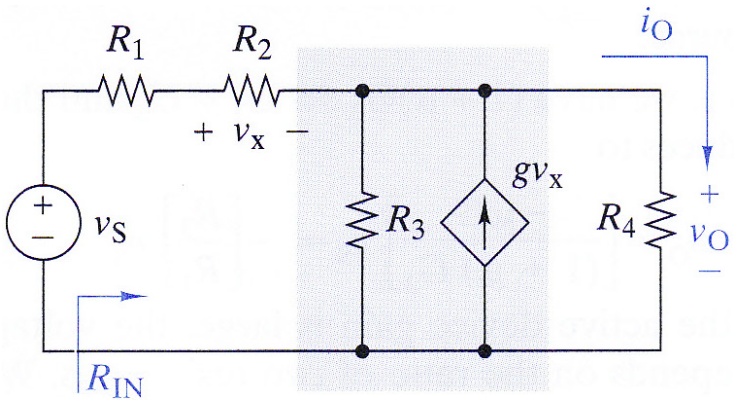
This is my solution. The simulation took 14 seconds.

```
type
s\dc("ji,0,x,is:r1,x,0,1'k:r2,x,o,2'k:jd,0,o,vx/500:ro,o,0,500"):
{vo,iro}
```

The answer, **{1000*is,2*is}**, is correct: v_O=*1000* i_S and i_O=*2* i_S.

TR5's Example 4.4

Find v_O and the equivalent resistance R_{IN}, in terms of v_S, when R_1 is 50 Ω, R_2 is 1 kΩ, R_3 is 100 Ω, R_4 is 5 kΩ and g is 100 mS (i.e. 100'm).



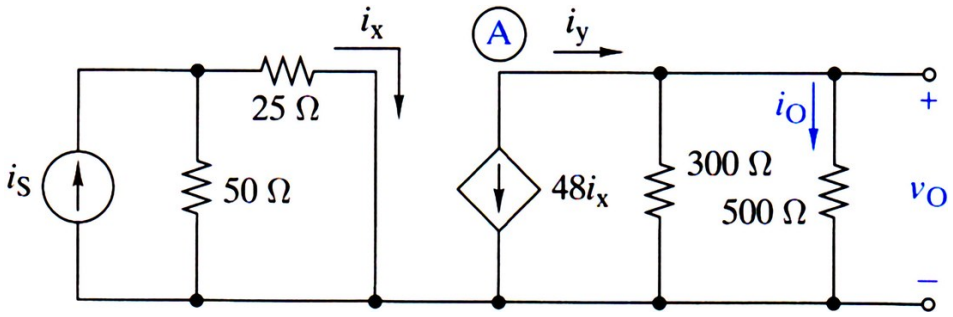
This is my solution. The simulation took 18 seconds.

```
type
s\dc("e,1,0,vs:r1,1,2,50:r2,2,o,1'k:r3,o,0,100:
r4,o,0,5'k:j,0,o,100'm*vr2"):approx({vo,re})
```

The answer, **{.904*vs,10952.}**, is correct: v_O=*.904* v_S and R_{IN} = *10.95 kΩ*.

TR5's Figure 4-4

Find the voltage drop, current, and power consumed by the 500Ω resistor, and the ratio of that power to that delivered by the independent source, all in terms of i_S.



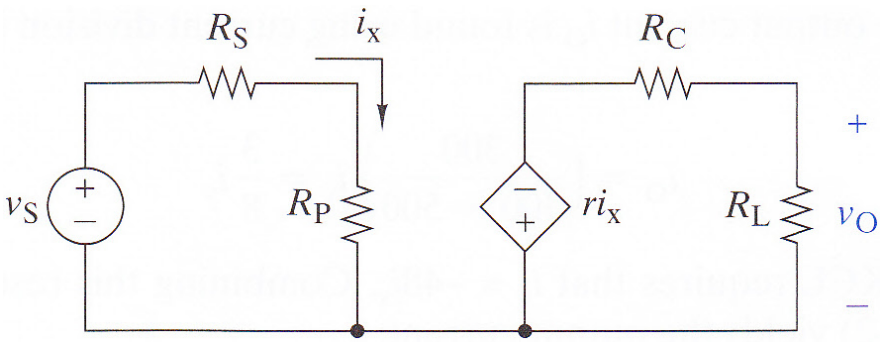
I have not labeled the nodes in the figure, so you can practice doing it. My solution:

```
type
s\dc("js,0,1,is:r50,1,0,50:rx,1,0,25:jd,o,0,48irx:
r3,o,0,300:ro,o,0,500"):{iro,vo,pro,pro/(-pjs)}
```

The answers we get are correct: i_O=**-12is**, v_O=**-6000is**, p_O=**72000is**^{2}, and p_O/p_S=**4320**.

TR5's Example 4.1 (Symbolic)

Find v_O.



I did not label the nodes, so you can practice. Remember not to use **rc**: it is reserved.

```
type
s\dc("ei,1,0,vs:rs,1,2,rs:rx,2,0,rp:ed,0,3,r*irx:rrc,3,o,rrc:rl,o,0,rl"):
vo
```

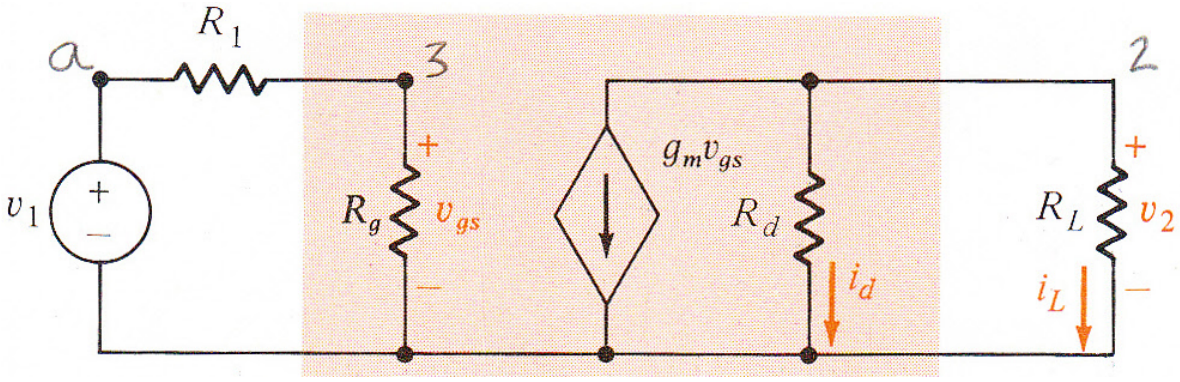
The answer, shown first below, is correct. The textbook's answer follows it.

$$\frac{-rl \cdot vs \cdot r}{(rl + rrc) \cdot (rp + rs)}$$

$$v_O = \left[\frac{-R_L r}{(R_S + R_P)(R_C + R_L)}\right] v_S$$

Bo2's Example 1.11 (Symbolic)

Determine v_2.



My solution: In my solution I named the value of the source v_1, to keep it similar to the book. This required avoiding naming any node as **1**: if there was a node **1**, Symbulator would store in **v1** the voltage of the node, creating trouble. There is no problem with using r1 as a value, since nothing will be stored in that r1 value.

```
type
s\dc("e,a,0,v1:r1,a,3,r1:rg,3,0,rg:j,2,0,gm*vrg:rd,2,0,rd:rl,2,0,rl"):v2
```

The simulation took 25 seconds. The answer I got is shown first below (the textbook's follows it.)

$$\frac{-gm \cdot rd \cdot rg \cdot rl \cdot v1}{(r1 + rg) \cdot (rd + rl)}$$

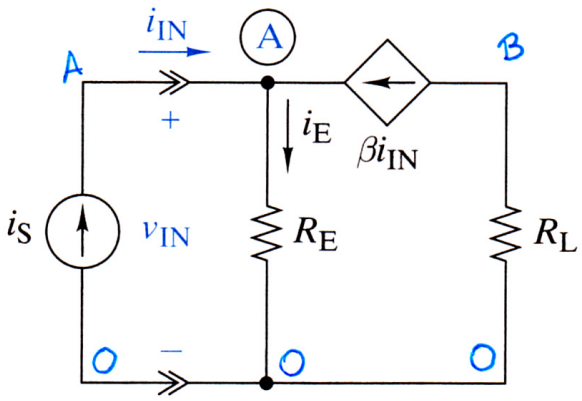
$$v_2 = \frac{-g_m R_d R_g R_L v_1}{(R_g + R_1)(R_L + R_d)}$$

TR5's Example 4-7 (Symbolic)

Find R_{IN}, i.e. the resistance as seen by the current source. My solution:

```
type
s\dc("ji,0,a,is:re,a,0,re:jd,b,a,β*is:rl,b,0,rl"):rji
```

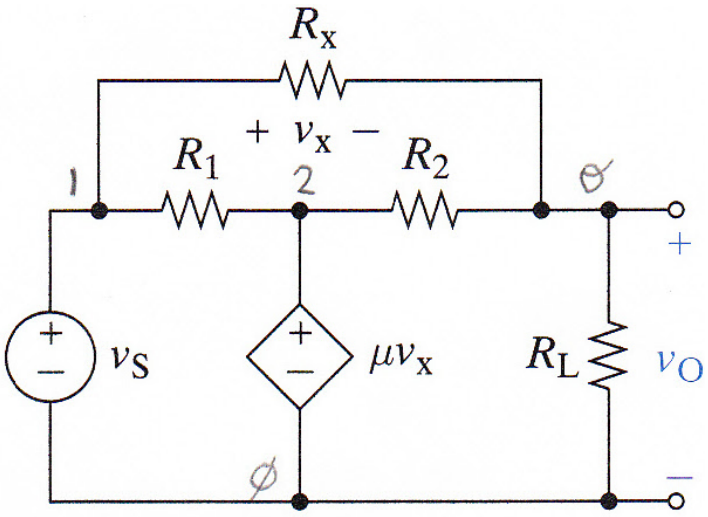
The answer we get — `re*(β+1)` — is correct, as can be seen by comparing it to the textbook's answer.



$$R_{IN} = (\beta + 1)R_E$$

TR5's Exercise 4.3 (Symbolic)

Find v_O, in terms of the value in the circuit. For resistors, use their conductance value.



This is my solution. We use μ as a constant in the dependent source. It is not confused with the SI prefix for micro because the prefix has an apostrophe.

```
type
s\dc("ei,1,0,vs:ed,2,0,μ*(vrx):r1,1,2,1/g1:
r2,2,o,1/g2:rx,1,o,1/gx:rl,o,0,1/gl"):vo
```

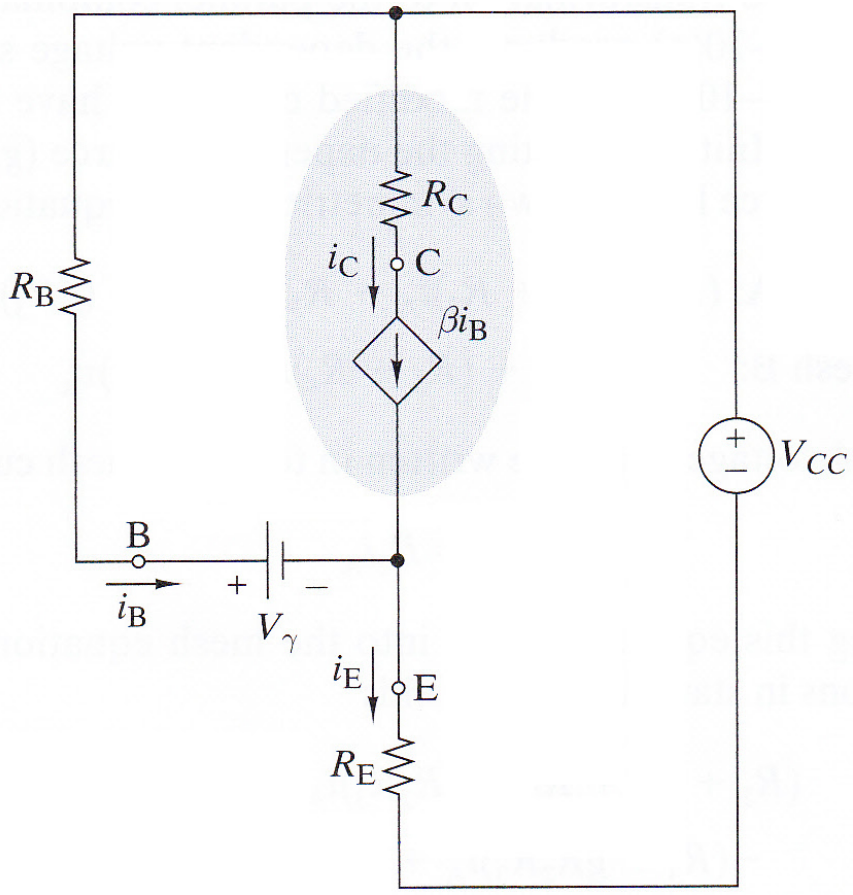
This is the answer we get. It is correct. Compare it with the textbook's answer below it.

$$\frac{(g2 \cdot \mu + gx) \cdot vs}{g2 \cdot (\mu + 1) + g1 + gx}$$

$$v_O = \frac{G_x + \mu G_2}{G_x + G_L + (\mu + 1)G_2} v_S$$

TR5's Example 4.5 (Symbolic)

Find i_B.



My solution is shown below. Be patient. This simulation takes more than one minute.

```
type
s\dc("e1,1,0,vcc:rb,1,b,rb:e2,e,b,vγ:re,e,0,re:rrc,1,c,rrc:j,c,e,β*irb"):
irb
```

Compare my answer, first below, to the book's answer beneath it.

$$\frac{vcc + v\gamma}{rb + re \cdot (\beta + 1)}$$

$$i_B = \frac{V_{CC} - V_\gamma}{R_B + (\beta + 1)R_E}$$

The last four problems show Symbulator at its DC best. I don't know of any calculator-based program that was able to provide this kind of purely symbolic answer to a circuit simulator back in 1999 when I made Symbulator. As a matter of fact, even today – a quarter of a century later - I know of no other calculator-based simulator that can do this.

LESSON 4

Shorts, equivalent resistance and Thévenin/Norton

Learn how to describe a *short circuit* using the **s** element. Learn how to find *equivalent resistances* using the **er** script, and *Thévenin and Norton equivalents* using the **th** script.

In this lesson you will learn how to describe a *short circuit* with the **s** element, how to find *equivalent resistances* with the **er** script, and *Thévenin and Norton equivalents* with the **th** script.

4.1 How to describe a short circuit

Shorts are used mostly to find a current in a part of the circuit with no element in it already. Otherwise we would just make it a single node.

> ⓘ NOTE
> **Describing short circuits**
> In Symbulator, an ideal short circuit is described with three bits of information, separated by commas: the name of the short, which must start with the letter **s**, then the names of its first and second node.
> For example, an ideal short circuit called **s1**, between nodes **3** and **5**, is described: `s1,3,5`

What answers do you get

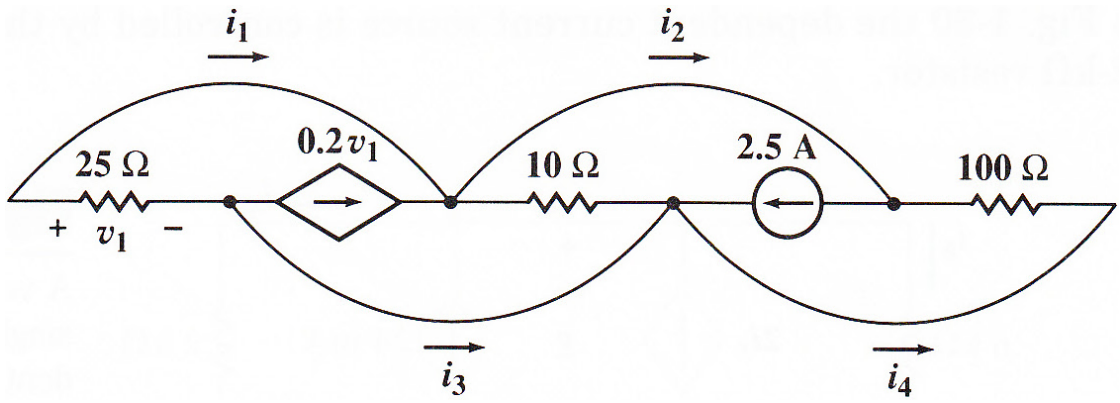
No power is consumed, and no voltage is dropped, in a short circuit. For each short in a circuit, Symbulator stores only the current through it, flowing from the first node towards the second. For a short called **sx**, that is `isx`.

HK5's Drill Problem 1-13

Find i_1, i_2, i_3 and i_4.

SOLUTION

My solution: I define the shorts in the same direction as the arrows in the schematic.



HK5's Drill Problem 1-13

```
type
s\dc("r1,1,0,25:jd,0,2,.2v1:r2,2,3,10:ji,4,3,2.5:r3,4,5,100:s1,1,2:s2,2,4:
s3,0,3:s4,3,5")
```

We ask for the values of the variables:

```
type
approx({is1,is2,is3,is4})
```

```
returns
{-2., 3., -8., -.5}
```

These are correct, and they can only be found using short circuits.

4.2 The equivalent resistance script: er

As we saw, Symbulator gives the equivalent resistance of a circuit as seen from any source. That lets us solve problems like this one.

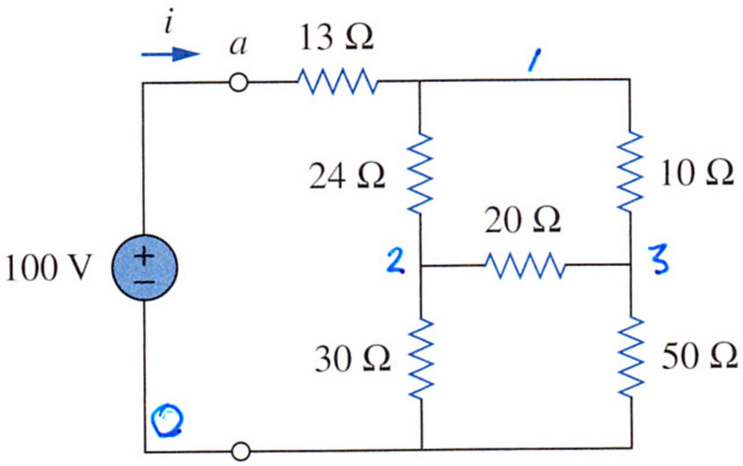
AS2's Practice Problem 2.15

Find the equivalent resistance as seen by the 100 V source, and the value of current i.

SOLUTION

```
type
s\dc("e,a,0,100:r13,a,1,13:r24,1,2,24:r10,1,3,10:r20,2,3,20:r30,2,0,30:
r50,3,0,50"):re
```

Evaluating `re` gives the equivalent resistance as seen by the source **e**: 40 Ω. Evaluating `ir13` gives **2.5** A for current ***i***. The answers came easily, because a source sat between the two nodes we wanted.



AS2's Practice Problem 2.15

What if there is no independent source?

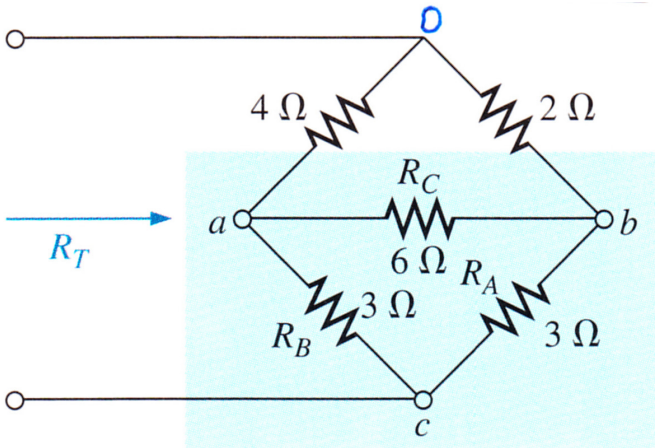
But how do we find the equivalent resistance of a **passive circuit**, one with no *independent* source in it? One way is to connect a 1 A current source between the two nodes and read the voltage drop across it. That is the manual way.

An easier way is to let Symbulator do that for us.

Just run the **er** script and it does the same thing automatically. It finds the equivalent resistance of a passive circuit and stores it — for a DC analysis — in **req**. It takes three arguments: the circuit description as a string, and the two nodes to measure between.

B11's Example 8.29

Calculate the equivalent resistance of the circuit shown.



B11's Example 8.29

SOLUTION

My solution: I label the nodes, then describe the circuit and store it in a variable.

```
type
"r4,0,a,4:r2,0,b,2:r6,a,b,6:rb,a,c,3:ra,b,c,3"→cir
s\er(cir,0,c)
```

When prompted, choose DC as analysis type. Once it is done, evaluate:

```
type
approx(req)
```

```
returns
2.89
```

The value is 2.89 Ω. This is correct.

What counts as passive

That example was made of resistors only. The er script also handles a second kind of passive circuit: one with resistors and dependent sources, but no independent sources. Such a circuit can only be reduced to an equivalent resistance, not to a Thévenin or Norton equivalent, so **er** is the script to use — in exactly the same way, provided the dependent sources are described properly.

4.3 The Thévenin / Norton script: th

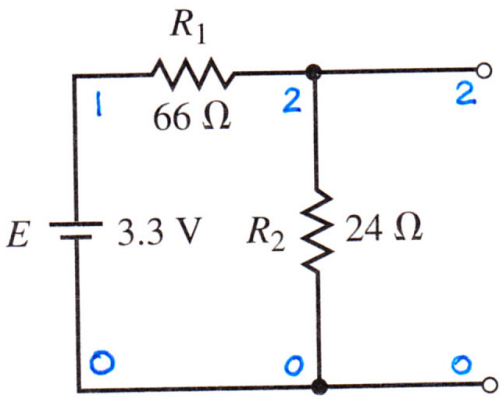
Just as a *passive circuit* can be reduced to an equivalent resistance, an **active circuit** — one with independent sources — can be reduced to a Thévenin or Norton equivalent.

One way is to run a first simulation for the voltage between the two nodes where we want the equivalent (the Thévenin voltage, V_{TH}), then a second with a short between those nodes for the current through it (the Norton current, I_{NO}). R_{EQ} is then V_{TH}/I_{NO}. That is the manual way.

An easier way is the **th** script, which does exactly that automatically. It takes three arguments: the circuit description, the first node and the second node.

RM3's Practice Problem 9-4

Find the Thévenin and Norton equivalents of the circuit.



RM3's Practice Problem 9-4

SOLUTION

This is my circuit description, and this is how we run the th script:

```
type
"e,1,0,3.3:r1,1,2,66:r2,2,0,24"→cir
s\th(cir,2,0)
```

We could also have passed the circuit description directly:

```
type
s\th("e,1,0,3.3:r1,1,2,66:r2,2,0,24",2,0)
```

When prompted to select a type of analysis, choose DC. Symbulator tells you what it is doing: one simulation for the Thévenin voltage, then a second for the Norton current and the equivalent resistance. In this case, V_{TH} = **0.88** V, I_{NO} = **0.05** A and R_{EQ} = **17.6** Ω.

When you press ENTER, Symbulator asks whether you are running a problem with a load connected to this equivalent circuit. For now, say No. The script then stores these variables:

- **vth** has the Thévenin voltage
- **ino** has the Norton current
- **req** has the equivalent resistance
- **pmax** has the maximum power that the equivalent can deliver to a hypothetical load

Problems with a load

One type of problem books and professors like when teaching the Thévenin / Norton equivalents is what I call an R_L problem. A typical one goes like this: "First, reduce the circuit, as seen by resistor R_L, to its Thévenin or Norton equivalent. Then, find the value of the voltage drop, current and/or power consumed in the load resistor R_L if its value is (whatever) ohms."

Since this is such a typical problem, I've made some provisions in Symbulator to help you solve them. Right after a Thévenin or Norton equivalent is found, the **th** script asks whether you are planning to connect a load. The default is No, but if you select Yes, Symbulator will save some special answers for that very typical case, as functions of the **load** variable:

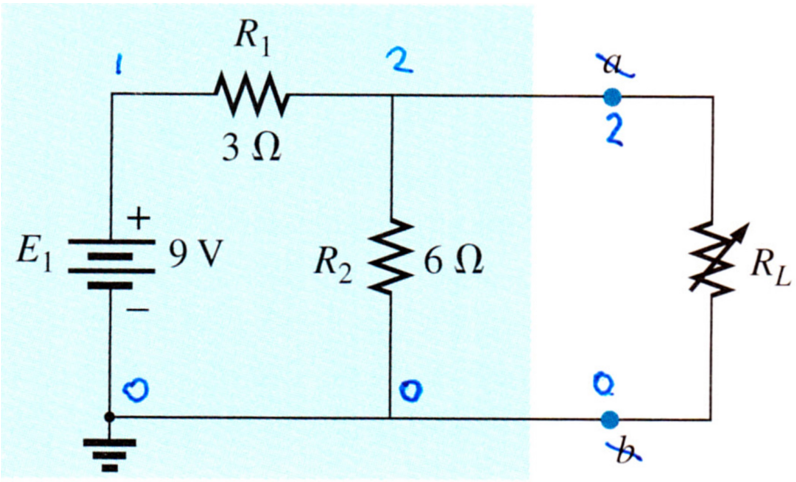
- **irl** has the current in the load
- **vrl** has the voltage drop in the load
- **prl** has the power consumed in the load

B11's Example 9.6

Find the Thévenin equivalent circuit for the network in the shaded area. Then find the current through R_L for R_L values of 2 Ω, 10 Ω and 100 Ω.

SOLUTION

First we find the Thévenin equivalent.



B11's Example 9.6

```
type
s\th("e1,1,0,9:r1,1,2,3:r2,2,0,6",2,0):{vth,req}
```

```
returns
{6, 2}
```

Correct. Now we find the current in the load for each value. We can do this in a single push, or separately. Here I find them in one go:

```
type
{irL|Load=2.,irL|Load=10.,irL|Load=100.}
```

```
returns
{1.5, .5, .059}
```

Where | is the "given" operator. The answers are correct.

Power transfer problems

Another problem often associated with the Thévenin / Norton equivalents is power transfer to a load, particularly the maximum possible. Maximum power is transferred when the load R_L equals the R_{EQ} of the equivalent. Symbulator's th script gives you the maximum power that can be delivered in **pmax**, and the power transferred to the load as a function of its value in **prl**.

What if it's more than a load?

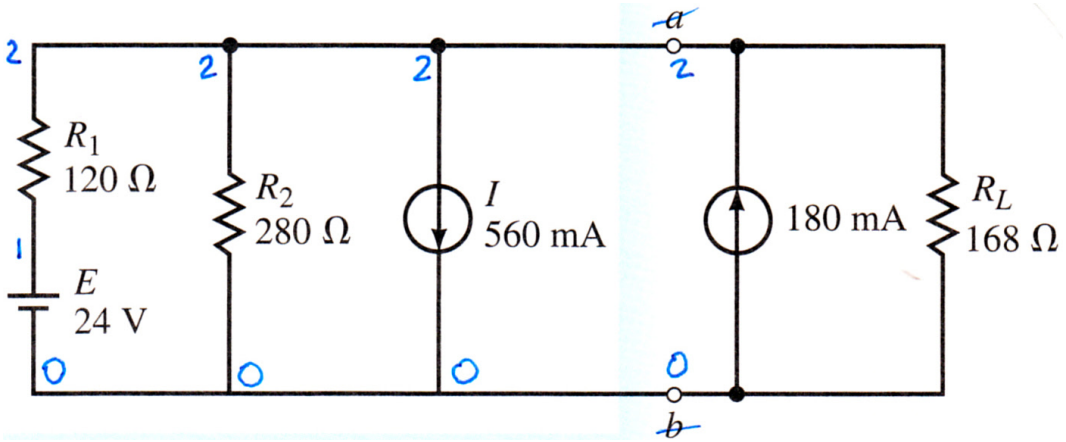
The formulas for the current, voltage and power in the load apply only when a load is the only thing connected to the equivalent circuit. If the problem you want to solve includes something more complicated, you will have to run your own simulation.

To help you in those cases, let me show you one more goody of the **th** script, at risk of promoting vagrancy among EE students: once it has found the Norton equivalent of a

circuit, it will automatically write for you the circuit description of that Norton equivalent connected to a load, and store it in a string called **eqcir**. You can use it as a starting point. It has the Norton equivalent connected, between nodes **n** and **0**, to a load called **rl** with a symbolic value of **load**, in ohms.

RM3's Example 9-8

Find the Norton equivalent of the circuit left of a-b; then find the current through R_L.



RM3's Example 9-8

SOLUTION

Let's first find the circuit equivalent:

```
type
s\th("e,1,0,24:r1,1,2,120:r2,2,0,280:j,2,0,560'm",2,0):{ino,req}
```

```
returns
{.36, 84.}
```

Correct. Now to the second part of the question. In order to find the current through R_L, we cannot use the load expressions, because now the load is not the only thing connected to the terminals of the equivalent: there is also a current source. We have to run a new simulation.

The fastest way is to start from the equivalent circuit description:

```
returns
"jN,0,n,iNo:rE,n,0,rEq:rL,n,0,L"
```

We change the value of the load to 168 Ω, and add the 180 mA source flowing from node **0** to node **n**. Then we run a dc simulation and ask for the current in the load:

```
type
s\dc("jN,0,n,iNo:rE,n,0,rEq:rL,n,0,168:j,0,n,180'm"):irL
```

returns

```
-.06
```

Correct: there is a current of 60 mA flowing through R_L from 0 to n.

Using the equivalent circuit description is meant to save you time. If you find it confusing to use, just don't use it.

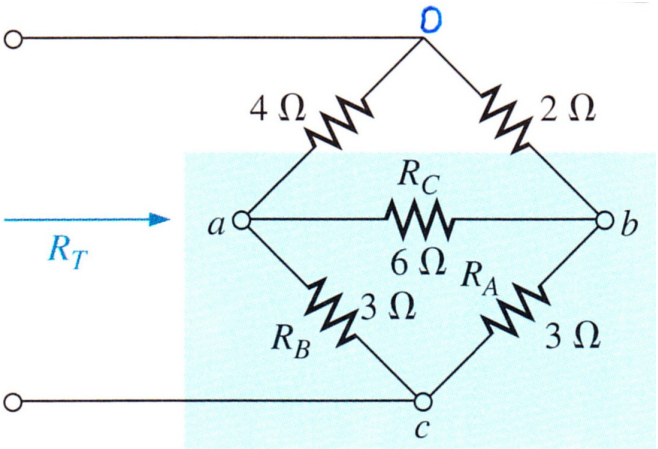
4.4 Instructive solved examples

Practice problems for resistive circuits

B11's Example 8.29

Calculate the equivalent resistance of the circuit shown.

You saw this circuit solved step by step in the walkthrough above; here it is again, in the compact form the rest of these problems use.



After I label the nodes, I describe the circuit. In this case, I store it in a variable.

type

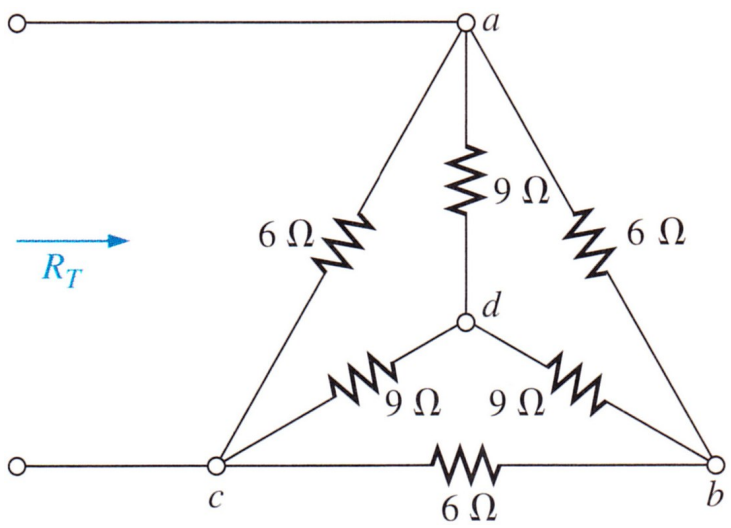
```
"r4,0,a,4:r2,0,b,2:r6,a,b,6:rb,a,c,3:ra,b,c,3"→cir
```

Run the **er** script, giving it as arguments the circuit and the nodes: `s\er(cir,0,c)` When prompted, choose DC as analysis type. Once it is done, evaluate `approx(req)` The value is **2.89** Ω.

This is correct. Below are many practice examples of this type.

B11's Example 8.30

Find the equivalent resistance of the circuit.

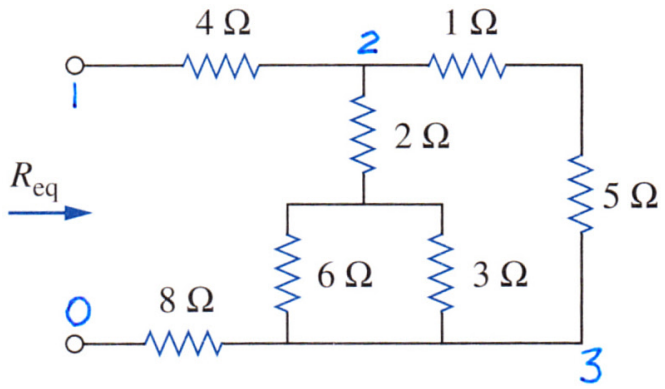


type
```
s\er("rac,a,c,6:rad,a,d,9:rab,a,b,6:rcd,c,d,9:rbc,b,c,6:rbd,b,d,9",a,c)
```

Choose DC. When *Done*, use `approx(req)` to find the equivalent resistance is **3.27** Ω.

AS2's Example 2.9

Find the equivalent resistance of the circuit.



We don't need to run a simulation for this. We can reduce it using s\pr.

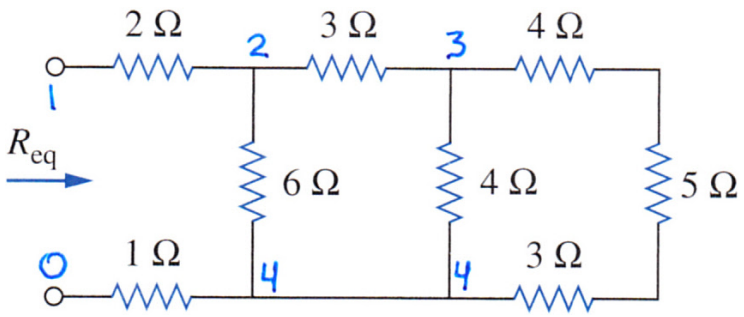
type
```
4+s\pr({1+5,2+s\pr({6,3})})+8
```

Evaluating approximately gives us the equivalent resistance: **14.4** Ω.

AS2's Practice Problem 2.9

Find the equivalent resistance of the circuit.

We don't need to run a simulation for this. We can reduce it using s\pr.

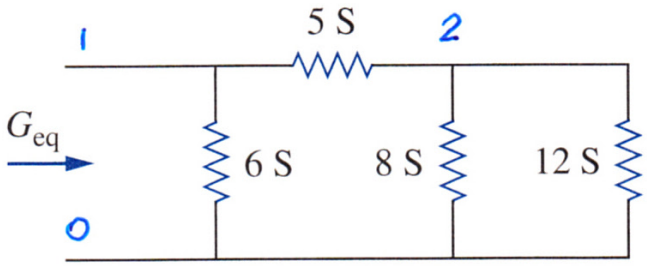



```
type
2+s\pr({6,3+s\pr({4,4+5+3})})+1
```

Evaluating approximately gives us the equivalent resistance: **6** Ω.

AS2's Example 2.11

Find the equivalent *conductance* of the circuit.



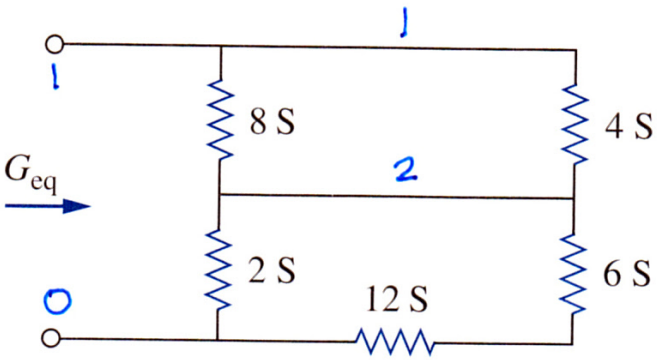
We don't need to run a simulation for this. We can reduce it using s\pr.

```
type
1/(s\pr({1/6,1/5+s\pr({1/8,1/12})}))
```

Evaluating approximately gives us the equivalent conductance: **10** S.

AS2's Practice Problem 2.11

Find the equivalent *conductance* of the circuit.



We don't need to run a simulation for this. We can reduce it using s\pr.

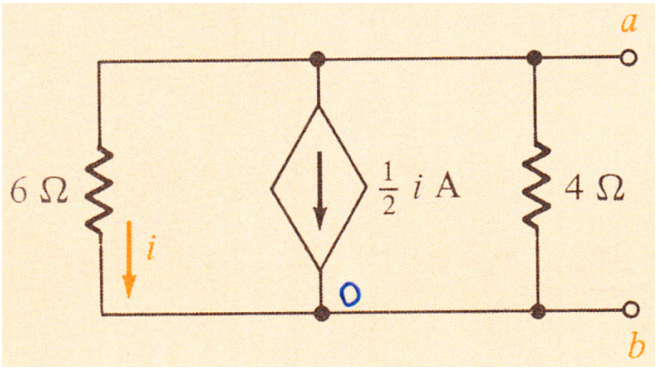
type
```
1/(s\pr({1/8,1/4})+s\pr({1/2,1/12+1/6}))
```

Evaluating approximately gives us the equivalent conductance: **4** S.

Examples with dependent sources

Bo2's Drill Problem 3.14

Find the equivalent resistance of the circuit.

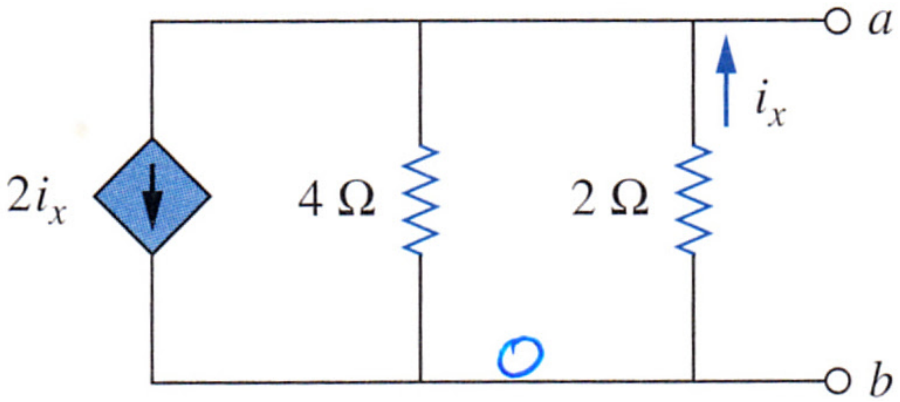


type
```
s\er("r4,a,0,4:ri,a,0,6:ji,a,0,iri/2",a,0)
```

Choose DC. Wait for *Done*. Evaluate `req` to find the equivalent resistance is **2** Ω.

AS2's Example 4.10

Find the equivalent resistance of the circuit.

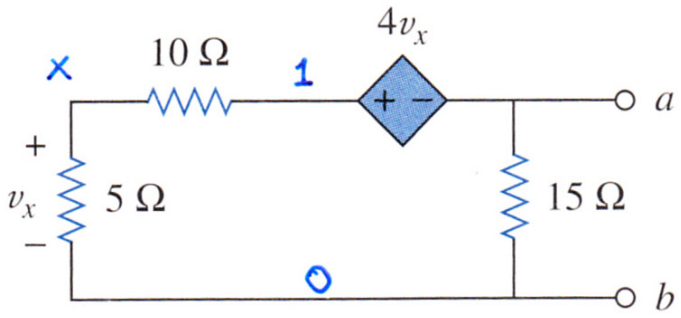


type
```
s\er("r4,a,0,4:rx,0,a,2:j,a,0,2irx",a,0)
```

Choose DC. Wait for *Done*. Evaluate `req`. The equivalent resistance is **-4** Ω. It may be surprising to have a negative resistance. This is the result of the dependent sources.

AS2's Practice Problem 4.10

Find the equivalent resistance of the circuit.

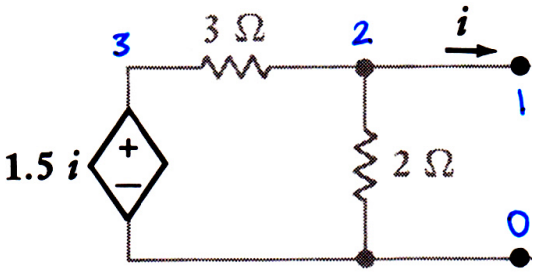



```
type
s\er("r15,a,0,15:e,1,a,4vrx:r10,1,x,10:rx,x,0,5",a,0)
```

Choose DC. Wait for *Done*. Evaluating `req` approximately, we find the equivalent resistance is **-7.5** Ω.

HK5's Figure 2-29

Find the equivalent resistance of the circuit.




```
type
s\er("e,3,0,1.5is:r3,3,2,3:r2,2,0,2:s,2,1",1,0)
```

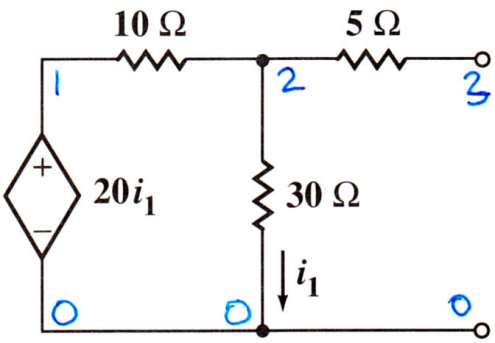
Choose DC. Wait for *Done*. Evaluate `req`. The equivalent resistance is **0.6** Ω.

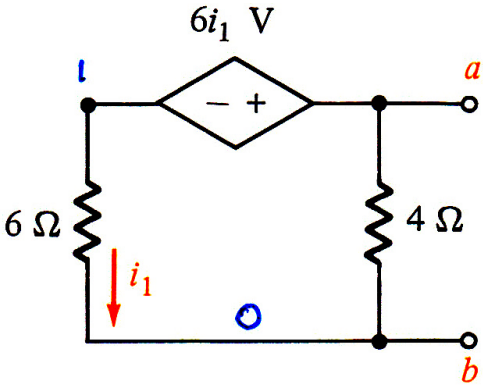
HK5's Drill Problem 2-9d

Find the equivalent resistance of the circuit.

```
type
s\er("r10,1,2,10:r5,2,3,5:r1,2,0,30:e,1,0,20ir1",3,0)
```

Choose DC. Wait for *Done*. Evaluate `req`. The equivalent resistance is **20** Ω.





Bo2's Example 3.12

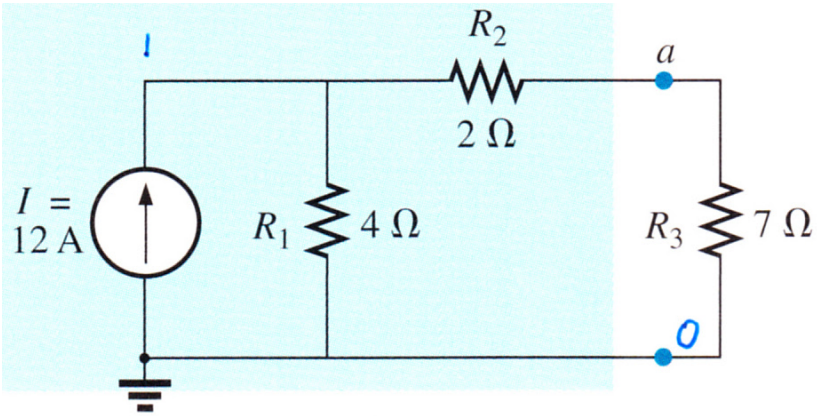
Find the equivalent resistance of the circuit.

```
type
s\er("r1,1,0,6:r4,a,0,4:e,a,1,6ir1",a,0)
```

Choose DC. Wait for *Done*. Evaluate `req`. The equivalent resistance is **3** Ω.

B11's Example 9.7

Find the Thévenin equivalent of the circuit, as seen from the R_3 resistor.

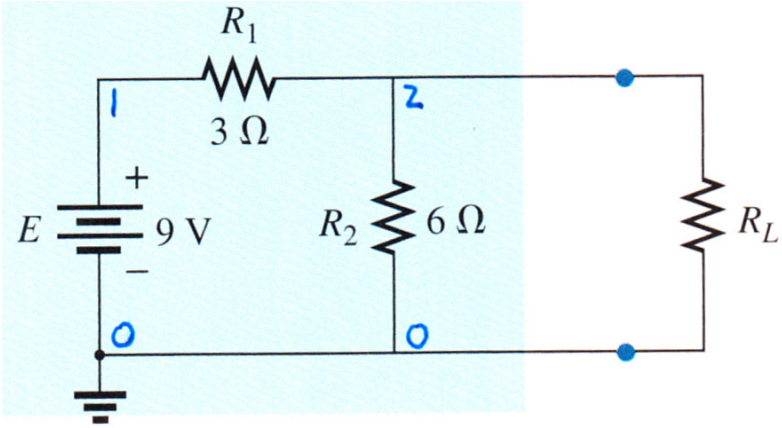


type
```
s\th("j,0,1,12:r1,1,0,4:r2,1,a,2",a,0)
```

Choose DC. Via `vth` we find V_{TH} = **48** V. Via `req` we find R_{EQ} = **6** Ω.

B11's Example 9.11

Find the Norton equivalent of the circuit, as seen from the R_L resistor.

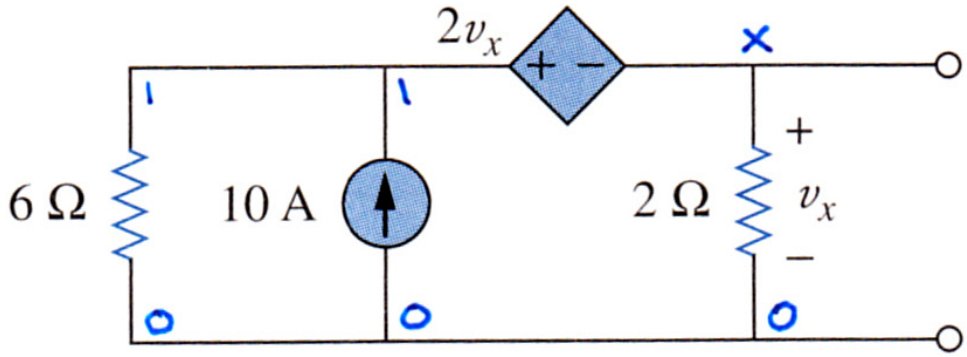


type
```
s\th("e,1,0,9:r1,1,2,3:r2,2,0,6",2,0)
```

Choose DC. Via `ino` we find I_{NO} = **3** A. Via `req` we find R_{EQ} = **2** Ω.

AS2's Practice Problem 4.12

Find the Norton equivalent of the circuit.

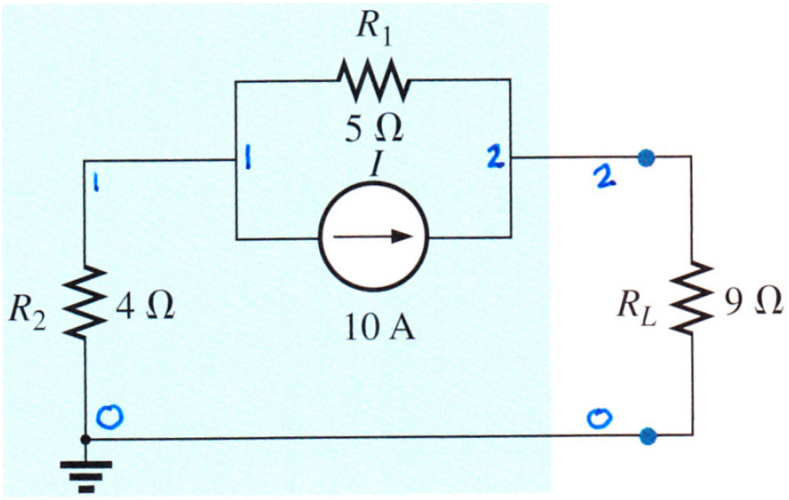


type
```
s\th("r6,1,0,6:j,0,1,10:r2,x,0,2:e,1,x,2vx",x,0)
```

Choose DC. Via `ino` we find I_{NO} = **10** A. Via `req` we find R_{EQ} = **1** Ω.

B11's Example 9.12

Find the Norton equivalent of the circuit, as seen from the R_L resistor.

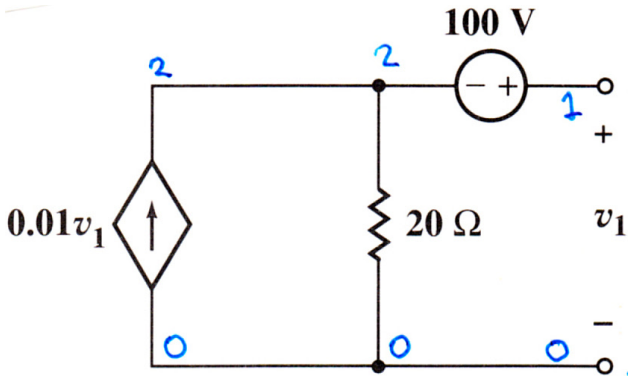


type
```
s\th("r2,1,0,4:r1,1,2,5:j,1,2,10",2,0)
```

Choose DC. Via `ino` we find I_{NO} = **5.56** A. Via `req` we find R_{EQ} = **9** Ω.

HK5's Drill Problem 2-8b

Find the Thévenin equivalent of the circuit.



type
```
s\th("j,0,2,0.01v1:r,0,2,20:e,1,2,100",1,0)
```

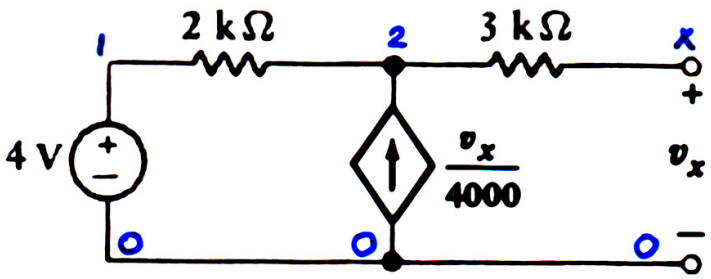
Choose DC. Via `vth` we find V_{TH} = **125** V. Via `req` we find R_{EQ} = **25** Ω.

HK5's Figure 2-27

Find the Thévenin equivalent of the circuit.

type
```
s\th("e,1,0,4:r2,1,2,2'k:r3,2,x,3'k:j,0,2,vx/4000",x,0)
```

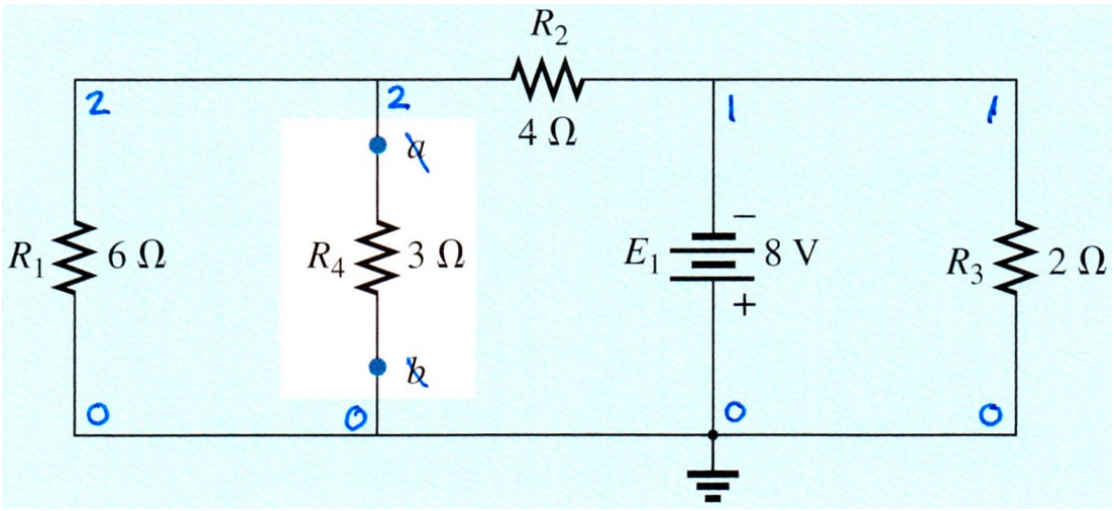
Via `vth` we find V_{TH} = **8** V. Via `req` we find R_{EQ} = **10** kΩ.



B11's Example 9.8

Find the Thévenin equivalent of the circuit, as seen from the R_4 resistor.

I ignore the textbook's decision to call the nodes **a** and **b**, since **b** is ground anyway.

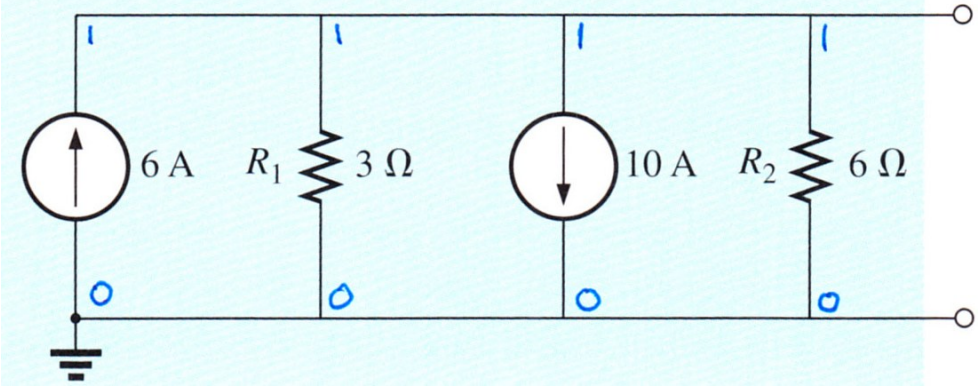



```
type
s\th("r1,2,0,6:r2,2,1,4:r3,1,0,2:e,0,1,8",2,0)
```

Choose DC. Via `vth` we get V_{TH}= **-4.8** V. Via `req` we get R_{EQ}= **2.4** Ω.

B11's Example 8.6

Find the Norton equivalent of the circuit.

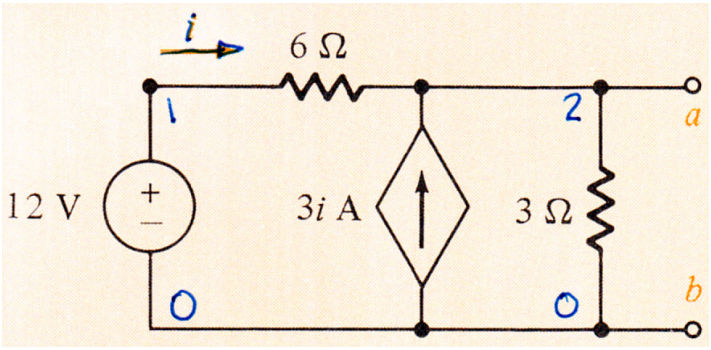


type
```
s\th("j1,0,1,6:r1,1,0,3:j2,1,0,10:r2,1,0,6",1,0)
```

Choose DC. Via `ino` we find I_{NO} = **-4** A. Via `req` we find R_{EQ} = **2** Ω.

Bo2's Drill Exercise 3.12

Find the Norton equivalent of the circuit.

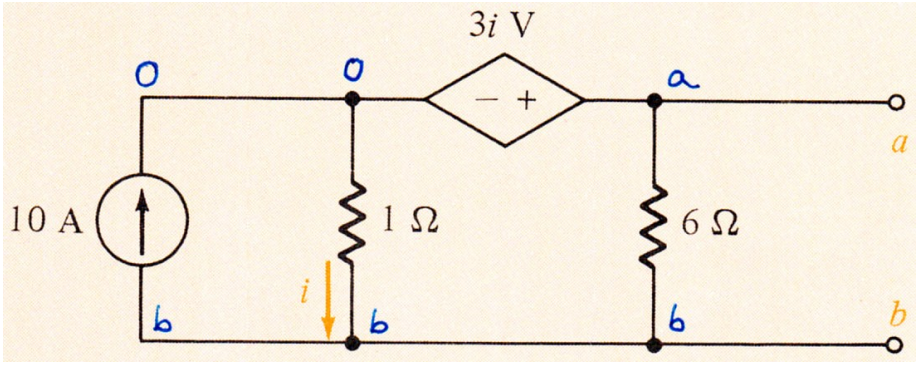


type
```
s\th("e,1,0,12:r6,1,2,6:j,0,2,3ir6:r3,2,0,3",2,0)
```

Choose DC. Via `ino` we find I_{NO} = **8** A. Via `req` we find R_{EQ} = **1** Ω.

Bo2's Drill Exercise 3.9

Find the Thévenin equivalent of the circuit.

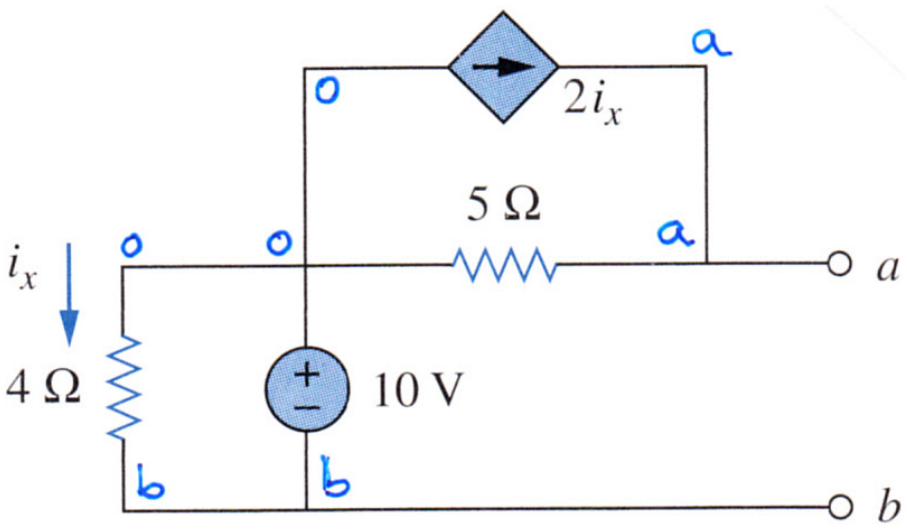


type
```
s\th("j,b,0,10:r1,0,b,1:e,a,0,3ir1:r6,a,b,6",a,b)
```

Choose DC. Via `vth` we find V_{TH} = **24** V. Via `req` we find R_{EQ} = **2.4** Ω.

AS2's Example 4.12

Find Norton equivalent of the circuit.

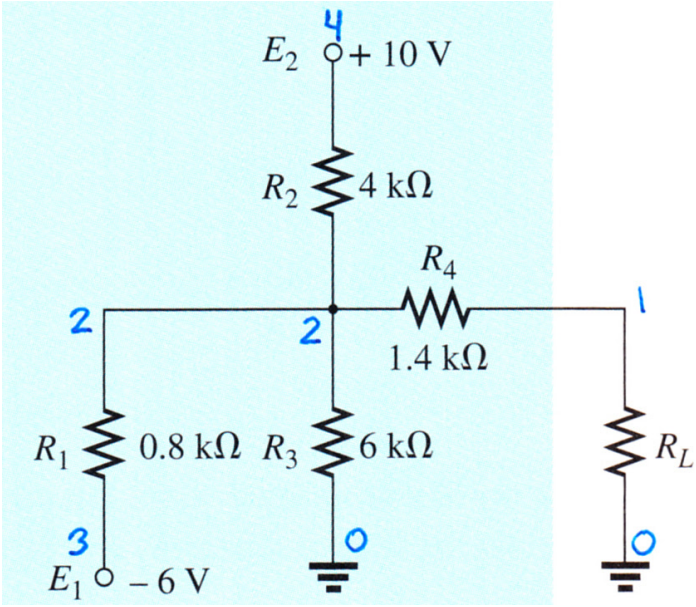


type
```
s\th("rx,0,b,4:e,0,b,10:r5,0,a,5:j,0,a,2irx",a,b)
```

Choose DC. Via `ino` we find I_{NO} = **7** A. Via `req` we find R_{EQ} = **5** Ω.

B11's Example 9.10 (Hidden source)

Find the Thévenin equivalent of the circuit.



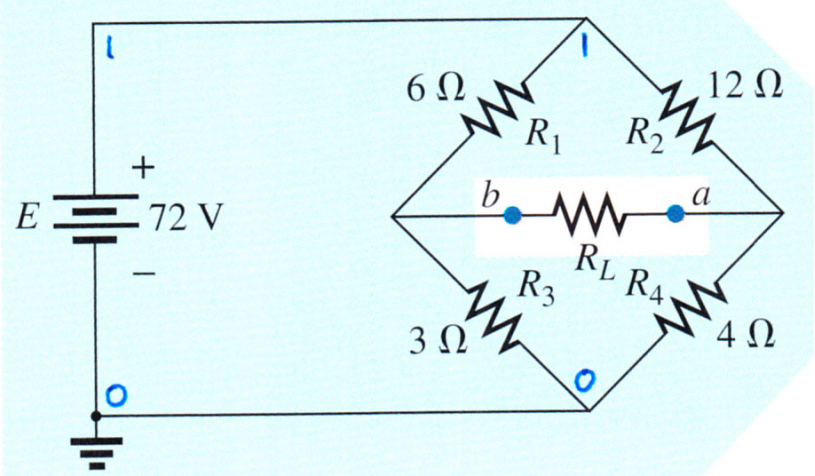
My solution is below:

type
```
s\th("e1,3,0,-6:e2,4,0,10:r1,2,3,0.8'k:
r2,2,4,4'k:r3,2,0,6'k:r4,2,1,1.4'k",1,0)
```

Choose DC. Via `vth` we find V_{TH} = **-3** V. Via `req` we find R_{EQ} = **2** kΩ.

B11's Example 9.9

Find the Thévenin equivalent of the circuit, as seen from the R_L resistor.

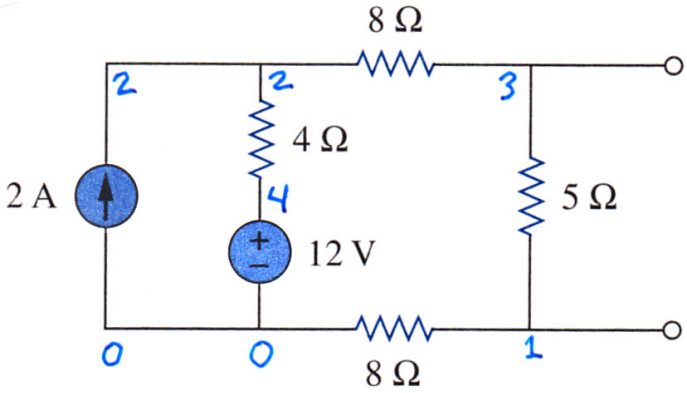


type
```
s\th("e,1,0,72:r1,1,b,6:r2,1,a,12:r3,0,b,3:r4,0,a,4",b,a)
```

Choose DC. Via `vth` we find V_{TH} = **6** V. Via `req` we find R_{EQ} = **5** Ω.

AS2's Example 4.11

Find the Norton equivalent of the circuit.

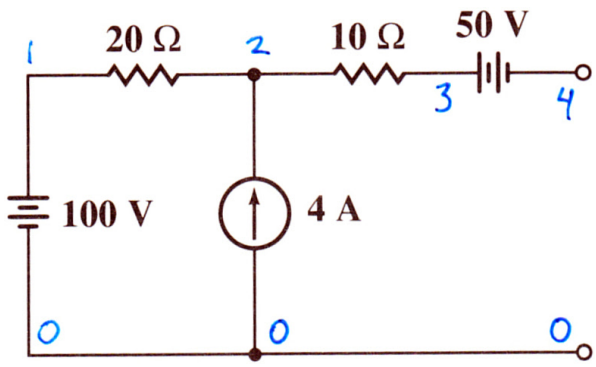


type
```
s\th("j,0,2,2:e,4,0,12:r1,2,4,4:r2,2,3,8:r3,0,1,8:r4,3,1,5",3,1)
```

Choose DC. Via `ino` we find I_{NO} = **1** A. Via `req` we find R_{EQ} = **4** Ω.

HK5's Drill Problem 2-8a

Find the Thévenin equivalent of the circuit.

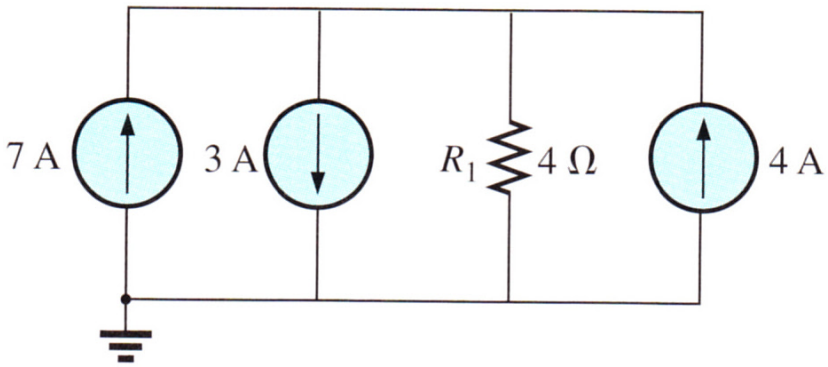


type
```
s\th("e1,1,0,100:r2,1,2,20:j,0,2,4:r1,2,3,10:e5,3,4,50",4,0)
```

Choose DC. Via `vth` we find V_{TH} = **130** V. Via `req` we find R_{EQ} = **30** Ω.

B11's Example 8.7

Find Norton equivalent of the circuit.

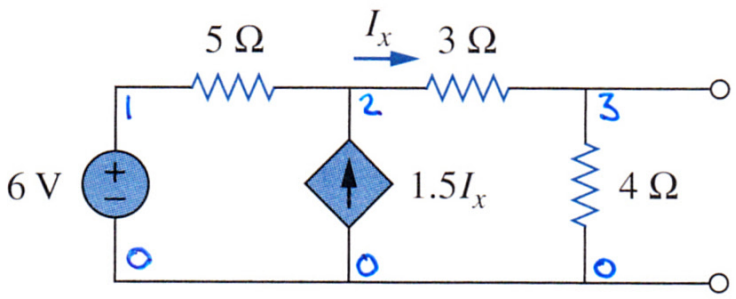


type
```
s\th("j7,0,1,7:j3,1,0,3:r1,1,0,4:j4,0,1,4",1,0)
```

Choose DC. Via `ino` we find I_{NO} = **8** A. Via `req` we find R_{EQ} = **4** Ω.

AS2's Practice Problem 4.9

Find the Thévenin equivalent of the circuit.

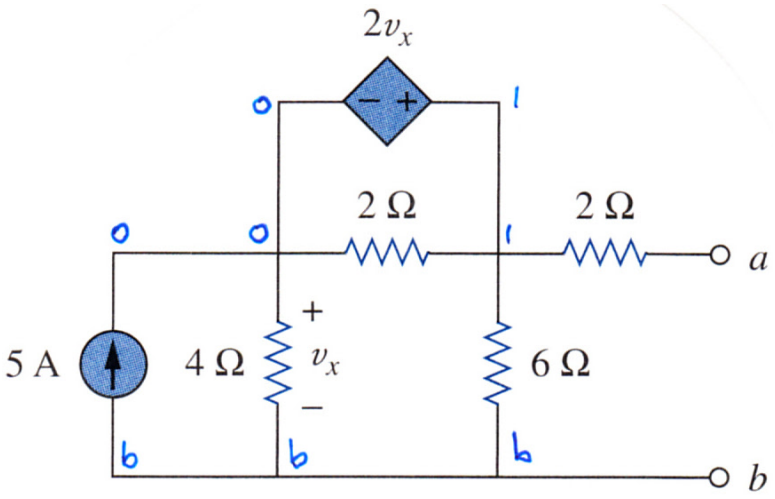


type
```
s\th("e,1,0,6:r5,1,2,5:rx,2,3,3:j,0,2,1.5irx:r4,3,0,4",3,0)
```

Choose DC. Via `vth` we find V_{TH} = **5.33** V. Via `req` we find R_{EQ} = **0.44** Ω.

AS2's Example 4.9

Find the Thévenin equivalent of the circuit.

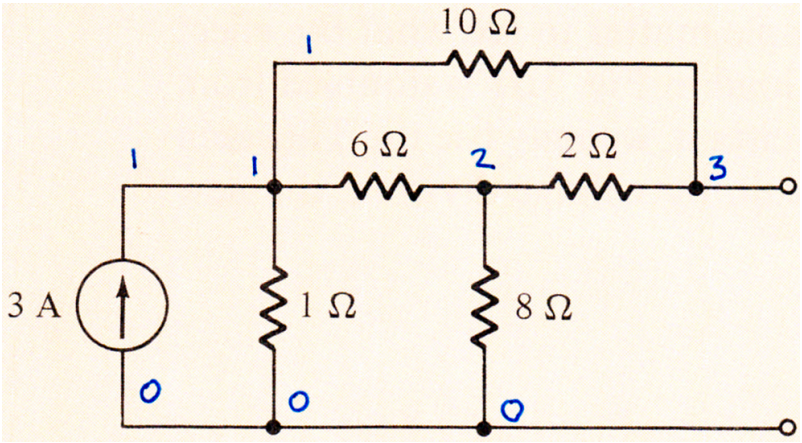


type
```
s\th("j,b,0,5:rx,0,b,4:r1,0,1,2:r2,1,b,6:r3,1,a,2:e,1,0,2vrx",a,b)
```

Choose DC. Via `vth` we find V_{TH} = **20** V. Via `req` we find R_{EQ} = **6** Ω.

Bo2's Drill Exercise 3.8

Find the Thévenin equivalent of the circuit.

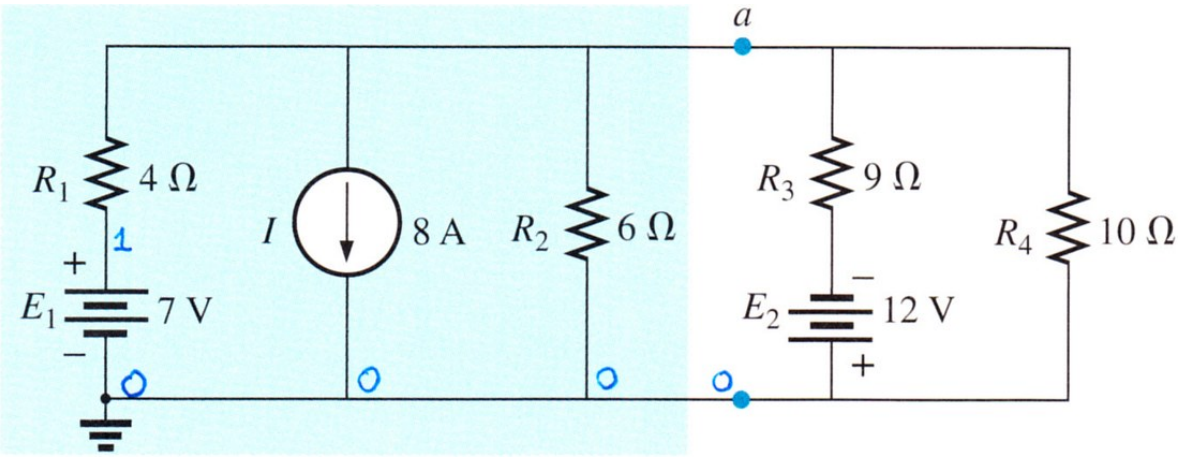


type
```
s\th("j,0,1,3:r1,0,1,1:r6,1,2,6:r10,1,3,10:r8,2,0,8:r2,2,3,2",3,0)
```

Choose DC. Via `vth` we find V_{TH} = **2** V. Via `req` we find R_{EQ} = **4** Ω.

B11's Example 9.13

Find Norton equivalent of the shaded part of the circuit.




```
type
s\th("e1,1,0,7:r1,1,a,4:j,a,0,8:r2,a,0,6",a,0)
```

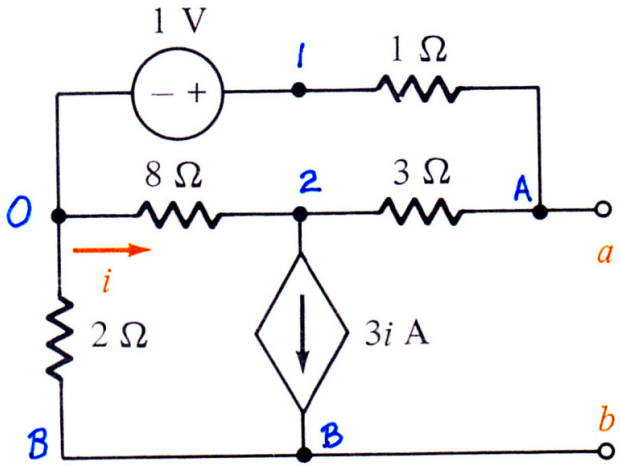
Choose DC. Via `ino` we find I_{NO} = **-6.25** A. Via `req` we find R_{EQ} = **2.4** Ω.

Tricky problems

Some circuit theory books – and some professors – find it entertaining or instructive to surprise unsuspecting students with tricky problems, like the two we solve below.

Bo2's Example 3.11 (Tricky)

Find the Norton equivalent of the circuit.



Since you do not know beforehand that this is a tricky problem, you go for the usual:

```
type
s\th("r2,0,b,2:r8,0,2,8:r3,2,a,3:r1,1,a,1:e,1,0,1:j,2,b,3ir8",a,b)
```

You choose DC, press Enter and wait. Symbulator reports the calculator was unable to solve the equations. This often means there is a division by zero somewhere.

Clean the variables from the MAIN folder and try again, this time – following Symbulator's advise –using a symbolic value. We chose to use **x** instead of **3** in the dependent source.

```
type
s\th("r2,0,b,2:r8,0,2,8:r3,2,a,3:r1,1,a,1:e,1,0,1:j,2,b,x*ir8",a,b)
```

Now it solves. The expression for `ino` is fine, but the one for `req`,

$$\frac{9x - 35}{4(x - 3)}$$

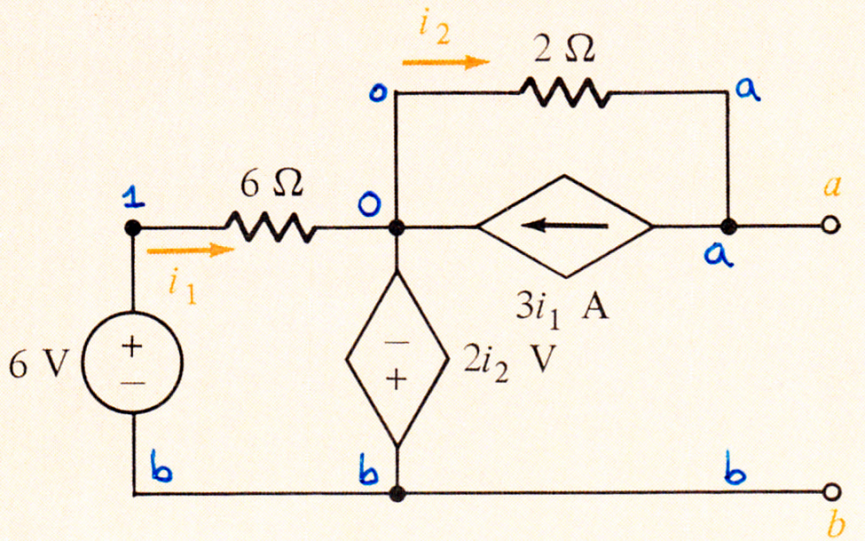
will divide by zero at $x = 3$ — the very value we replaced.

Via `Define x=3: {ino,req}` we find that I_{NO} = **1** A, and R_{EQ} is undefined.

This means the equivalent resistance is, for practical purposes, infinite. Your idea of fun, right?

Bo2's Drill Exercise 3.13 (Tricky)

Another tricky one. Find the Norton equivalent of the circuit.



As in the previous example, we get an error if we simulate using **2** and **3** in the dependent sources. So, we use **2x** and **3x** instead, where **x** will be later defined as 1.

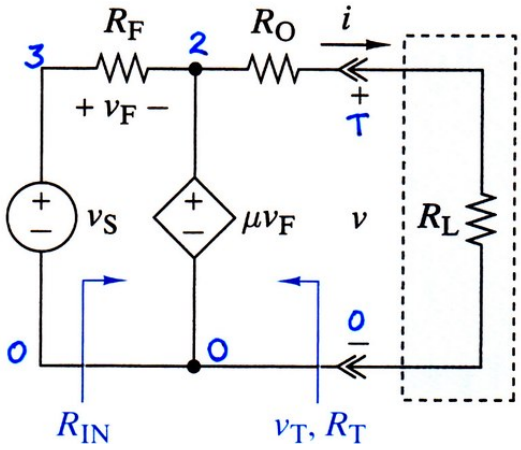
```
type
s\th("ei,1,b,6:ed,b,0,2*x*ir2:r1,1,0,6:r2,0,a,2:jd,a,0,3*x*ir1",a,b)
```

Exploring the answers, we see that the denominator of the expression for `req`, (x-1), results in a division by zero at x = 1.

Via `1→x:{ino,req}` we find that I_{NO} = **-3**A, and R_{EQ} is undefined or, for practical purposes, infinite.

TR5's Exercise 4-6 (Symbolic)

Find the input resistance and output Thévenin equivalent circuit of the circuit



$$R_{IN} = (1 + \mu)R_F$$

$$v_T = \frac{\mu}{\mu + 1} v_S$$

$$R_T = R_O$$

First let's find the input resistance, R_{IN}, i.e. the resistance as seen by the v_S source. We use μ for the constant in the dependent source.

```
type
s\dc("ei,3,0,vs:rf,3,2,rf:ro,2,t,ro:ed,2,0,μ*vrf"):rei
```

We get `rf*(μ+1)`, which is correct. The textbook's answers are shown right of the circuit schematic. Now we find the output Thévenin equivalent circuit as seen by R_L.

```
type
s\th("ei,3,0,vs:rf,3,2,rf:ro,2,t,ro:ed,2,0,μ*vrf",t,0):{vth,req}
```

We get

$$vth = \frac{vs\,\mu}{\mu + 1}$$

$$req = ro$$

which is correct, as can be seen in the textbook's answers for v_T and R_T, shown right of the circuit schematic above.

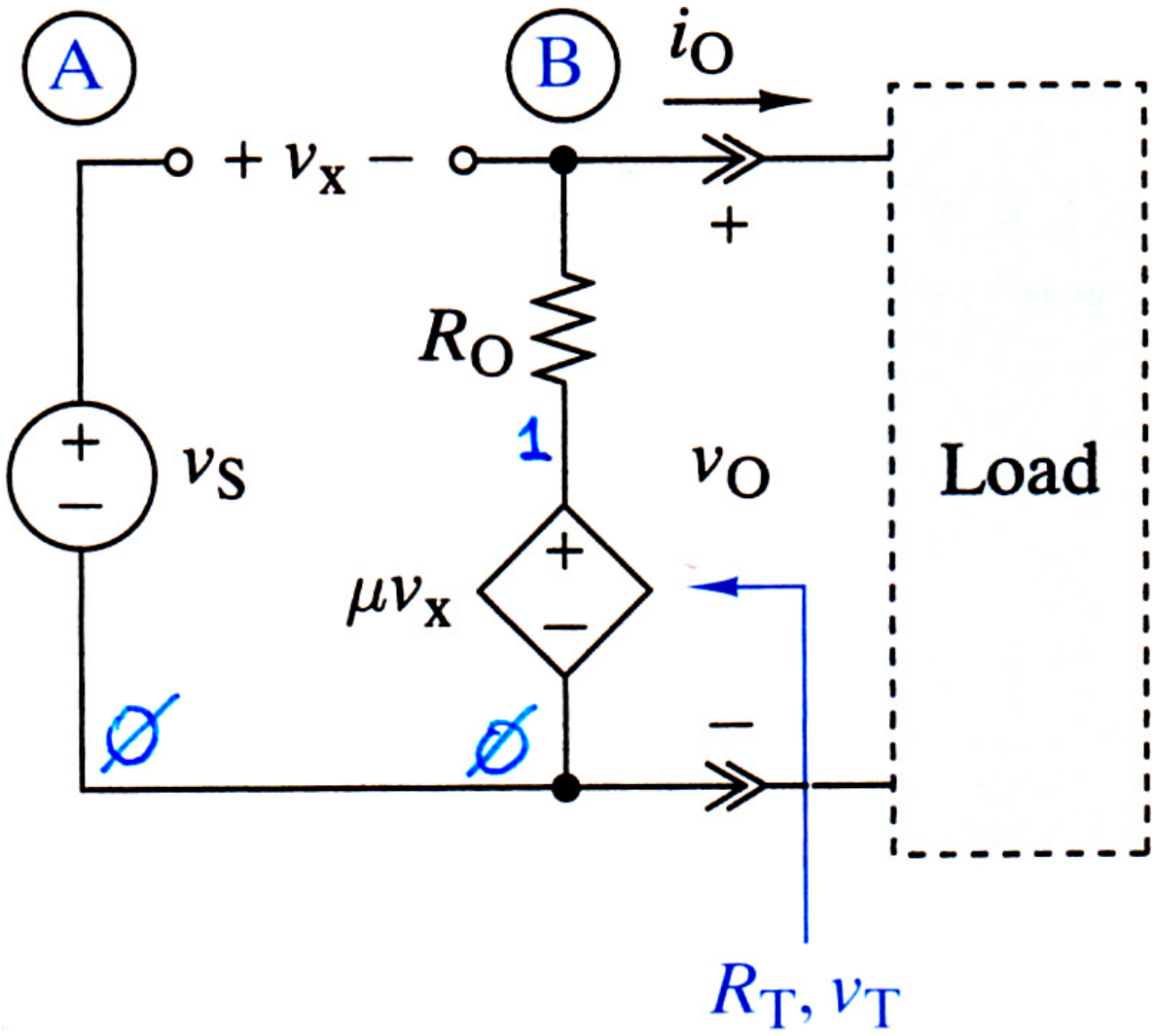
TR5's Example 4-8 (Symbolic)

Find the Thévenin equivalent as seen by the load.

My answer starts by defining $\mathbf{v}_x$ as **va-vb**. This is done thus:

```
type
Define vx=va-vb
```

Now I run the th script, with this circuit description:




```
type
s\th("ei,a,0,vs:ed,1,0,μ*(vx):ro,b,1,ro",b,0):
{vth,req}
```

The answers we get,

$$vth = \frac{vs\,\mu}{\mu + 1}$$

$$req = \frac{ro}{\mu + 1}$$

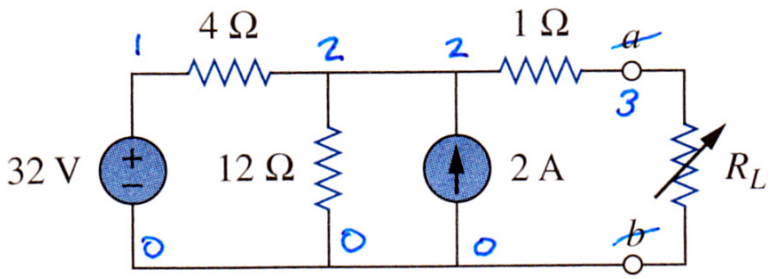
are correct, as can be seen by comparing them to those in the book:

$$v_T = \frac{\mu v_S}{\mu + 1} \quad \text{and} \quad R_T = \frac{R_O}{\mu + 1}$$

I am not sure there is any other circuit simulator for calculators that can do this.

AS2's Example 4.8

Find the Thévenin equivalent of the circuit shown to the left of terminals a-b. Then find the current through R_L = 6, 16 and 36Ω.



My solution:

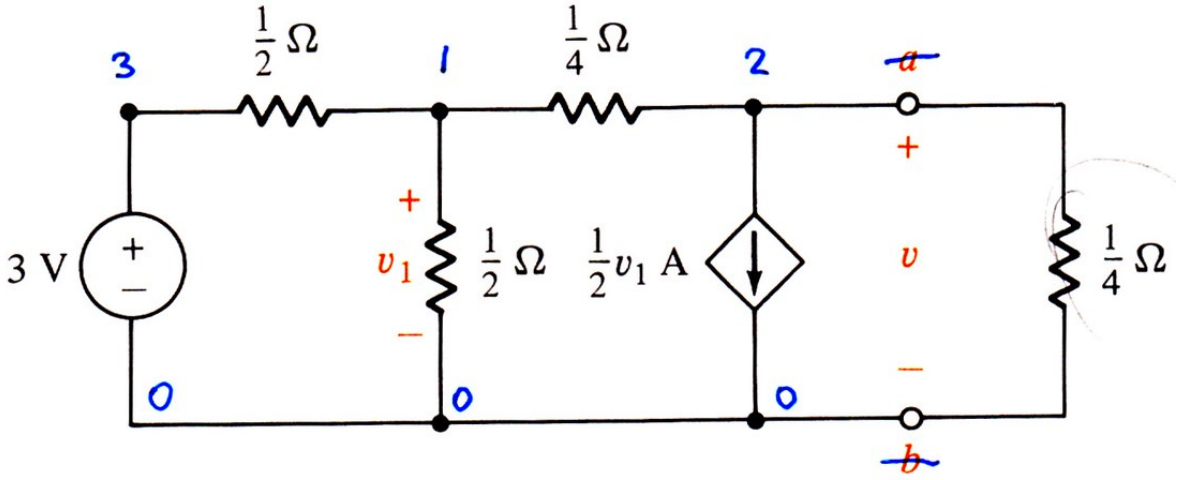
```
type
s\th("e,1,0,32.:r4,1,2,4:r12,2,0,12:j,0,2,2:r1,2,3,1",3,0):
{vth,req,irL|Load=6,irL|Load=16,irL|Load=36}
```

Choose DC. Answer Y when offered the load formulas.

The answer, **{30.,4.,3.,1.5,.75}**, is correct.

Bo2's Example 3.10

Find the Norton equivalent of the circuit left of the a-b terminals, and then find the voltage drop and the current through the ¼ Ω resistor. My one-line solution:



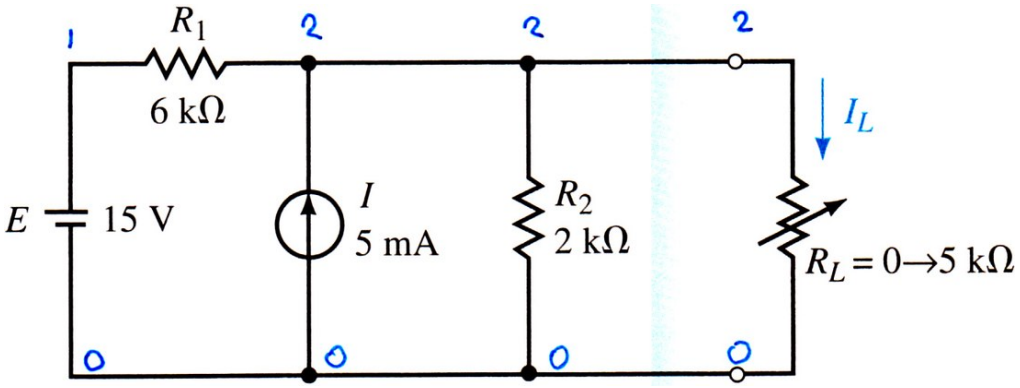

```
type
s\th("e,3,0,3:r31,3,1,1/2:r10,1,0,1/2:r12,1,2,1/4:j,2,0,v1/2",2,0):
{ino,req,vrL|Load=1/4,irL|Load=1/4}
```

Choose DC. Answer Y when offered the load formulas.

The book gives the answers as fractions. We get it right: **{21/8, 4/9, 21/50, 42/25}**.

RM3's Example 9-7

Find the Norton equivalent of the circuit external to R_L. Then determine the load current I_L when R_L = 0 Ω, 2 kΩ and 5 kΩ. My one-line solution:



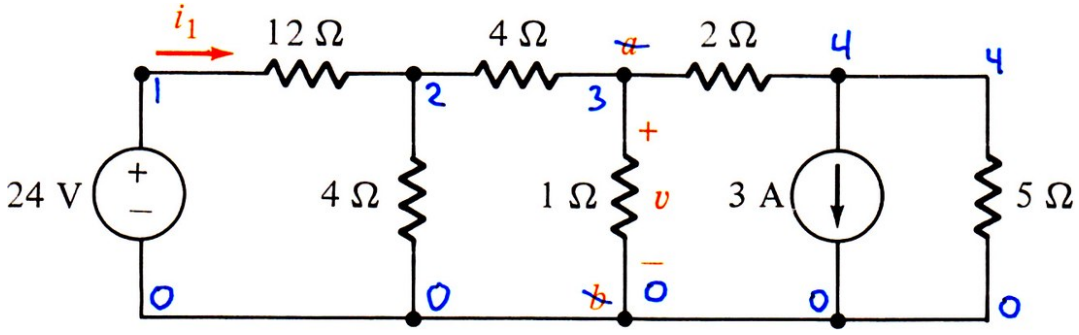

```
type
s\th("e,1,0,15.:r1,1,2,6'k:j,0,2,5'm:r2,2,0,2'k",2,0):
{ino,req, irL|Load=0,irL|Load=2000,irL|Load=5000}
```

Choose DC. Answer Y when offered the load formulas.

The answer, **{.0075, 1500., .0075, .00321, .00173}**, is correct.

Bo2's Example 3.5

Find the Norton equivalent of the circuit external to the 1Ω resistor. Then determine the voltage drop across this 1Ω resistor.



My solution:

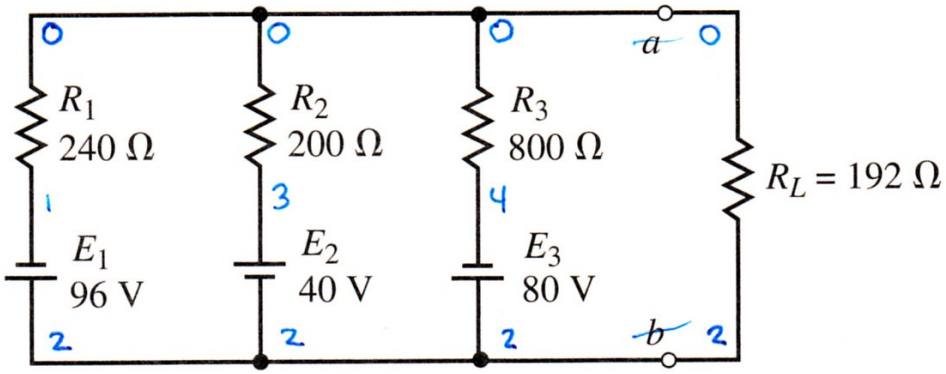
```
type
s\th("e,1,0,24:r12,1,2,12:r20,2,0,4:r23,2,3,4:r34,3,4,2:
j,4,0,3:r40,4,0,5",3,0):{ino,req,vrL|Load=1}
```

Choose DC. Answer Y when offered the load formulas.

The answer, **{-9/7,7/2,-1}**, is correct.

RM3's Example 9-13

Use Millman's Theorem to simplify the circuit left of a-b so that there is only one voltage and one resistor. Then find the current in the load resistor R_L.



I don't know Millman's Theorem, but we can find the Thévenin equivalent with Symbulator.

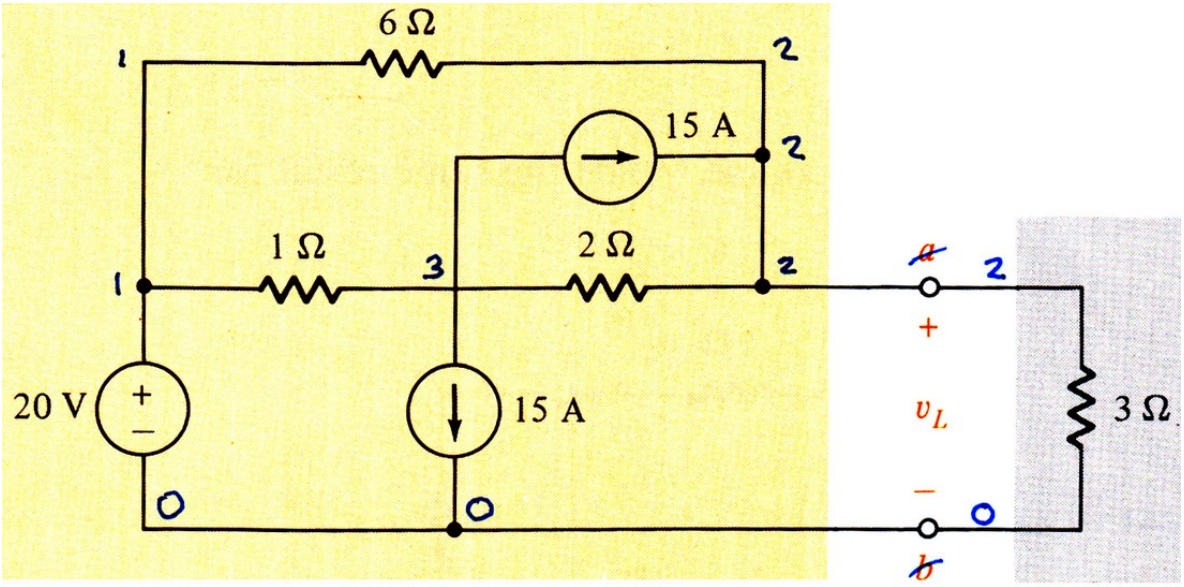
```
type
s\th("r1,0,1,240.:e1,2,1,96:r2,0,3,200:e2,3,2,40:
r3,0,4,800:e3,2,4,80",2,0):{vth,req,irL|Load=192}
```

Choose DC. Answer Y when offered the load formulas.

The answer, **{28.8,96.,.1}**, is correct.

Bo2's Example 3.7

Find the Thévenin equivalent for the circuit left of a-b. Then find the voltage across the 3 Ω resistor, and also if it was 6 Ω. My answer:



type

```
s\th("e,1,0,20:r6,1,2,6:r1,1,3,1:r2,3,2,2:j32,3,2,15:j30,3,0,15",2,0):
{vth,req,vrL|Load=3,vrL|Load=6.}
```

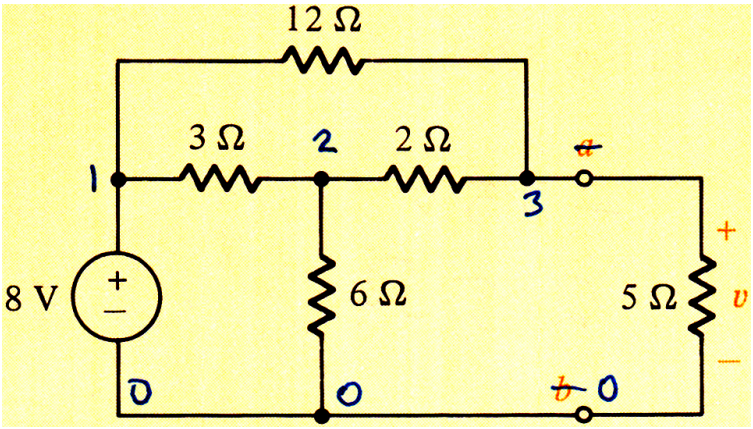
Choose DC. Answer Y when offered the load formulas.

The answer we find, **{30,2,18,22.5}**, is correct.

Bo2's Drill Exercise 3.7

Find the Thévenin equivalent for the circuit left of a-b. Then find the voltage *v*.

My answer:



type

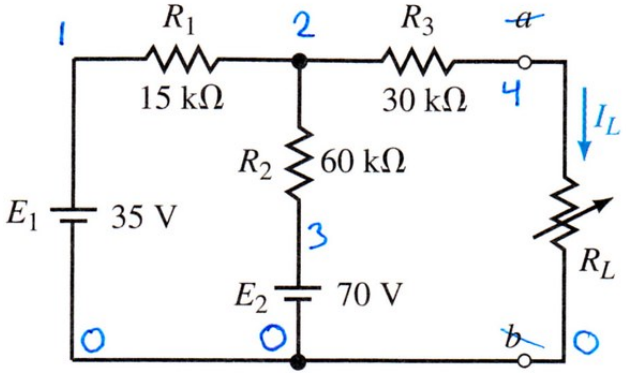
```
s\th("e,1,0,8:r3,1,2,3:r12,1,3,12:r6,2,0,6:r2,2,3,2",3,0):
{vth,req,vrL|Load=5.}
```

Choose DC. Answer Y when offered the load formulas.

The answer, **{6,3,3.75}**, is correct.

RM3's Practice Problem 9.5

Find the Norton equivalent external to R_L in the circuit. Solve for the current I_L when $R_L = 0$, 10 kΩ, 50 kΩ, and 100 kΩ.



Answers

$R_N = 42\ \text{k}\Omega$, $I_N = 1.00\ \text{mA}$

For $R_L = 0$: $I_L = 1.00\ \text{mA}$

For $R_L = 10\ \text{k}\Omega$: $I_L = 0.808\ \text{mA}$

For $R_L = 50\ \text{k}\Omega$: $I_L = 0.457\ \text{mA}$

For $R_L = 100\ \text{k}\Omega$: $I_L = 0.296\ \text{mA}$

My solution:

```
type
s\th("e1,1,0,35.:r1,1,2,15'k:r2,2,3,60'k:e2,3,0,70:r3,2,4,30'k",4,0):
{ino,req,irL|Load=0,irL|Load=1e4,irL|Load=5e4,irL|Load=1e5}
```

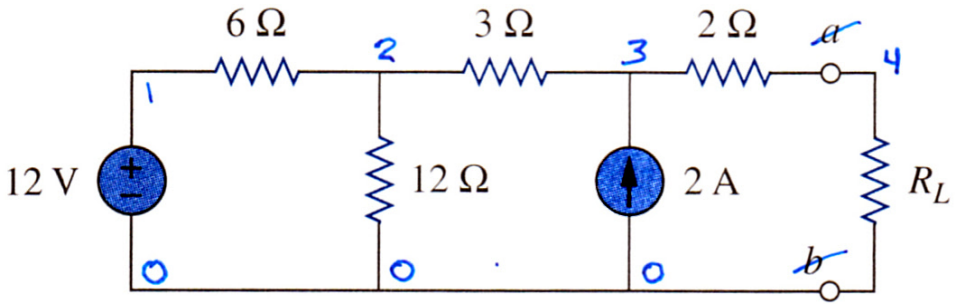
Choose DC. Answer Y when offered the load formulas.

The answer, **{.001,42000,.001,.000808,.000457,.000296}**, is correct.

Power transfer problems

AS2's Example 4.13

Find the R_L value for maximum power transfer and the maximum power transferred.



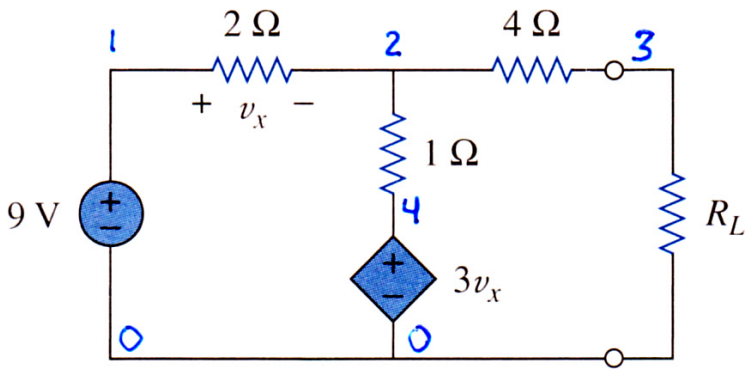
R_L for maximum transfer is `req`. The maximum power is in `pmax`. My solution:

```
type
s\th("e,1,0,12:r6,1,2,6:r12,2,0,12:r3,2,3,3:j,0,3,2:r2,3,4,2",4,0):
approx({req,pmax})
```

Choose DC. You can answer N when asked about the load equations. The answer, **{9.,13.44**}, is correct: the maximum transfer of power occurs when the load is **9**Ω. At this point, the power transferred is **13.44**W. Now let's solve another one.

AS2's Practice Problem 4.13

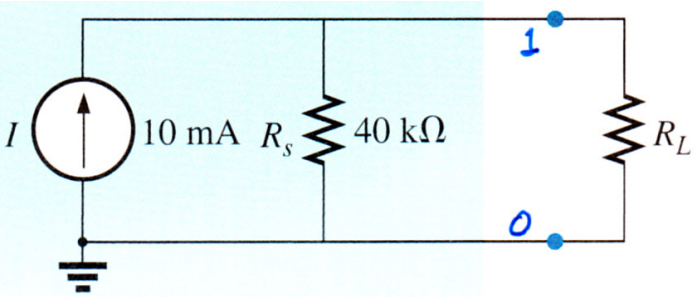
Find the R_L value for maximum power transfer and the maximum power transferred.



```
type
s\th("ei,1,0,9:rx,1,2,2:r1,2,4,1:ed,4,0,3vrx:r4,2,3,4",3,0):
approx({req,pmax})
```

Choose DC. You can answer N. The logic of this problem is identical to the previous one. The answer is **{4.22,2.901}**.

B11's Example 9.15



a. Find the load resistance that will result in maximum power transfer to the load, and find the maximum power delivered.
b. If the load were changed to 68 kΩ, would you expect a fairly high level of power transfer to the load based on the results of part (a)? What would the new power level be? Is your initial assumption verified?
c. If the load were changed to 8.2 kΩ, would you expect a fairly high level of power transfer to the load based on the results of part (a)? What would the new power level be? Is your initial assumption verified?

My one-line solution to all three questions is presented below.

```
type
s\th("j,0,1,10'm:rs,1,0,40'k",1,0):
{req,pmax,prl|load=68000.,prl|load=8200.}
```

Choose DC. Answer Y about the load formulas. The answer we obtain, **{40000, 1, .93, .57}**, is correct. Let's deconstruct it.

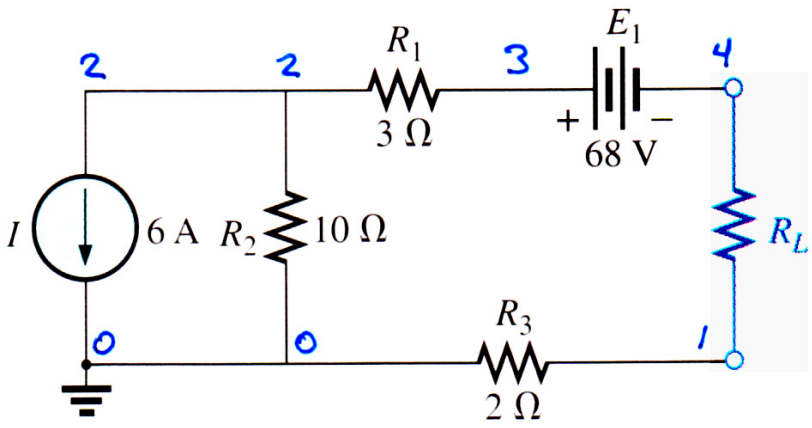
Part (a) is answered by the first two values: a 40 kΩ resistor as load would receive 1W power. Since this is the maximum – this is the most that any load could receive ever.

Part (b) is answered by the third value. Making use of the variable **prL**, which contains the power delivered by the circuit equivalent to the load, as a function of the load value **L**, we find that a load of 68 kΩ receives .93W, which is less than the maximum.

Part (c) is answered in similar manner by the fourth value. A load of 8.2 kΩ receives .57W, which is less than the maximum. Any resistance other than 40 kΩ gets less power.

B11's Example 9.17

Find the R_L value for maximum power transfer and the maximum power transferred.




```
type
s\th("j,2,0,6:r2,2,0,10:r1,2,3,3:r3,0,1,2:e,3,4,68",4,1):
approx({req,pmax})
```

Choose DC. You can answer N. The answer is **{15., 273.07}**. Let's now see one that is a little different.

LESSON 5

Operational amplifiers

Learn how to describe an ideal *operational amplifier* (Op Amp) using the **o** element. And practice what you learn in a variety of solved op-amp problems, both easy and hard.

In this lesson, you will learn how to describe an ideal *operational amplifier* — also called an Op Amp — using the **o** element. And you will practise what you learn in a variety of solved op-amp problems, both easy and hard.

5.1 How to describe an ideal operational amplifier

There are many types of operational amplifiers. Symbulator can simulate the ideal, linear type.

> ⓘ NOTE
> **Describing op amps**
> In Symbulator, an ideal op amp is described using four bits of information, separated by commas: first, the name of the op amp, which must start with the letter **o**; then, the names of the two input nodes, separated by a comma (the polarity doesn't matter, you can give these input nodes in any order); and finally, the name of the output node.
> For example, an ideal op amp called **o1**, with input nodes **2** and **3** and output node **5**, is described `o1,2,3,5`. And one called **o**, with input nodes **p** and **n** and output node **o**, is `o,p,n,o`

What answers do you get

For each ideal op amp in a circuit, Symbulator will store the following answers:

- The current through the output node, flowing from the output node outwards. For an op amp called **o**, that is `io`.
- The power consumed by the visible part of the op amp. For an op amp called o2, `po2`; the power delivered is the negative of that.
- And, as with every element's nodes, the voltage of each of its nodes with reference to ground. For node **o**, `vo`.

Livin' on the edge

Since I was born to be bad, I like to play with node names to get the answers as close to the book as possible. So do not be surprised if I call one op amp "o" and name its output node "o" as well. That way I can ask for `vo` and `io` and get the output voltage and current. But realise this: `vo` asks for the voltage in node **o**, while in `io` the o stands for the element **o**.

You, too, can be bad to the bone, and live dangerously with your node naming, as long as you anticipate what variables Symbulator will create to store the answers. I tell you this because there are limits to what you can get away with.

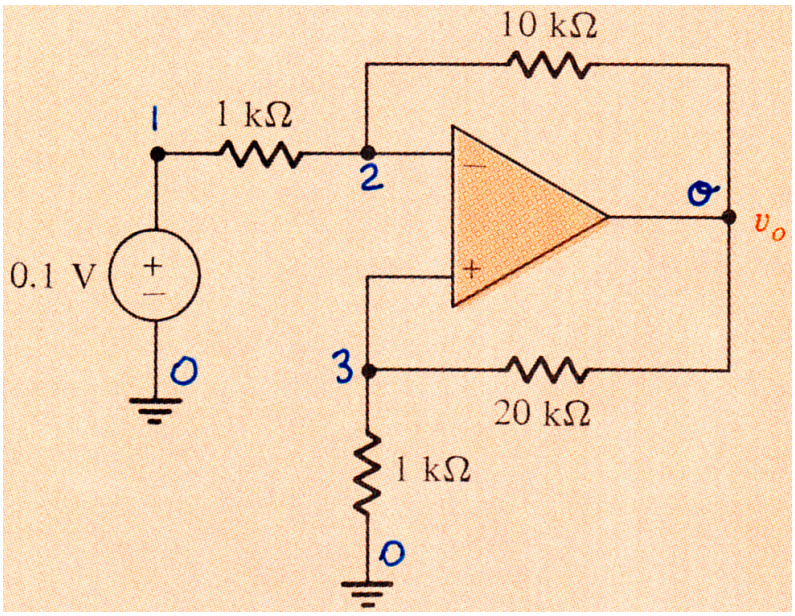
For example, when naming nodes, remember that for a node called #, Symbulator creates a variable v# for its voltage. So never give a node the same name as an element that has a voltage drop of its own: r, e, j, c, l. If in doubt, play it safe and give every node a name of its own, different from any other node and from any element.

In this example, we got away with it because op amps are not that type of element. Symbulator does not save a voltage drop for op amps in a variable called vo, so there is no problem having a node called **o** in the same circuit.

> CAUTION
>
> **Never describe a source as e,#,0,v#**
>
> If you describe a source as `e1,1,0,v1`, Symbulator will define the voltage of node **1**, namely v1, as having the value you provided for the source, which you declared to be, wait for it... v1! The resulting equation, v1 = v1, is discarded as trivial by the calculator, and this leaves Symbulator one equation short.

Bo2's Drill Exercise 3.2

For the op-amp circuit shown, find vo and the power absorbed by the op amp.



Bo2's Drill Exercise 3.2

SOLUTION

My solution:

```
type
s\dc("e,1,0,.1:r12,1,2,1'k:r2o,2,o,10'k:r30,3,0,1'k:r3o,3,o,20'k:o,3,2,
o")
```

We ask for the values of the variables `vo` and `po`:

```
type
{vo,po}
```

```
returns
{-2.1, -.00063}
```

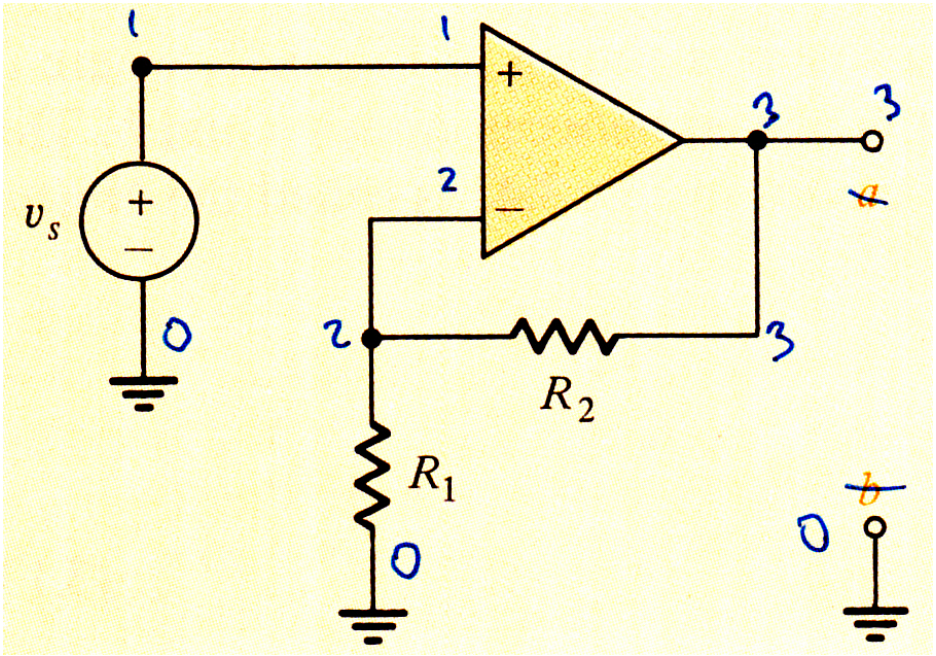
Which are correct. The voltage in node **o** is –2.1 V, and the op amp is absorbing –.63 mW, which is to say it is delivering .63 mW of power to the circuit.

5.2 Instructive Op Amp examples

Solved Op Amp problems

Bo2's Drill Exercise 3.11 (Thévenin)

Find the Thévenin equivalent.




```
type
s\th("e,1,0,vs:r1,2,0,r1:r2,2,3,r2:o,1,2,3",3,0)
```

The **th** script tells us it found the Thévenin voltage, but could not find the Norton current. This is not a surprise, since an ideal op amp has zero output resistance and a fixed voltage, an infinite current when short-circuited. So the Thévenin equivalent is given by V_{TH} and no resistance (or R_{EQ} = **0**Ω). Evaluating `vth` results in

((r1+r2) vs)/r1

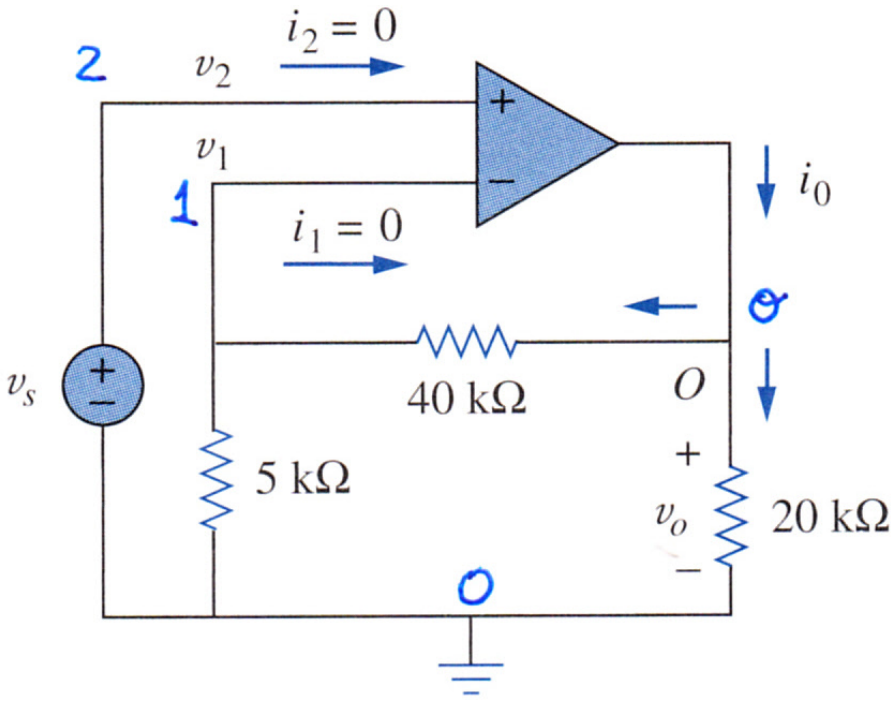
This is correct, as can be seen by comparing it to the book's answer, shown below.

$$\left(1 + \frac{R_2}{R_1}\right) v_s$$

The Thévenin resistance, as explained above, is 0Ω.

AS2's Example 5.2

Find v_o and i_o.



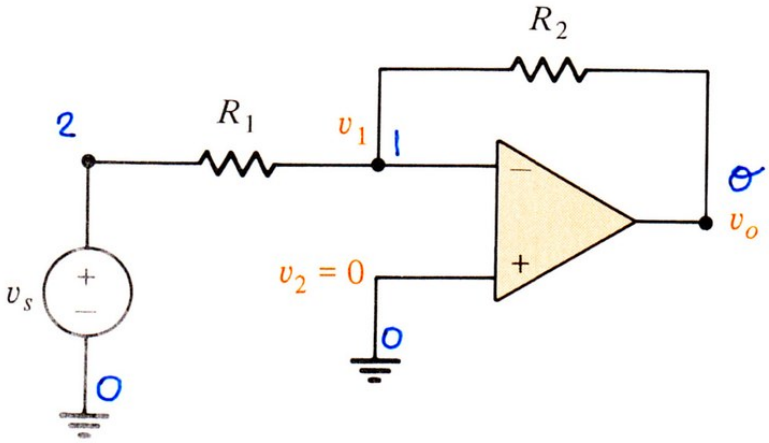
type
```
s\dc("e,2,0,1.:r5,1,0,5'k:r4,1,o,40'k:r2,o,0,20'k:o,2,1,o"):{vo,io}
```

The answer, **{9.,.00065}**, is correct: v_o is **9**V and i_o is **0.65**mA

Bo2's Figure 3.3 (Inverting)

Find the gain of the overall circuit, v_o/v_S.

type
```
s\dc("e,2,0,vs:r1,2,1,r1:r2,1,o,r2:o,1,0,o"):vo/vs
```

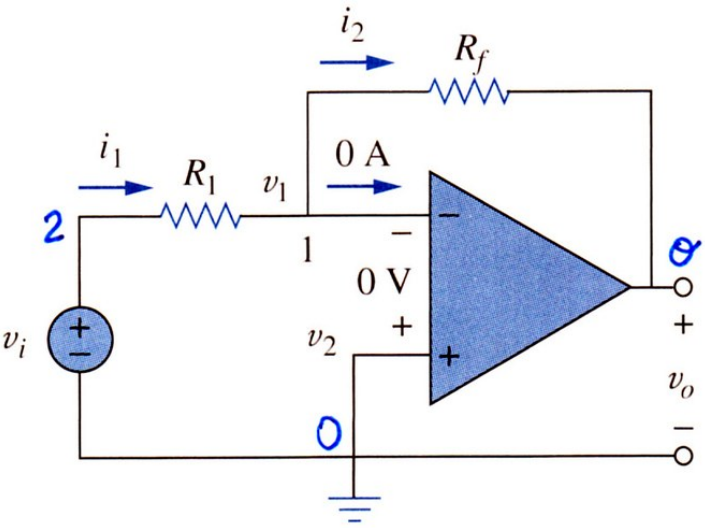


$$\frac{v_o}{v_s} = -\frac{R_2}{R_1}$$

We get **-r2/r1**, which is correct, as can be seen in the book's answer above.

AS2's Figure 5.10 (Inverting)

Find v_o.



$$v_o = -\frac{R_f}{R_1} v_i$$

```
type
s\dc("e,2,0,vi:r1,2,1,r1:rf,1,o,rf:o,0,1,o"):vo
```

This problem is almost identical to the one above. The answer we get is correct:

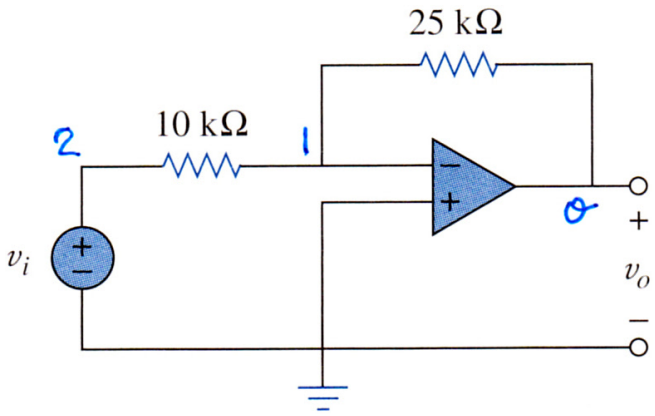
-(rf/r1)vi

AS2's Example 5.3 (Inverting)

If v_i is 0.5V, calculate the output voltage v_o and the current in the 10 kΩ resistor.

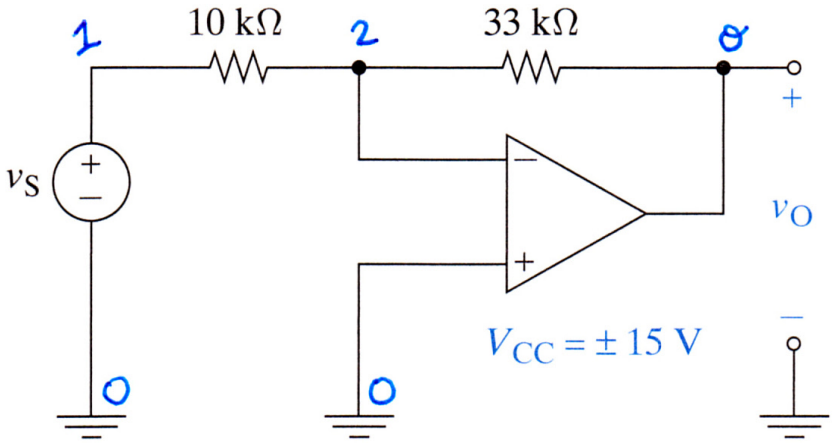
```
type
s\dc("e,2,0,.5:r1,2,1,10'k:rf,1,o,25'k:o,0,1,o"):{vo,ir1}
```

The answer, **{-1.25, 5e-5}**, is correct.



TR5's Exercise 4-11 (Inverting)

Find v_O when v_S is 2V, -4V and 6V. Notice the output of the op amp is limited to ±15V.



This may be the only non-linear problem you will see in this book, because I solved it before I realized it included the ±15V constraint. But anyway, here it goes.

```
type
s\dc("e,1,0,vs:r1,1,2,10'k:r2,2,o,33'k:o,0,2,o"):
{vo|vs=2.,vo|vs=-4.,vo|vs=6.}
```

The answer, **{-6.6,13.2,-19.8}**, is correct within the linear realm, but since the output is constrained to no more than 15V or less than -15V, the answer is v_O=**-15**V for v_S = **6**V.

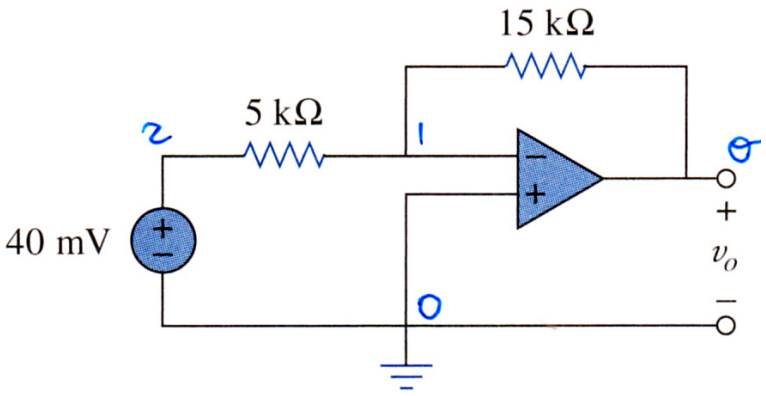
AS2's Practice Problem 5.3 (Inverting)

Find the output voltage of the op amp (i.e. v_o) and calculate the current through the feedback resistor (i.e. the 15 kΩ resistor).

My answer: Notice we used the m for milli in the value of the voltage source.

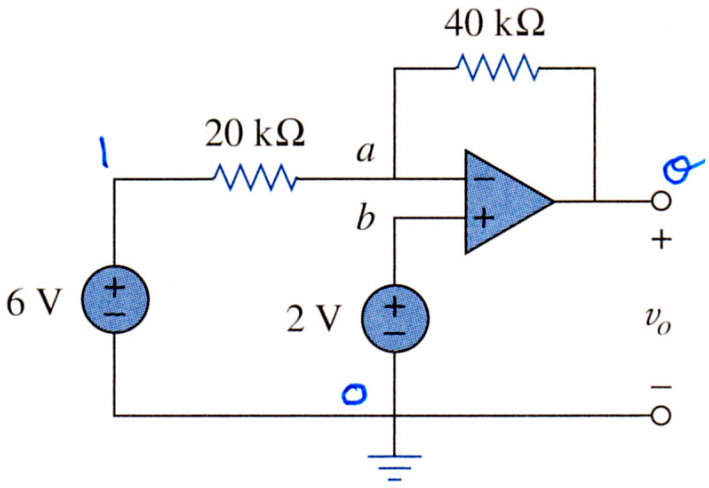
```
type
s\dc("e,2,0,40'm:r1,2,1,5'k:rf,1,o,15'k:o,0,1,o"):approx({vo,irf})
```

The answer, **{-.12, 8e-6}**, is correct.



AS2's Example 5.4 (Inverting)

Determine v_o.

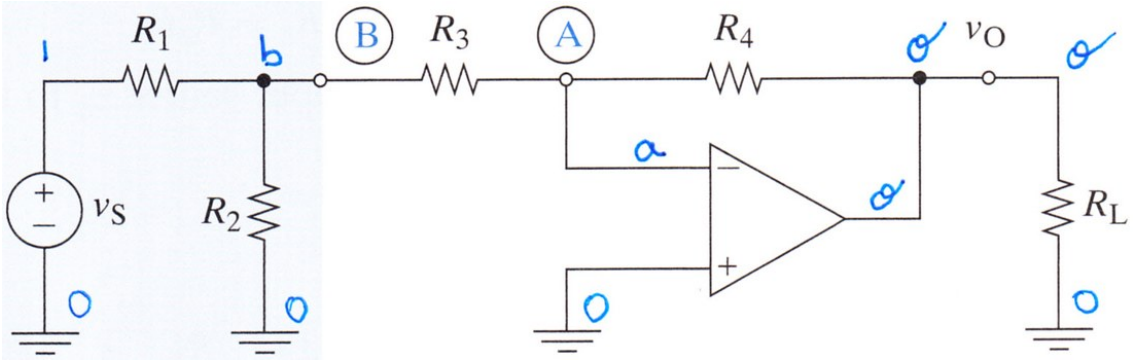



```
type
s\dc("e6,1,0,6:r1,1,a,20'k:e2,b,0,2:rf,a,o,40'k:o,a,b,o"):vo
```

The answer, **-6**, is correct.

TR5's Example 4-14 (Inverting)

Find the input-output relationship of the circuit.



The 'formal' way to *symbulate* this circuit would be as described below.

```
type
s\dc("e,1,0,vs:r1,1,b,r1:r2,b,0,r2:r3,b,a,r3:r4,a,o,r4:o,0,a,o:rl,o,0,
rl")
```

However, since we are only interested in the ratio of the input to the output, a smarter (and faster – as in 49 seconds instead of 55 seconds) way to *symbulate* it is this:

```
type
s\dc("e,1,0,1:r1,1,b,r1:r2,b,0,r2:r3,b,a,r3:r4,a,o,r4:o,0,a,o:rl,o,0,1")
```

When we evaluate `vo/v1`, both approaches get the same answer:

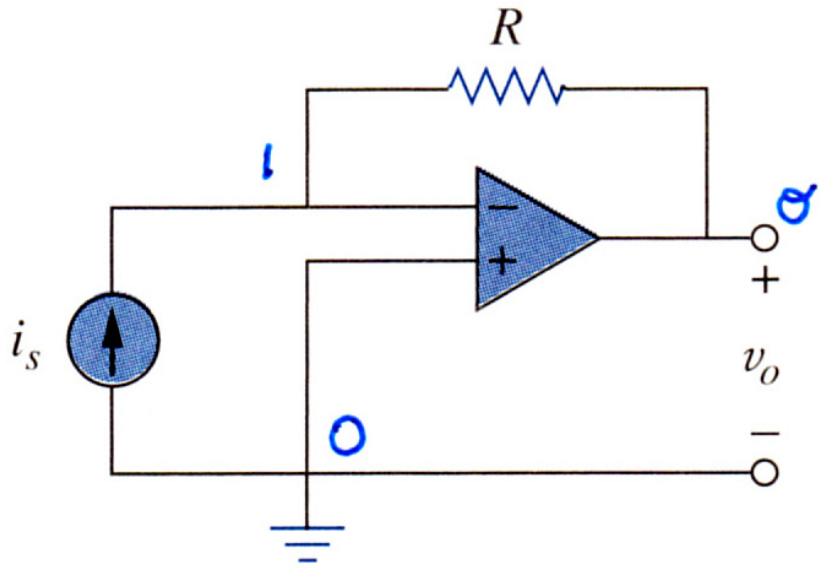
$$\frac{\text{- r2 r4}}{\text{r1 (r2 + r3) + r2 r3}}$$

which is correct, as can be seen by comparing it to the book's answer.

$$-\frac{R_2R_4}{R_1R_2 + R_1R_3 + R_2R_3}$$

AS2's Practice Problem 5.4a (Transresistance)

This is a current-to-voltage converter, also called a *transresistance amplifier*. Find $\boldsymbol{v_o/i_S}$.

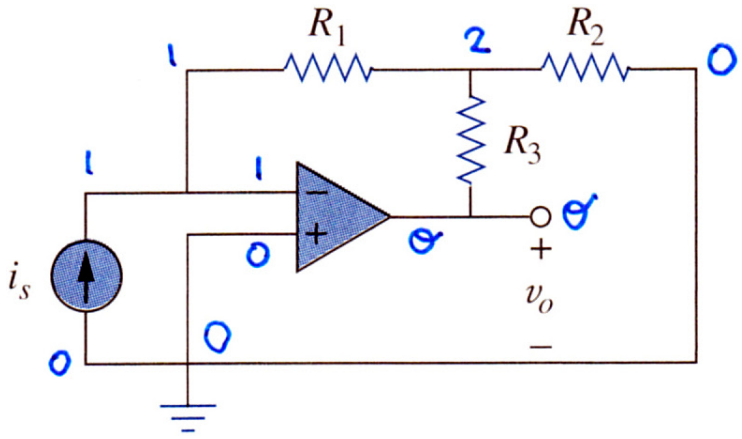



```
type
s\dc("j,0,1,is:r,1,o,r:o,0,1,o"):vo/is
```

The answer, **-r**, is correct.

AS2's Practice Problem 5.4b (Transresistance)

This is another *transresistance amplifier*. Again, find v_o/i_S.




```
type
s\dc("j,0,1,is:r1,1,2,r1:r2,2,0,r2:r3,2,o,r3:o,0,1,o")
```

Asking:

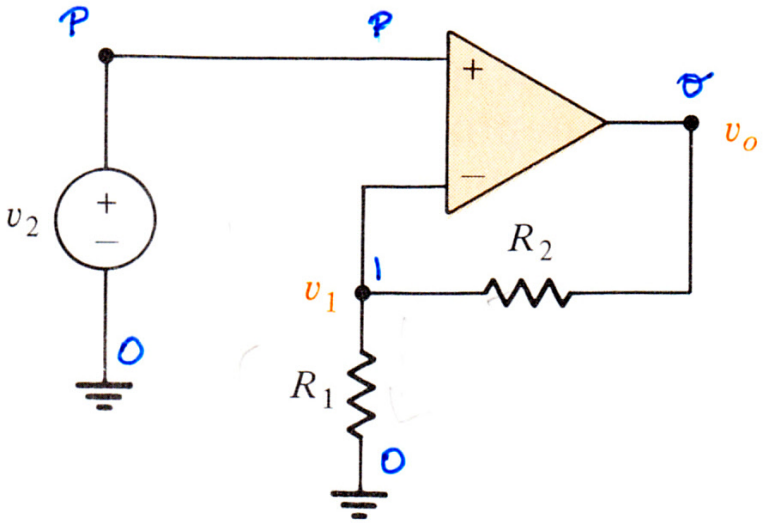
`expand(vo/is)`

gets **-r1*r3/r2-r1-r3**, which is equivalent to the book's answer.

$$\frac{v_o}{i_s} = -R_1\left(1 + \frac{R_3}{R_1} + \frac{R_3}{R_2}\right)$$

Bo2's Example 3.1 (Non-Inverting Amplifier)

Find v_o/v_1.

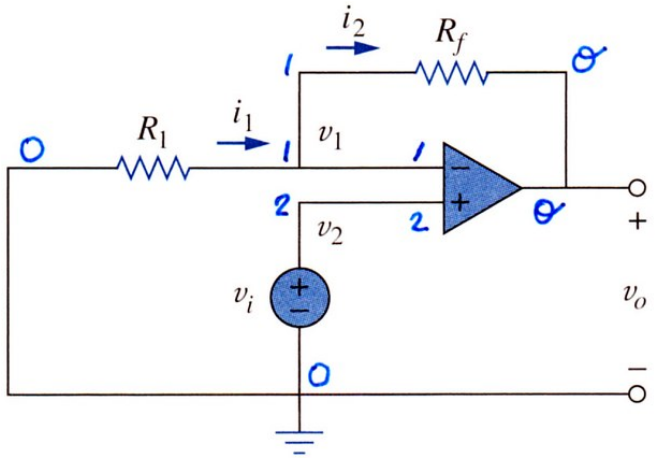


type
```
s\dc("e,p,0,v2:r1,1,0,r1:r2,1,o,r2:o,p,1,o"):vo/v1
```

We get $1 + \frac{r_2}{r_1}$, which is correct.

AS2's Figure 5.16 (Non-Inverting Amplifier)

Find v_o.



$$v_o = \left(1 + \frac{R_f}{R_1}\right) v_i$$

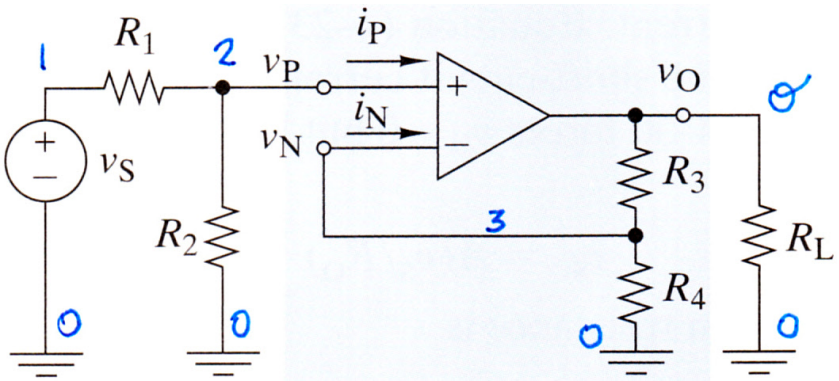
type
```
s\dc("e,2,0,vi:r1,0,1,r1:rf,1,o,rf:o,2,1,o"):vo
```

We get the right answer (below), an expression equivalent to the book's answer.

$$\left(\frac{r1 + rf}{r1}\right) vi$$

TR5's Example 4-13 (Non-Inverting Amplifier)

Find V_o/V_s.



We can solve this problem in one simulation, as shown below.

type
```
s\dc("e,1,0,vs:r1,1,2,r1:r2,2,0,r2:o,2,3,o:r3,o,3,r3:r4,3,0,r4"):vo/vs
```

We get

$$\frac{r2\,(r3+r4)}{(r1+r2)\,r4}$$

which is correct, as can be seen by comparing it to the book's answer.

$$\left[\frac{R_2}{R_1+R_2}\right]\left[\frac{R_3+R_4}{R_4}\right]$$

We can also solve it in stages, as shown below. First, simulate the left half.

type
```
s\dc("e,1,0,vs:r1,1,2,r1:r2,2,0,r2"):v2/vs
```

We get

$$\frac{r2}{r1+r2}$$

which is correct for this part. Then, simulate the right half.

type
```
s\dc("e,2,0,1:o,2,3,o:r3,o,3,r3:r4,3,0,r4"):vo/v2
```

We get

$$\frac{r3+r4}{r4}$$

which is correct for this part. The product of these two partial answers produces the same expression shown above after the big simulation, and is the right answer.

AS2's Figure 5.17 (Voltage Follower)

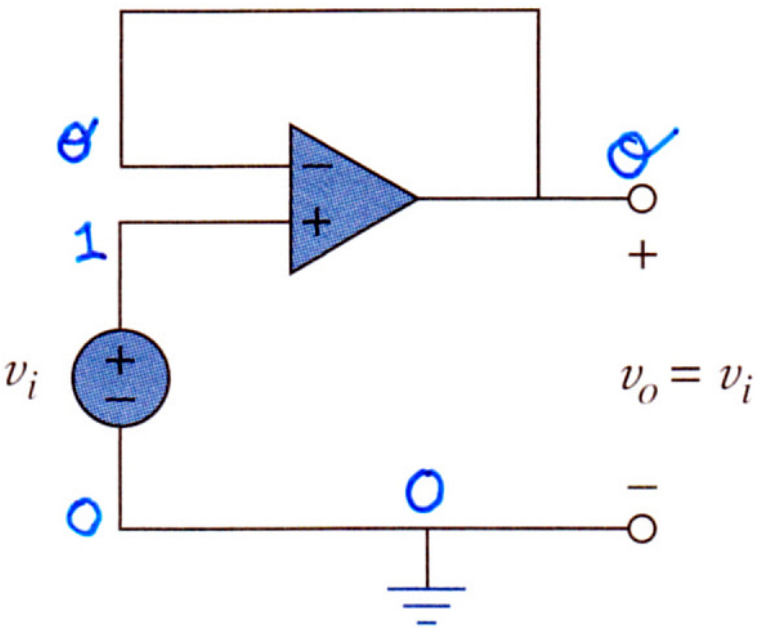
Find v_o.

type
```
s\dc("e,1,0,vi:o,1,o,o"):vo
```

The answer, **vi**, is correct.

TR5's Figure 4-32 (Voltage Follower)

Alright, so first we simulate the (b) circuit, to find its maximum power and load power:



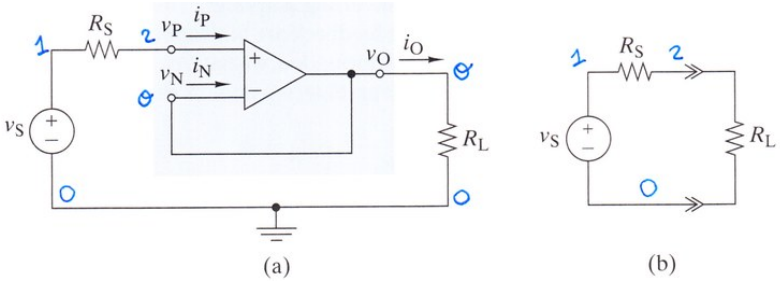


FIGURE 4–32 *(a) Source-load interface with a voltage follower. (b) Interface without the voltage follower.*

The circuits in Figure 4–32 have $v_S = 1.5$ V, $R_S = 2$ kΩ, and $R_L = 1$ kΩ. Compute the maximum power available from the source. Compute the power absorbed by the load resistor in the direct connection in Figure 4–32(b) and in the voltage follower circuit in Figure 4–32(a). Discuss any differences.

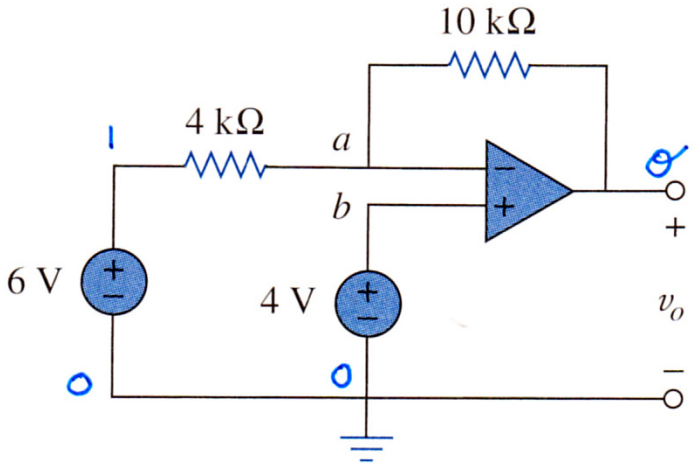
Answers: 281 μW; 250 μW; 2250 μW

```
type
s\th("e,1,0,1.5:rs,1,2,2'k",2,0):{pmax,prL|L=1000}
```

The answers we get, **{2.8125e-4,2.5e-4**}, are correct. Now we simulate the (a) circuit.

```
type
s\dc("e,1,0,1.5:rs,1,2,2'k:o,2,o,o:rl,o,0,1'k"):prL
```

The answer, **.00225**, is correct. The apparent paradox — the load in (a) drawing more power than the source in (b) seems able to provide — evaporates once we remember that the ideal op amp in the schematic is only part of the truth: the real one has its own source of power, which makes up the difference.



AS2's Example 5.5 (Inverting)

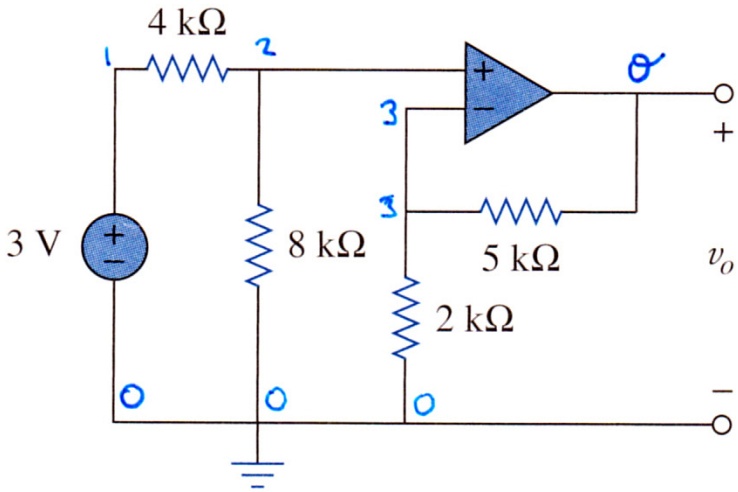
Find v_o.

```
type
s\dc("e,1,0,6:r4,1,a,4'k:r10,a,o,10'k:e4,b,0,4:o,b,a,o"):vo
```

The answer, **-1**, is correct.

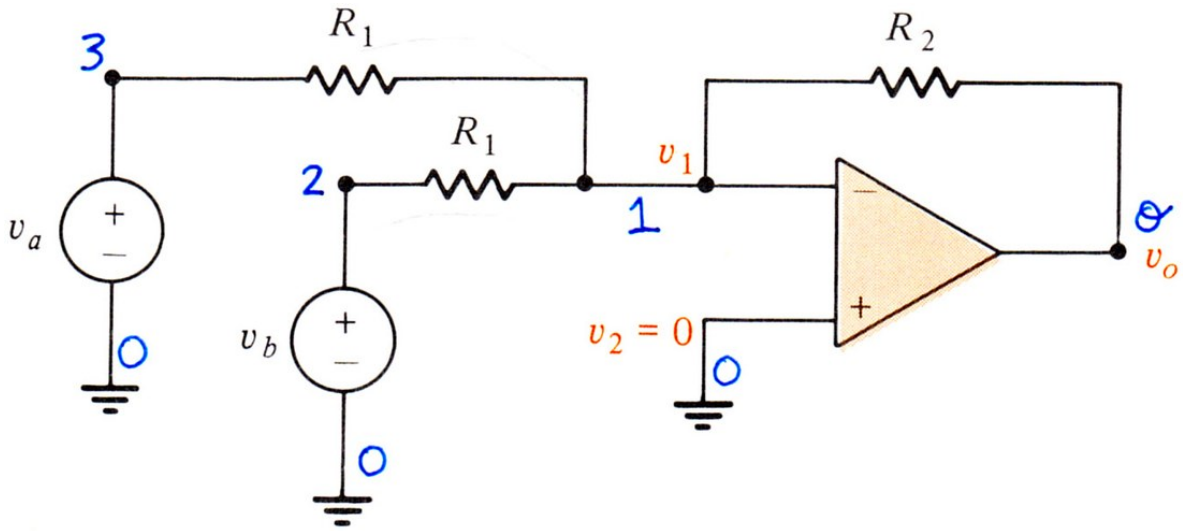
AS2's Practice Problem 5.5 (Non-Inverting)

Calculate v_o.




```
type
s\dc("e,1,0,3:r4,1,2,4'k:r8,2,0,8'k:r2,3,0,2'k:r5,3,o,5'k:o,2,3,o")
```

The answer for `vo`, **7**, is correct.



Bo2's Example 3.2 (Adder or Summing)

Find v_o.

```
type
s\dc("ea,3,0,va:eb,2,0,vb:r31,3,1,r1:r21,2,1,r1:r1o,1,o,r2:o,0,1,o"):vo
```

-(r2/r1)(va+vb)

which is correct, as can be seen by comparing it to the book's answer.

$$v_o = -\frac{R_2}{R_1}(v_a + v_b)$$

AS2's Figure 5.21 (Adder or Summing)

Find v_o.

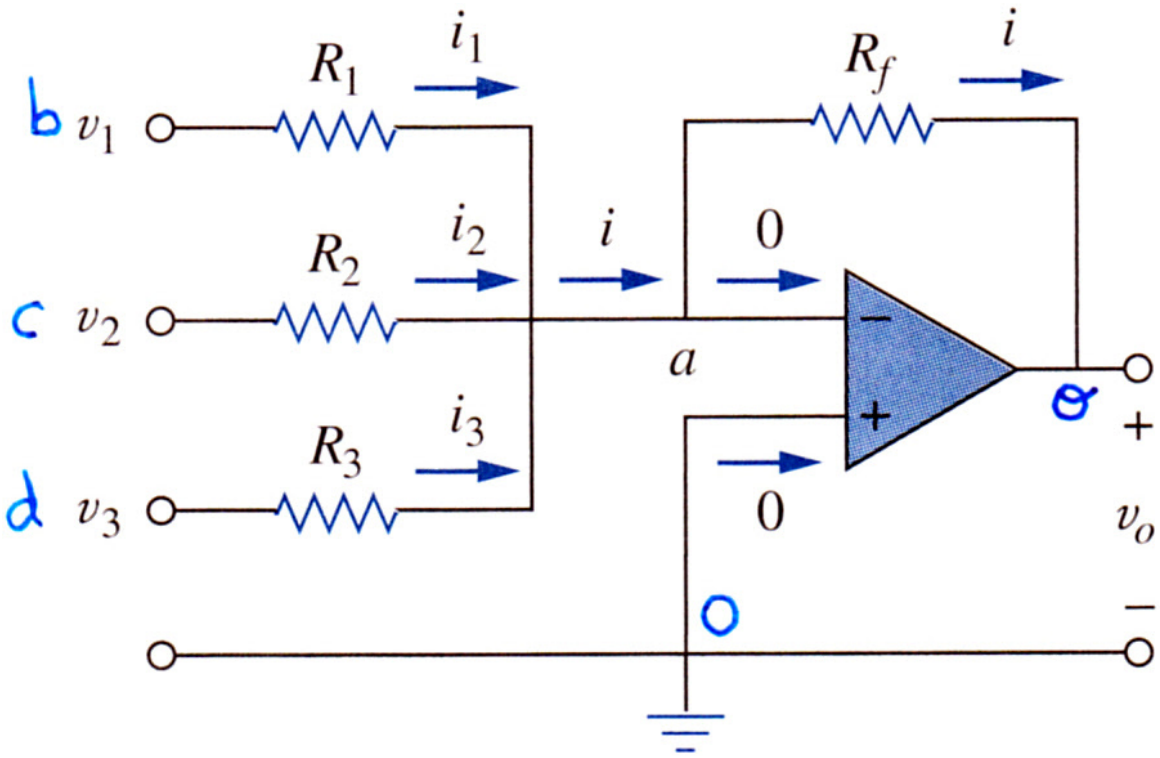
```
type
s\dc("e1,b,0,v1:e2,c,0,v2:e3,d,0,v3:r1,b,a,r1:
r2,c,a,r2:r3,d,a,r3:rf,a,o,rf:o,0,a,o"):expand(vo)
```

-rf v1/r1 - rf v2/r2 - rf v3/r3

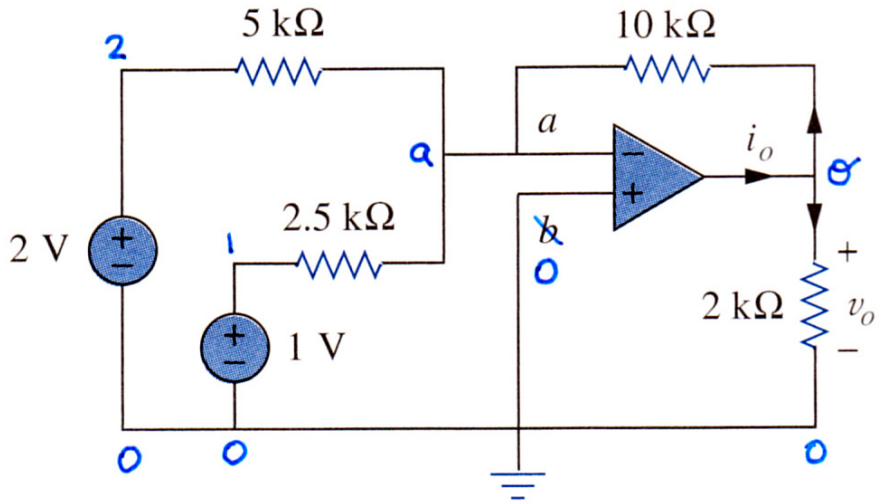
which is correct, as can be seen by comparing it to the book's answer.

AS2's Example 5.6 (Adder or Summing)

Find v_o and i_o.



$$v_o = -\left(\frac{R_f}{R_1}v_1 + \frac{R_f}{R_2}v_2 + \frac{R_f}{R_3}v_3\right)$$

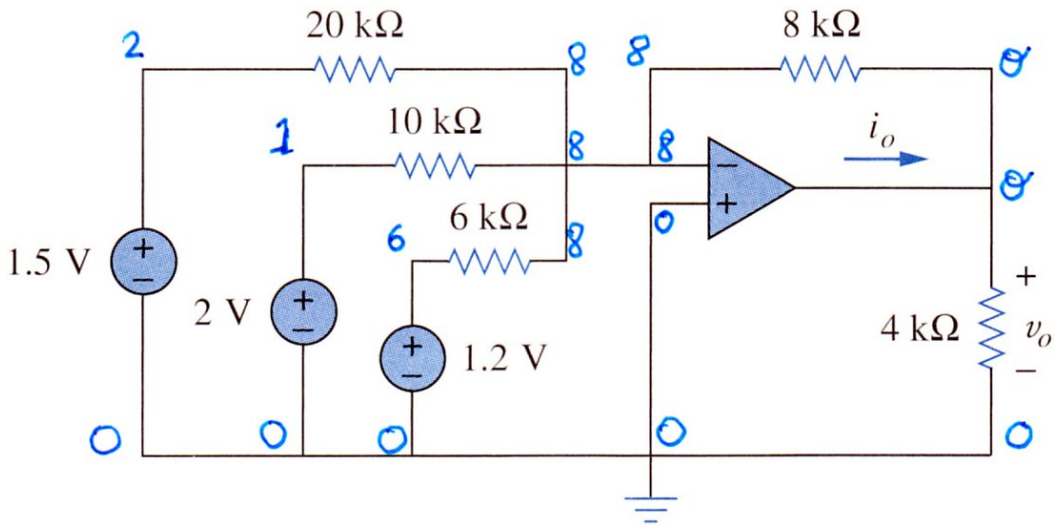



```
type
s\dc("e1,1,0,1:e2,2,0,2:r1,2,a,5'k:
r2,1,a,2.5'k:r3,a,o,10'k:r4,o,0,2'k:o,0,a,o"):{vo,io}
```

The answer, **{-8.,-.0048}**, is correct. Notice a current of 4.8mA is going into the op amp.

AS2's Practice Problem 5.6 (Adder or Summing)

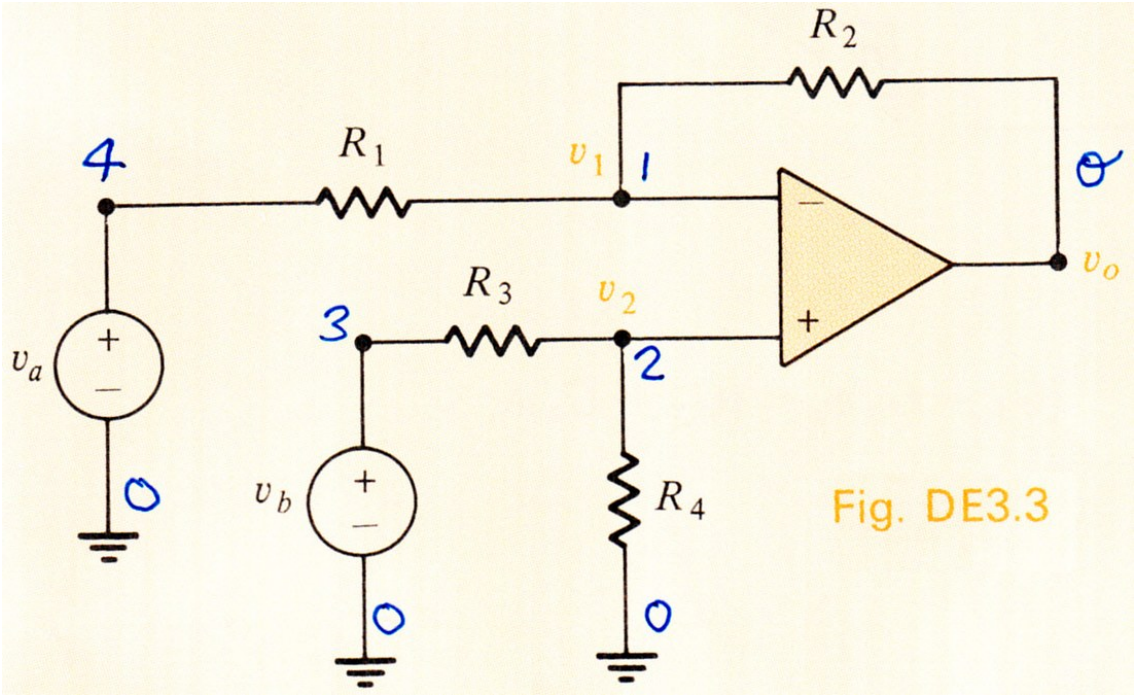
Find v_o and i_o.



type
```
s\dc("e2,2,0,1.5:e1,1,0,2:e6,6,0,1.2:r2,2,8,20'k:
r1,1,8,10'k:r6,6,8,6'k:r8,8,o,8'k:r4,o,0,4'k:o,0,8,o"):{vo,io}
```

The answer, **{-3.8,-.001425}**, is correct. Again, the current is going into the op amp.

Bo2's Drill Exercise 3.3 (Difference or Differential)



Find v_o for the **difference-amplifier** circuit shown in Fig. DE3.3 when $R_3 = R_1$ and $R_4 = R_2$.

Answer: $\frac{R_2}{R_1}(v_b - v_a)$

type

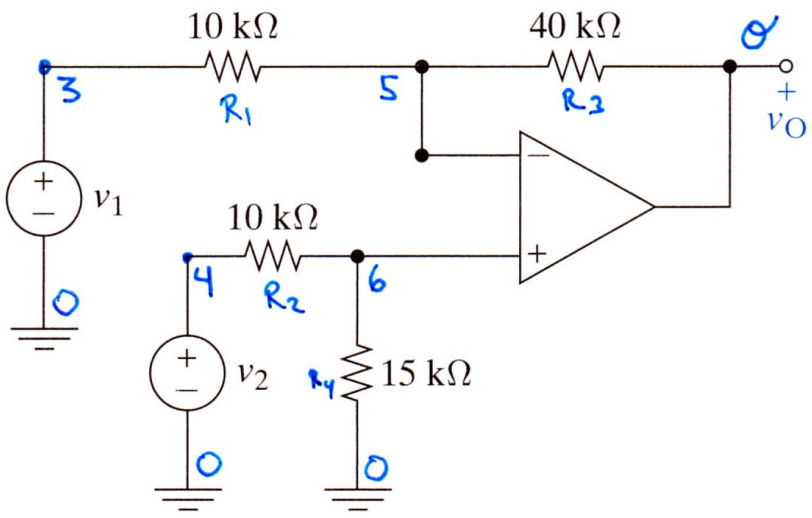
```
s\dc("ea,4,0,va:eb,3,0,vb:r1,4,1,r1:r2,1,o,r2:
r3,3,2,r1:r4,2,0,r2:o,2,1,o")
```

Evaluating vo we get the correct answer, equivalent to the book's answer above.

((vb-va) r2)/r1

TR5's Exercise 4-13 (Difference or Differential)

Find v_o.



type

```
s\dc("e1,3,0,v1:e2,4,0,v2:r1,3,5,10'k:r2,4,6,10'k:
r3,5,o,40'k:r4,6,0,15'k:o,6,5,o")
```

Evaluating vo we get **3 v2 – 4 v1**, which is the correct answer.

AS2's Figure 5.24 (Difference or Differential)

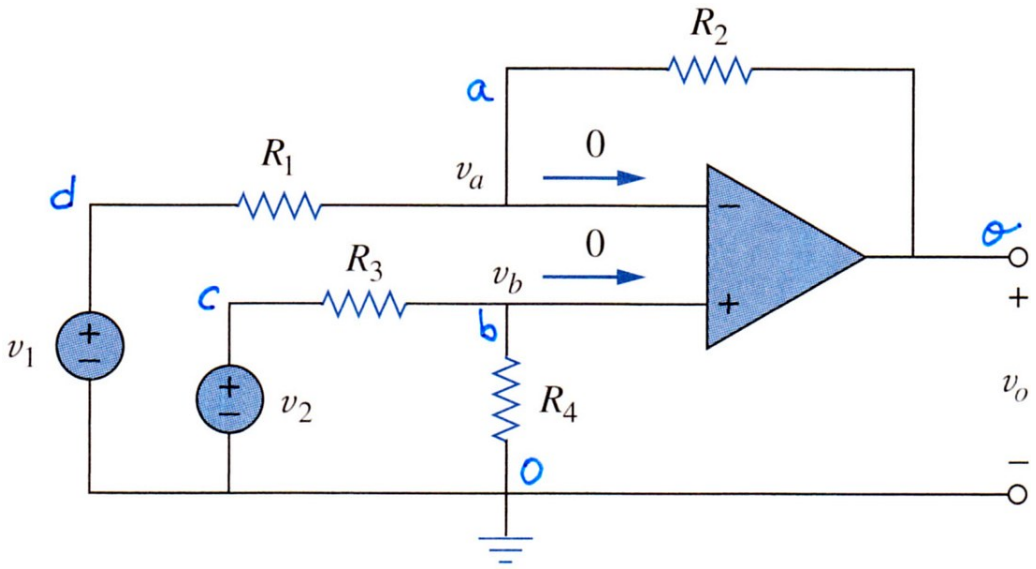
Find v_o. (And keep it in the memory, for you will use it in the next three problems.)

type

```
s\dc("e1,d,0,v1:e2,c,0,v2:r1,d,a,r1:r3,c,b,r3:
r2,a,o,r2:r4,b,0,r4:o,b,a,o"):vo
```

$$\frac{r1\,r4\,v2 - r2\left(r3\,v1 + r4\,(v1 - v2)\right)}{r1\,(r3 + r4)}$$

This expression is equivalent to the book's answer.



$$v_o = \frac{R_2}{R_1}\frac{(1 + R_1/R_2)}{(1 + R_3/R_4)}v_2 - \frac{R_2}{R_1}v_1$$

AS2's Figure 5.24 (Subtractor)

For the same circuit of the previous problem, find v_o when R_1=R_2 and R_3=R_4.

Since we already have the expression for vo stored in the memory, we only do this:

```
type
expand(vo)|r2=r1 and r3=r4
```

The answer we get, **v2-v1**, is correct.

AS2's Example 5.7 (Difference or Differential)

Design an op amp circuit with inputs v_1 and v_2 such that v_o = -5v_1 + 3v_2.

My solution follows. The problem statement is a fancy way of saying: for the circuit of the previous problem, find the resistor values that give an output v_o = -5v_1 + 3v_2. Not strictly a Symbulator problem, but it shows how Symbulator fits into design problems.

First we take that part of v_o that is a factor of v_1, and make it equal to -5. Thus:

```
type
Define v1=1:Define v2=0:expand(vo)=-5
```

We get **-r2/r1=-5**. Now make that part of v_o that is a factor of v_2 equal to 3. Thus:

```
type
Define v1=0:Define v2=1:expand(vo)=3
```

We get **r2*r4/(r1*(r3+r4))+r4/(r3+r4)=3** Now, since you have two equations, you can

solve for two unknowns. Of the four resistors you get to choose, two can be whatever you want. The book recommends R_1 = 10 kΩ and R_3 = 20 kΩ. Now let's find R_2 and R_4.

```
type
solve(ans(1) and ans(2),{r2,r4})|r1=10000 and r3=20000
```

The expression above assumes that ans(1) and ans(2) are pointing to the two equations we found before. We get **r2**=**50000** and **r4**=**20000**. This is correct.

Now, if this problem was part of a test, I'd like to verify that the answer is correct. To confirm this, simulate the circuit using the four values given above for the resistors.

```
type
s\dc("e1,d,0,v1:e2,c,0,v2:r1,d,a,10'k:
r3,c,b,20'k:r2,a,o,50'k:r4,b,0,20'k:o,b,a,o")
```

Evaluating vo gives us the desired output, **3*v2-5*v1**. The resistor values are correct.

AS2's Practice Problem 5.7 (Difference or Differential)

Design a difference amplifier with gain 4.

My solution follows. Again the statement is a fancy way of saying: for the circuit of the previous problem, find the resistor values that give an output v_o = **4** (v_2-v_1), that is, -4v_1 + 4v_2. Same as before.

```
type
Define v1=1:Define v2=0:vo=-4
```

```
type
Define v1=0:Define v2=1:expand(vo)=4
```

This time the book asks that you use R_1 = 10 kΩ and R_3 = 10 kΩ. So we do that.

```
type
solve(ans(1) and ans(2),{r2,r4})|r1=10000 and r3=10000
```

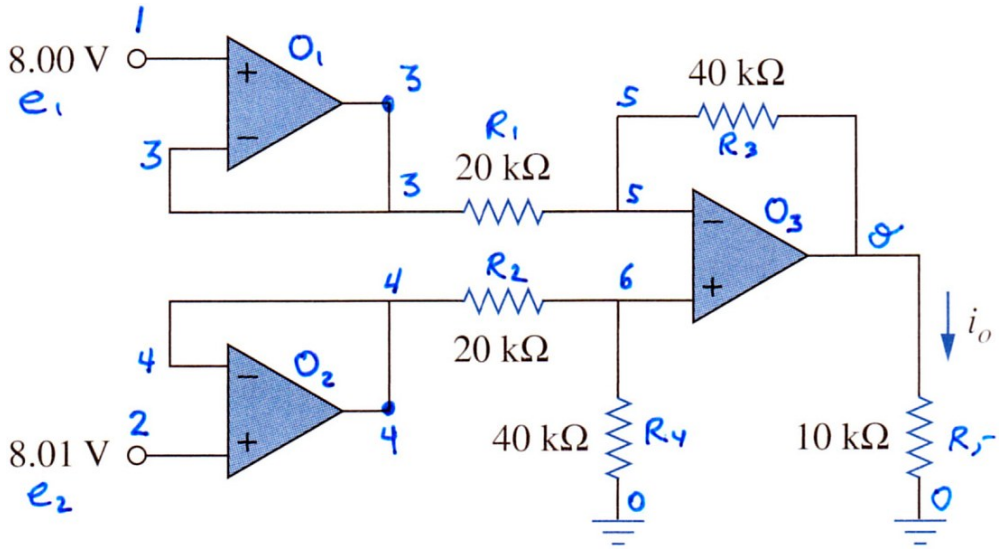
We get **r2**=**40000** and **r4**=**40000**, the correct values for the remaining resistors.

AS2's Practice Problem 5.8 (Instrumentation)

Find i_o.

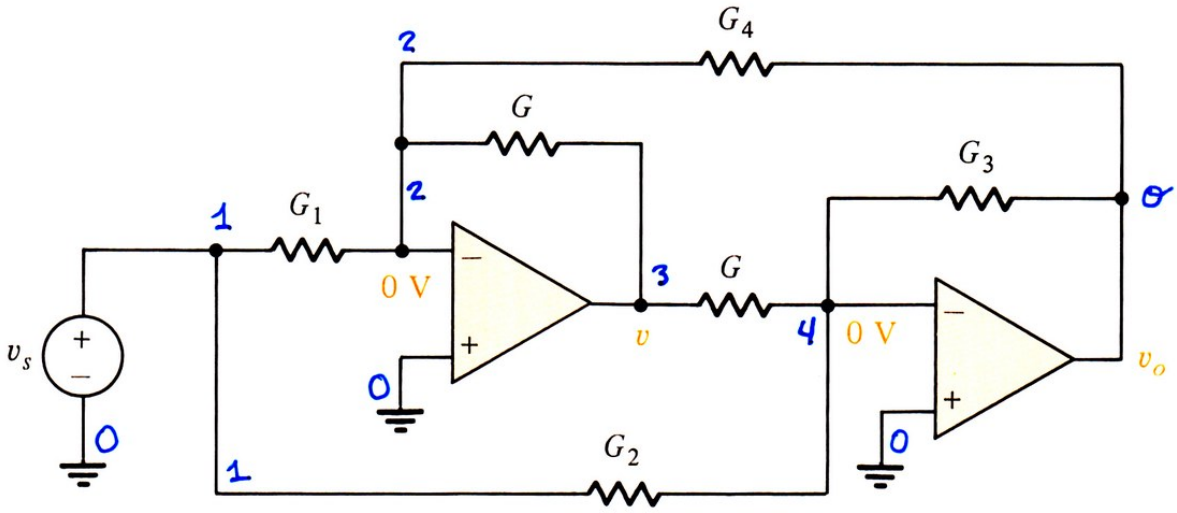
```
type
s\dc("e1,1,0,8.:e2,2,0,8.01:o1,1,3,3:o2,2,4,4:r1,3,5,20'k:
r2,4,6,20'k:r3,5,o,40'k:r4,6,0,40'k:o3,6,5,o:r5,o,0,10'k"):ir5
```

In the schematic, the current i_o corresponds to ir5. The answer, **2e-6**, is correct.



Bo2's Example 3.3 (Cascade)

Find v_o in terms of the conductances and the applied voltage v_S.




```
type
s\dc("e,1,0,vs:r12,1,2,1/g1:r14,1,4,1/g2:r4o,4,o,1/g3:
r2o,2,o,1/g4:r23,2,3,1/g:r34,3,4,1/g:o1,0,2,3:o2,0,4,o")
```

Evaluating vo we get:

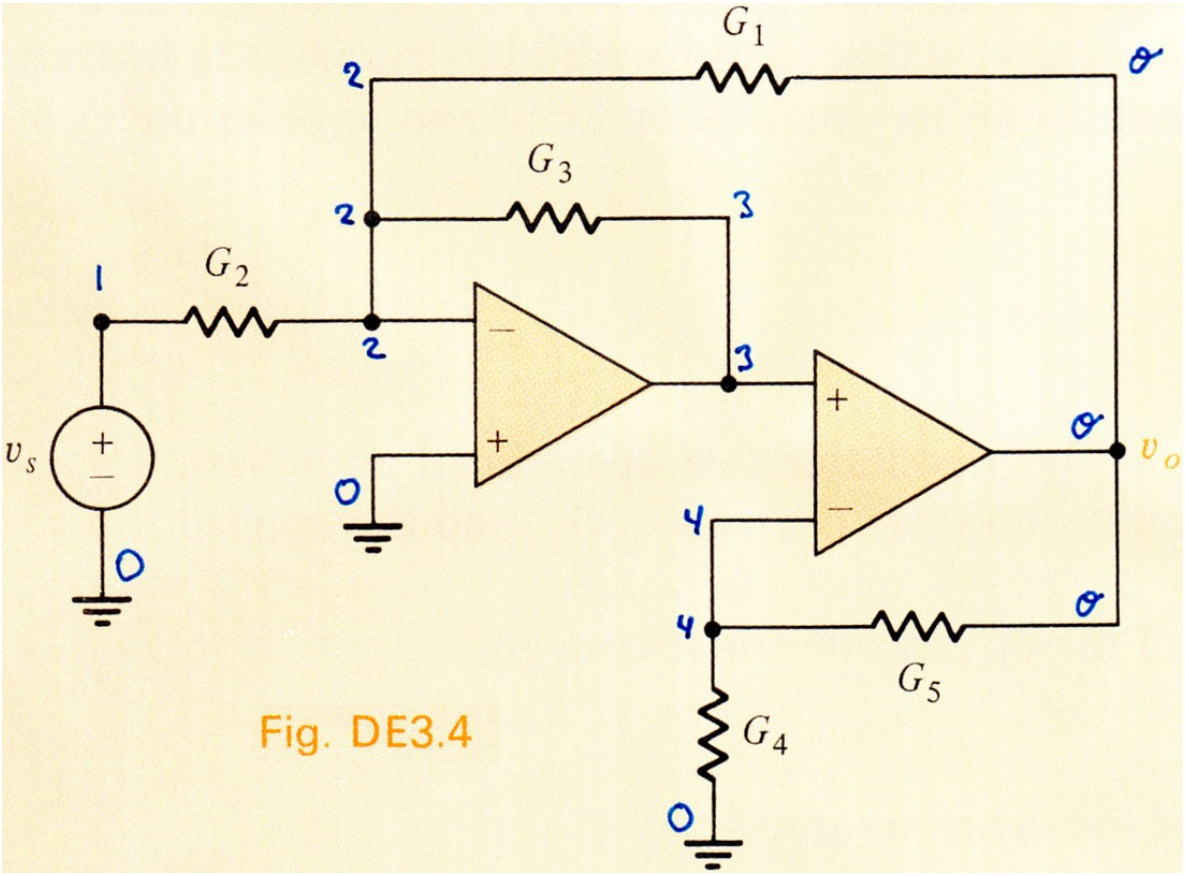
((g1-g2) vs)/(g3-g4)

which is correct, as can be seen by comparing it to the book's answer.

$$v_o = \frac{G_1 - G_2}{G_3 - G_4} v_s$$

Bo2's Drill Exercise 3.4 (Cascade)

For the op-amp circuit shown in Fig. DE3.4, suppose that $G_1 = 1$ S, $G_2 = 2$ S, $G_3 = 3$ S, $G_4 = 4$ S, and $G_5 = 5$ S. Find v_o in terms of v_s. Answer: $-0.75v_s$




```
type
s\dc("e,1,0,vs:r1,2,o,1/1:r2,1,2,1/2:r3,2,3,1/3:
r4,4,0,1/4:r5,4,o,1/5:o1,0,2,3:o2,3,4,o")
```

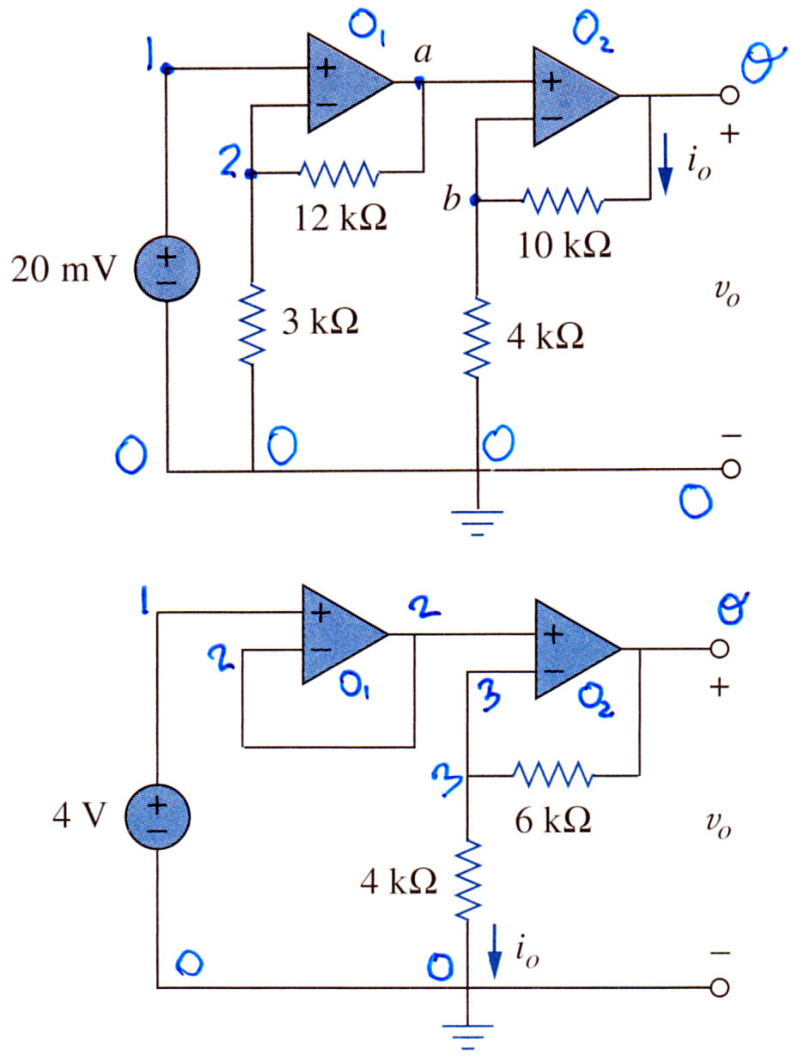
Evaluating `vo` we get: **-.75 vs**, which is correct.

AS2's Example 5.9 (Cascade)

Find v_o and i_o.

```
type
s\dc("e,1,0,20'm:o1,1,2,a:o2,a,b,o:ro,o,b,10'k:
r4,b,0,4'k:r2,a,2,12'k:r3,2,0,3'k")
```

Evaluating `approx({vo,iro})` we get the answer, **{.35,2.5e-5}**. This is correct.



AS2's Practice Problem 5.9 (Cascade)

Determine v_o and i_o.

```
type
s\dc("e,1,0,4:o1,1,2,2:o2,2,3,o:ro,3,0,4'k:r6,3,o,6'k"):{vo,iro}
```

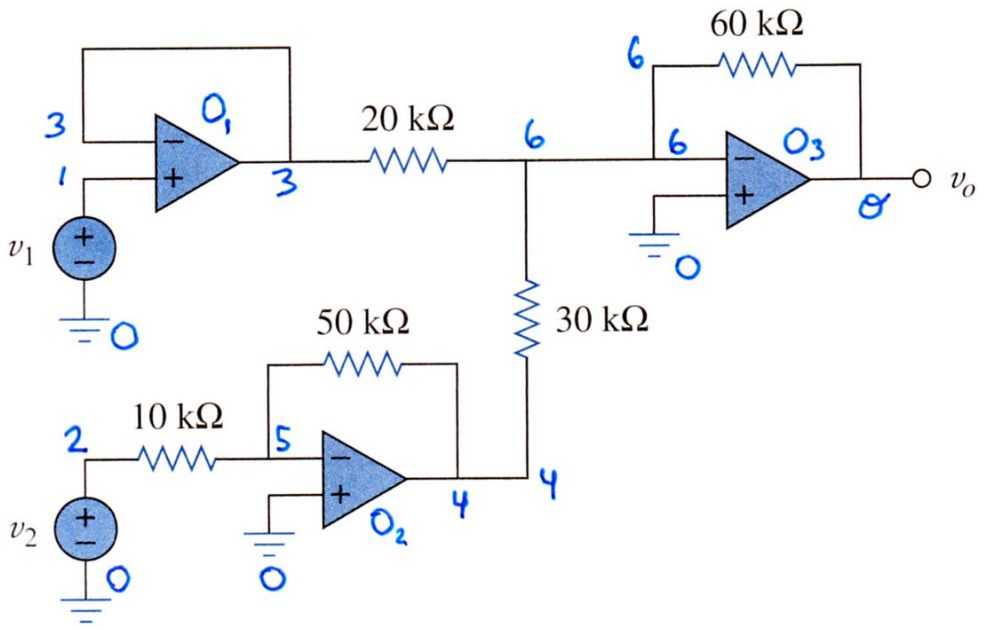
The answer, **{10,1/1000}**, is correct. To say a 1/1000 A current is the same as 1mA.

AS2's Practice Problem 5.10 (Cascade)

If v_1 = 2V and v_2 = 1.5V, find v_o in the circuit.

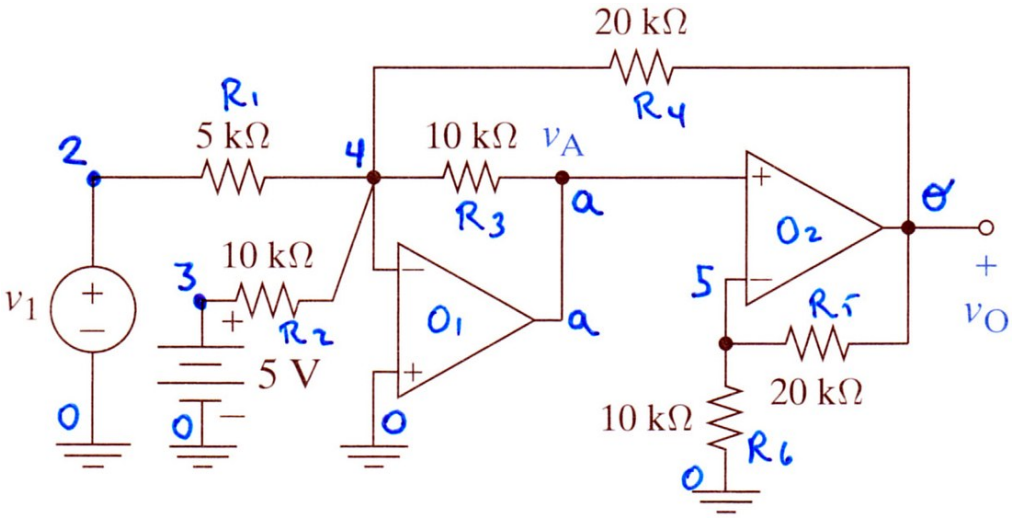
```
type
s\dc("e1,1,0,2:e2,2,0,1.5:o1,1,3,3:r1,2,5,10'k:r2,3,6,20'k:
r5,5,4,50'k:o2,0,5,4:r3,6,4,30'k:r6,6,o,60'k:o3,0,6,o"):vo
```

The answer, **9**, is correct.



TR5's Example 4-16 (Cascade)

Derive an expression for v_o in terms of the two inputs.

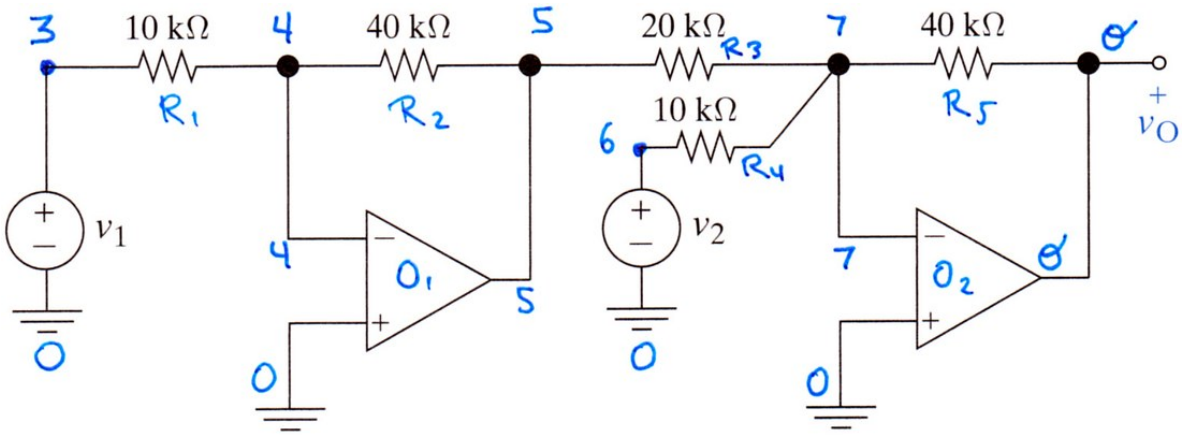



```
type
s\dc("e1,2,0,v1:e5,3,0,5:r1,2,4,5'k:r2,3,4,10'k:o1,0,4,a:r3,4,a,10'k:
r4,4,o,20'k:o2,a,5,o:r5,5,o,20'k:r6,5,0,10'k"):expand(approx(vo))
```

The answer, **-2.4 v1 – 6**, is correct.

TR5's Exercise 4-14 (Cascade)

Derive an expression for v_o in terms of the inputs v_1 and v_2.



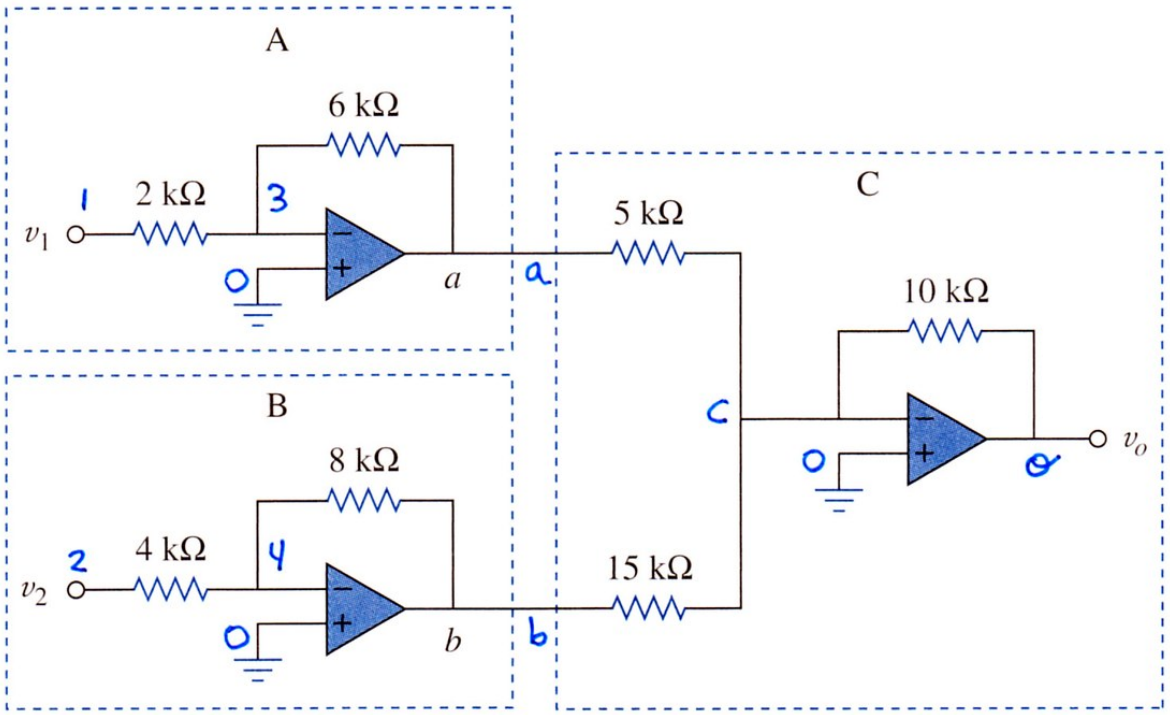
type

```
s\dc("e1,3,0,v1:r1,3,4,10'k:o1,0,4,5:r2,4,5,40'k:r3,5,7,20'k:
e2,6,0,v2:r4,6,7,10'k:r5,7,o,40'k:o2,0,7,o"):expand(vo)
```

The answer, **8 v1 – 4 v2**, is correct.

AS2's Example 5.10 (Cascade)

If v_1 = 1V and v_2 = 2V, find v_o in the circuit.



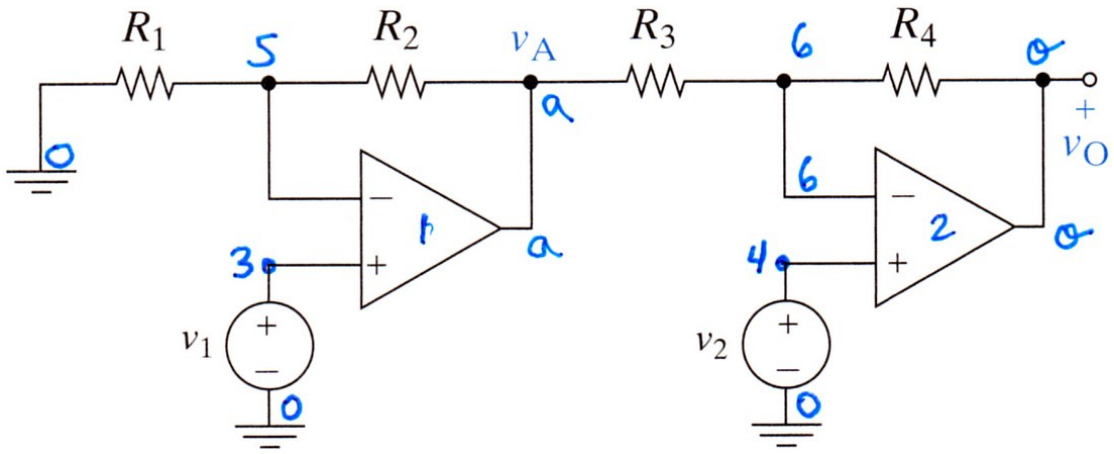
type

```
s\dc("e1,1,0,1:e2,2,0,2:r2,1,3,2'k:r4,2,4,4'k:r6,3,a,6'k:r8,4,b,8'k:r5,a,
c,5'k:r15,b,c,15'k:r10,c,o,10'k:oa,0,3,a:ob,0,4,b:oc,0,c,o"):approx(vo)
```

The answer, **8.667**, is correct.

TR5's Example 4-17 (Cascade)

Derive an expression for v_o in terms of the inputs v_1 and v_2.




```
type
s\dc("r1,5,0,r1:r2,5,a,r2:r3,a,6,r3:r4,6,o,r4:e1,3,0,v1:
e2,4,0,v2:o1,3,5,a:o2,4,6,o"):vo
```

We get the right answer. Let's compare it with the book's answer.

$$-\left[\frac{R_4}{R_3}\right]\left[\frac{R_1 + R_2}{R_1}\right]v_1 + \left[\frac{R_3 + R_4}{R_3}\right]v_2$$

Our answer, expanded via `expand(vo)`, is shown below:

$$-\frac{r2\,r4\,v1}{r1\,r3} - \frac{r4\,v1}{r3} + \frac{r4\,v2}{r3} + v2$$

Playing with it by hand we get a form that, in my opinion, is *prettier* ;-) than the book's:

$$v2\left(\frac{r4}{r3} + 1\right) - v1\left(\frac{r4}{r3}\right)\left(\frac{r2}{r1} + 1\right)$$

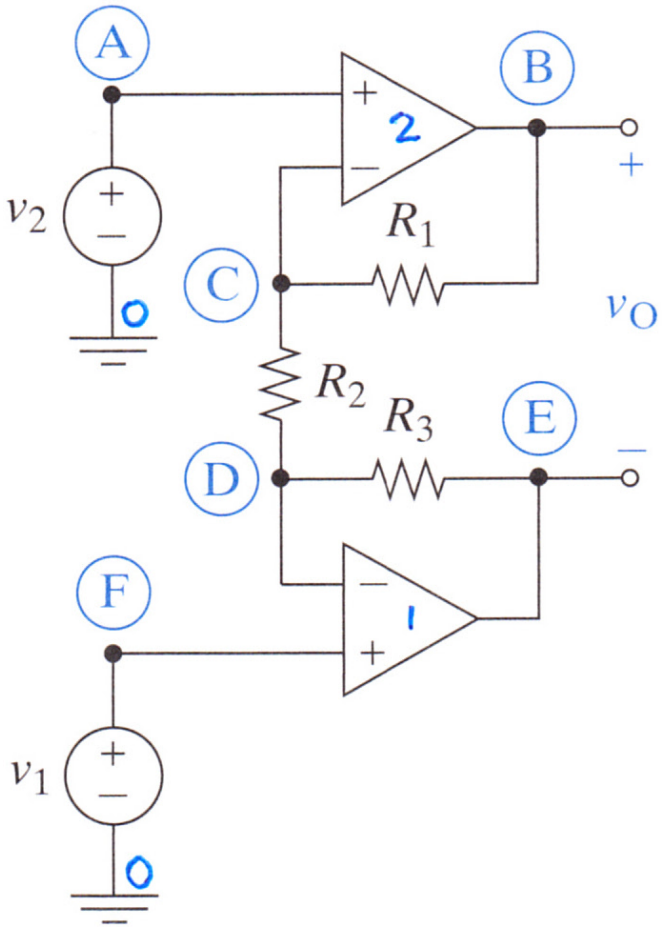
TR5's Example 4-18 (Multiple)

Derive an expression for v_o in terms of the inputs v_1 and v_2.

```
type
s\dc("e2,a,0,v2:e1,f,0,v1:o2,a,c,b:o1,f,d,e:r1,b,c,r1:r2,c,d,r2:r3,d,e,
r3")
```

To find vo, we ask for `vb-ve`. We get an expression that can easily be rearranged to look like this:

$$-\left(\frac{r1 + r2 + r3}{r2}\right)(v1 - v2)$$



This is exactly the answer from the book:

$$v_O = v_B - v_E = \left[\frac{R_1 + R_2 + R_3}{R_2}\right](v_2 - v_1)$$

If you are ever in doubt whether two expressions are the same, enter them both separately into the calculator and compare them with the equality sign. If the answer is '**true**', they are the same.

LESSON 6

Transient analysis

Learn to run a *transient* time-domain analysis with **tr**. Learn to describe *capacitors* with **c** and *inductors* with **l**. Save time with the **only** tool. Plot expressions in time with the **plot** tool.

Symbulator can simulate two energy-storing elements, capacitors and inductors, in three kinds of analysis: transient time-domain (TR), alternating current (AC), and complex frequency-domain (FD). This lesson teaches the transient analysis of a circuit, and how to describe capacitors and inductors for it. AC and FD come in later lessons.

6.1 How to describe a capacitor

> ⓘ NOTE
> **Describing capacitors**
> When describing an ideal capacitor for the purpose of a transient analysis, we need five pieces of information, separated by commas: a unique name to identify the capacitor (must start with the letter **c**), the names of its first and second nodes, its value in farads (F), and its initial condition in volts, understood as the voltage in the first node minus the voltage in the second.
> For example, an ideal capacitor called **ca**, connected between two nodes called **1** and **2**, with a capacitance of **3** F and an initial condition of 0.1 V, would be described as follows: `ca,1,2,3,0.1`

Values can use SI prefixes

The values of the capacitance and the initial condition of the capacitor can use prefixes of the International System.

What answers do you get

For each capacitor, a transient analysis will provide two answers: the voltage drop in the capacitor, in volts, and the current through it, flowing from the first node towards the second, in amperes.

> ⓘ NOTE
> **Note on capacitors in DC**

In a DC simulation, the current through a capacitor is defined as 0. It basically becomes an open circuit.

6.2 How to describe an inductor

NOTE

Describing inductors

When describing an ideal inductor for the purpose of a transient analysis, we need five pieces of information, separated by commas: a unique name to identify the inductor (must start with the letter **l**), the names of its first and second nodes, its value in henries (H), and its initial condition in amperes, understood as the current flowing through it from the first node towards the second.
For example, an ideal inductor called **l1**, connected between two nodes called **1** and **2**, with an inductance of **0.1** H and an initial condition of 0.2 A, would be described as follows: `l1,1,2,0.1,0.2`

For each inductor, a transient analysis will provide the voltage drop across it and the current through it, with the same conventions as for capacitors.

NOTE

Note on inductors in DC

In a DC simulation, the voltage drop in an inductor is defined as 0. It basically becomes a short circuit.

6.3 Running a transient analysis

You will learn the subtleties of transient simulations through a series of examples. For now, to run one we use an access program called **s\tr**, which takes one argument: the circuit description in string form.

For transient simulations, Symbulator uses the Laplace Functions by Lars Frederiksen. Make sure you have them properly installed!

Bo2's Example 4.15

SOLUTION

For $t \leq 0$ s, since the current source is inactive, there is nothing going on. Without doing a simulation, we know the voltage v is 0 volts.

For $0 \text{ s} < t \leq 2$ s, the source is active. From the graph, its value in this interval is t: the current in amperes equals the time in seconds. We describe the circuit and run the transient analysis:

EXAMPLE 4.15

For the circuit shown in Fig. 4.37(a), let us determine v given that i is described by Fig. 4.37(b). In this case, let us consider three intervals of time.

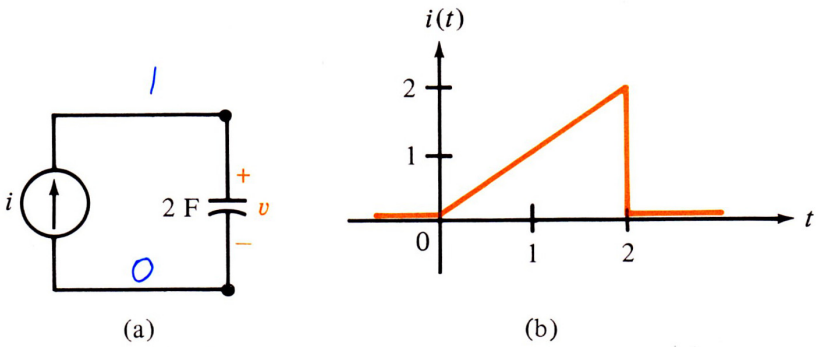


Fig. 4.37 (a) Circuit with a capacitor. (b) Applied current.

Bo2's Example 4.15

```
type
s\tr("j,0,1,t:c,1,0,2,0")
```

The source has four terms: name (starting with j), first node, second node, and value — here t. The capacitor has five: name, first node, second node, value in farads, and initial condition in volts — here 0 volts.

Once the simulation is complete, ask for the value of the voltage, `vc`. We get this value, which is correct:

```
returns
t^2/4
```

For 2 s < t, the voltage is equal to whatever value it had at time 2 s:

```
type
vc|t=2
```

We get a value of 1 volt, which is correct.

Describing source values

For the purpose of a transient analysis, Symbulator accepts many types of source values. The only thing you must do is describe these values properly.

Step values: symbolic. A source with an unknown step value starting at t=0 is described with a variable and the step function u(t). For example, a voltage source **e1** between nodes **1** and **0**, of V volts starting at t=0, is `e1,1,0,V*u(t)`.

Step values: numerical. A step source with a known numerical value can be described the same way, `e1,1,0,12*u(t)`. To save typing, you can skip the u(t) when the value is numerical, and Symbulator assumes a step: `e1,1,0,12`. The results are the same.

Impulse values. A source with an impulse value at t=0 is described with its value, sym-

bolic or numerical, and the delta function δ(t). For example, a current source **j1** between nodes **0** and **1**, with an impulse of i amperes at t=0, is `j1,0,1,i*δ(t)`.

Values as functions of time. A source whose value is a function of time, such as a ramp, a sinusoid or an exponential, is described by writing the value as an expression in terms of t, as in `j,0,1,t`. Values as functions of t will activate the Impala mode, in order to save time.

Dependent values. Sources with dependent values are described as we have seen before, for example `j,0,1,3*vr1`.

A word on intervals

In a transient analysis, every switch opening or closing ends one time interval and begins another, and a simulation covers one interval only. When we run a TR simulation and get answers in terms of t, this variable must be understood as the time elapsed since the start of *that* interval in particular.

This is a distinction without a difference as long as the interval we simulate starts at t=0, which is most often the case. However, sometimes problems have switch changes at other times — **Bo2's Example 5.6 (page 135)**, where a switch closes at t=1 s, and **Bo2's Drill Exercise 5.6 (page 136)**, where one closes at t=2 s, are examples in the solved problems.

6.4 Two useful tools

The only tool

Unless you say otherwise, a Symbulator simulation gives you the whole set of answers: voltages in all nodes, and voltage drops, currents and power consumed in all elements. When the expressions are complicated, storing them all takes time.

In case you don't want all the answers, but only a chosen few, you may save some time by using the **s\only** tool to tell Symbulator — before the simulation — which answers you want to save. Find it in Symbulator's custom menu. Its argument is either an empty variable or a string with a variable or a list of variables separated by commas. All of these are valid:

```
type
s\only(ir1)
s\only("ir1")
s\only("ir1,va,vr2")
```

The s\only tool is valuable in TR analysis: you save time by not having Symbulator find the inverse Laplace of answers you do not need.

The plot tool

Very often we are asked to plot functions of time. The calculator has extensive plotting capabilities, but just to save you some time, Symbulator includes a handy **s\plot** tool

that allows you to plot functions of time.

```
type
s\plot()
```

6.5 Instructive solved examples

Transient analysis of RC circuit

Bo2's Example 5.1

EXAMPLE 5.1

Let us begin this example by considering the circuit in Fig. 5.4, in which there is a switch that is closed for time $t < 0$, is opened at time $t = 0$, and stays open for all time $t > 0$. For this circuit, let us determine $v_C(t)$ for all values of t.

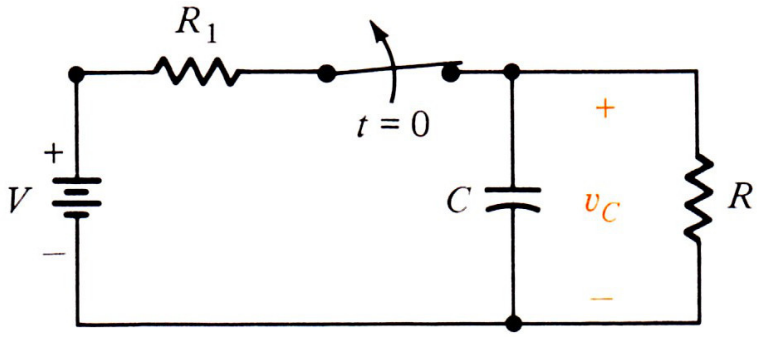


Fig. 5.4 *RC* circuit with a switch.

Solution. Here we have two intervals. In the first, **t < 0s**, we assume things have been steady for a long time and any transient has passed, so we run a DC simulation to find the initial conditions for the second.

```
type
s\dc("e,1,0,V:r1,1,2,r1:c,2,0,c:r,2,0,r")
```

When the simulation is *Done*, we ask for `vc`, the voltage in the capacitor **c**. We get the expression shown below, which is correct.

For the second interval, when **0s** $\leq$ **t**, we do a transient simulation. We give the capacitor as initial condition its voltage from the previous interval.

```
type
s\tr("c,2,0,c,r*v/(r+r1):r,2,0,r")
```

When the simulation is *Done*, we ask for `vc` again. We get the expression below, which is correct.

$$e^{-t/cr}(rv)/(r+r1)$$

Bo2's Drill Exercise 5.1

DRILL EXERCISE 5.1

For the circuit shown in Fig. DE5.1, find a single expression for all t for (a) $v_C(t)$ and (b) $i_C(t)$.
Answer: (a) $6 - 6(1 - e^{-4t})u(t)$ V; (b) $-2e^{-4t}$ A

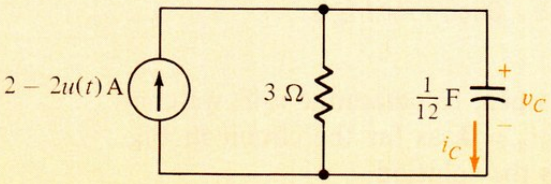


Fig. DE5.1

Solution. Again, we have two intervals: the first for DC, the second for TR.

For the first interval, when **t < 0s**, the current source is 2A. Simulate in DC.

type
```
s\dc("j,0,1,2:r,1,0,3:c,1,0,1/12"):{vc,ic}
```

returns
```
{ 6 , 0 }
```

So the voltage in the capacitor is 6V. We store this value in a variable called $\mathbf{vc}_0$, to use it as initial condition of the capacitor for the second interval.

type
```
vc→vc0
```

For the second interval, when **0s** $\leq$ **t**, we do a transient simulation. The current source is 0A, and the capacitor has an initial condition: voltage $\mathbf{vc}_0$.

type
```
s\tr("j,0,1,0:r,1,0,3:c,1,0,1/12,vc0"):{vc,ic}
```

When the simulation is *Done*, we get the expressions below. They are right.

$$\{6e^{-4t}, -2e^{-4t}\}$$

Bo2's p224 5.2

Solution. Again, we have two intervals: the first for DC, the second for TR.

For the interval when **t < 0s**, the voltage source is 12 V. Simulate in DC.

EXAMPLE 5.2

For the circuit shown in Fig. 5.7, let us determine $v_C(t)$, $i_C(t)$, and $v(t)$ for all time.

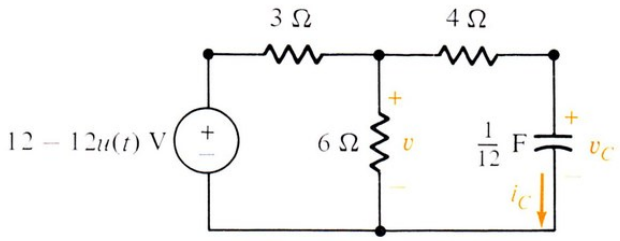


Fig. 5.7 Circuit for Example 5.2.

type
```
s\dc("e,1,0,12:r3,1,2,3:r6,2,0,6:r4,2,3,4:c,3,0,1/12"):{vc,ic,v2}
```

returns
```
{ 8 , 0 , 8 }
```

For the interval when **0s** $\leq$ **t**, the voltage source is 0V. Simulate in TR, giving the capacitor as initial condition its voltage from the previous interval.

type
```
s\tr("e,1,0,0:r3,1,2,3:r6,2,0,6:r4,2,3,4:c,3,0,1/12,8"):{vc,ic,v2}
```

$$\{8e^{-2t}, (-4/3)e^{-2t}, (8/3)e^{-2t}\}$$

Bo2's Drill Exercise 5.2

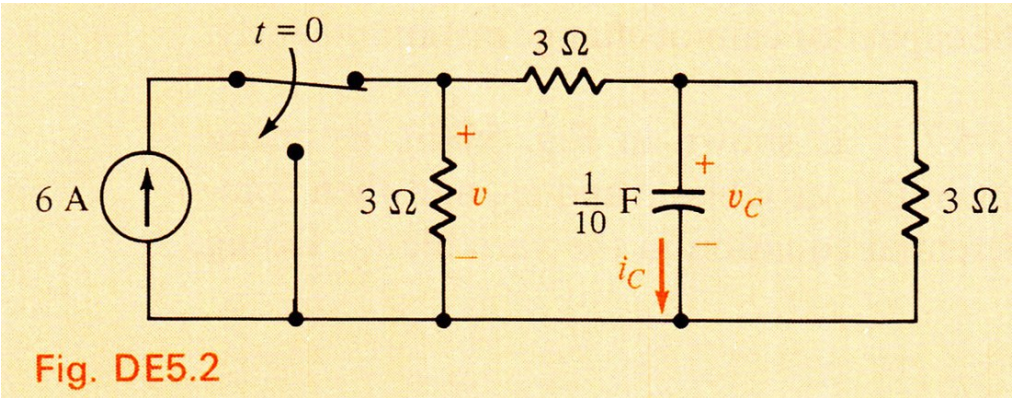


Fig. DE5.2

DRILL EXERCISE 5.2

For the circuit shown in Fig. DE5.2, find an expression for all t for (a) $v_C(t)$, (b) $i_C(t)$, and (c) $v(t)$.

Answer: (a) $6 - 6(1 - e^{-5t})u(t)$ V; (b) $-3e^{-5t}u(t)$ A; (c) $12 - 12u(t) + 3e^{-5t}u(t)$ V

Solution. For **t < 0s**, the 6A source is part of the circuit. Simulate in DC.

type
```
s\dc("j,0,1,6:r10,1,0,3:r12,1,2,3:r20,2,0,3:c,2,0,1/10"):{vc,ic,v1}
```

returns
```
{ 6 , 0 , 12 }
```

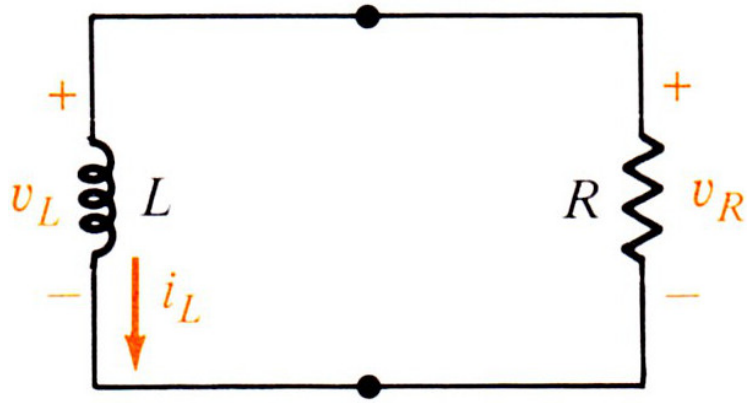
For **0s** $\leq$ **t**, the source is no longer relevant and can be excluded from the circuit. Simulate in TR, giving the capacitor its proper initial condition.

type
```
s\tr("r10,1,0,3:r12,1,2,3:r20,2,0,3:c,2,0,1/10,6"):{vc,ic,v1}
```

$$\{6e^{-5t}, -3e^{-5t}, 3e^{-5t}\}$$

Bo2's Figure 5.10a

Look at the simple resistor-inductor (RL) circuit shown, where at time t=0 the inductor current is $i_L(0)$. Determine $v_L(t)$, $i_L(t)$ and $v_R(t)$ for t≥0.



Since we are already given the initial condition, we only run the transient simulation for t≥0. In our circuit description below, the initial condition is **il0**.

type
```
s\tr("l,1,0,l,il0:r,1,0,r"):{vl,il,vr}
```

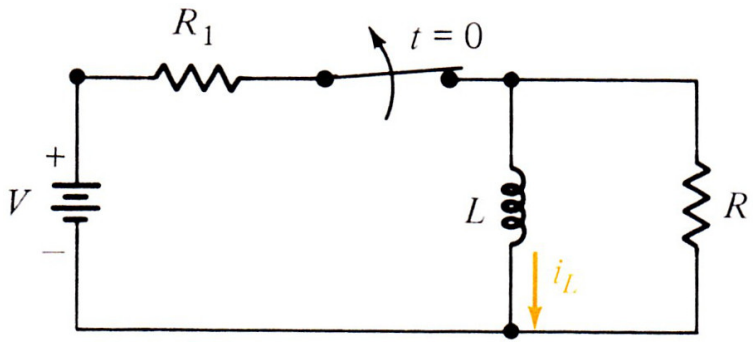
We get the right answers:

$$\{-il0re^{(-rt)/l}, il0e^{(-rt)/l}, -il0re^{(-rt)/l}\}$$

Bo2's Example 5.3

Determine $i_L(t)$ for all t.

Solution. For **t < 0s**, simulate in DC.



type
```
s\dc("e,1,0,v:r1,1,2,r1:l,2,0,l:r2,2,0,r"):il
```

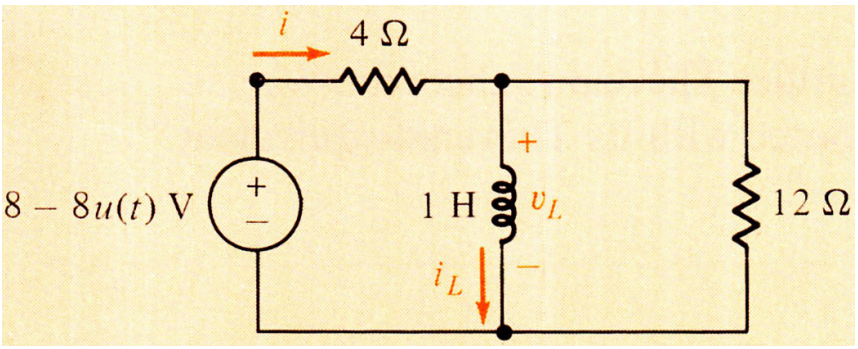
For **t ≥ 0s**, simulate in TR, giving the inductor its initial condition.

type
```
s\tr("l,2,0,l,v/r1:r2,2,0,r"):il
```

$$e^{(-rt)/l}v/r1$$

Bo2's Drill Exercise 5.3

Find $i_L(t)$, $v_L(t)$ and $i(t)$ for all t.



Solution. For **t < 0s**, simulate in DC.

type
```
s\dc("e,1,0,8:r4,1,2,4:l,2,0,1:r12,2,0,12"):{il,vl,ir4}
```

returns
```
{ 2 , 0 , 2 }
```

For **t ≥ 0s**, simulate in TR, giving the inductor its initial condition of 2A.

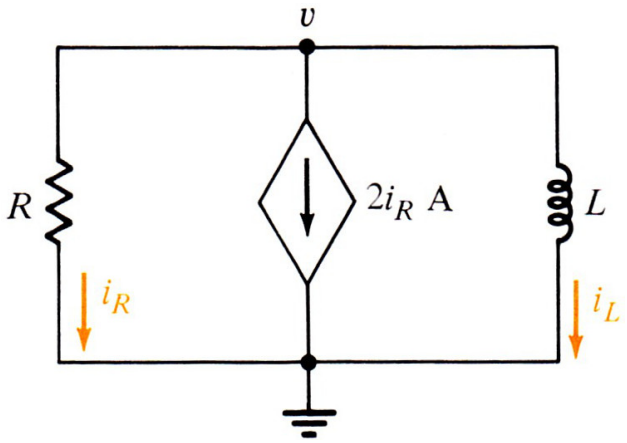
type
```
s\tr("e,1,0,0:r4,1,2,4:l,2,0,1,2:r12,2,0,12"):{il,vl,ir4}
```

$$\{2e^{-3t}, -6e^{-3t}, (3/2)e^{-3t}\}$$

In my machine the simulation took 30 seconds, 12 seconds of these (40%) were used in finding the inverse Laplace of the answers — time that the *only* tool, described above, can save.

Bo2's p230 (Dependent source)

Find $i_L(t)$ for t≥0, given that $i_L(0)$ = 5A.



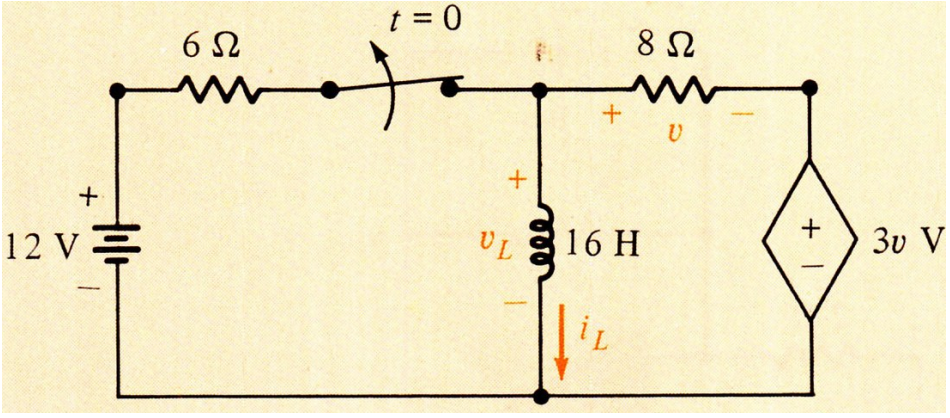
We only run the transient simulation for t≥0, with initial condition 5A.

```
type
s\tr("r,v,0,r:j,v,0,2ir:l,v,0,l,5"):il
```

$$5e^{((-Rt)/3L)}$$

Bo2's Drill Exercise 5.4 (Dependent source)

Find $i_L(t)$ and $v_L(t)$ for all t.



Solution. For **t < 0s**, simulate in DC. This gives us the initial condition.

type
```
s\dc("ei,1,0,12:r6,1,2,6:l,2,0,16:r8,2,3,8:ed,3,0,3vr8"):{il,vl}
```

returns
```
{ 2 , 0 }
```

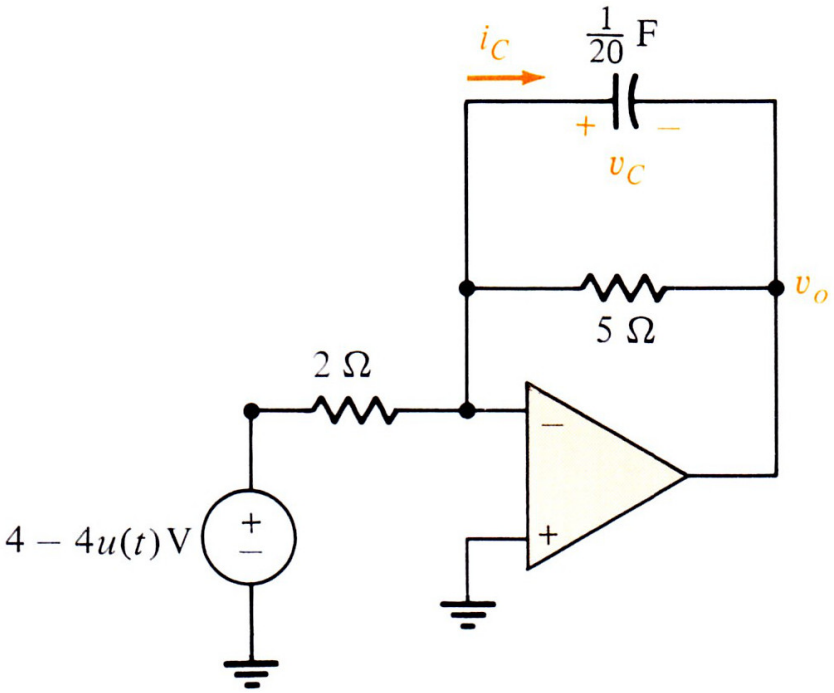
For **t ≥ 0s**, simulate in TR, giving the inductor its initial condition.

type
```
s\tr("l,2,0,16,2:r8,2,3,8:ed,3,0,3vr8"):{il,vl}
```

$$\{2e^{-2t}, -64e^{-2t}\}$$

Bo2's Example 5.5 (Op Amp)

Determine $v_C(t)$, $i_C(t)$ and $v_o(t)$ for all t.



Solution. For **t < 0s**, simulate in DC. This gives us the initial condition.

type
```
s\dc("e,3,0,4:r2,3,1,2:r5,1,o,5:c,1,o,1/20:o,0,1,o"):{vc,ic,vo}
```

returns
```
{ 10 , 0 , -10 }
```

For **t ≥ 0s**, simulate in TR, giving the capacitor its initial condition.

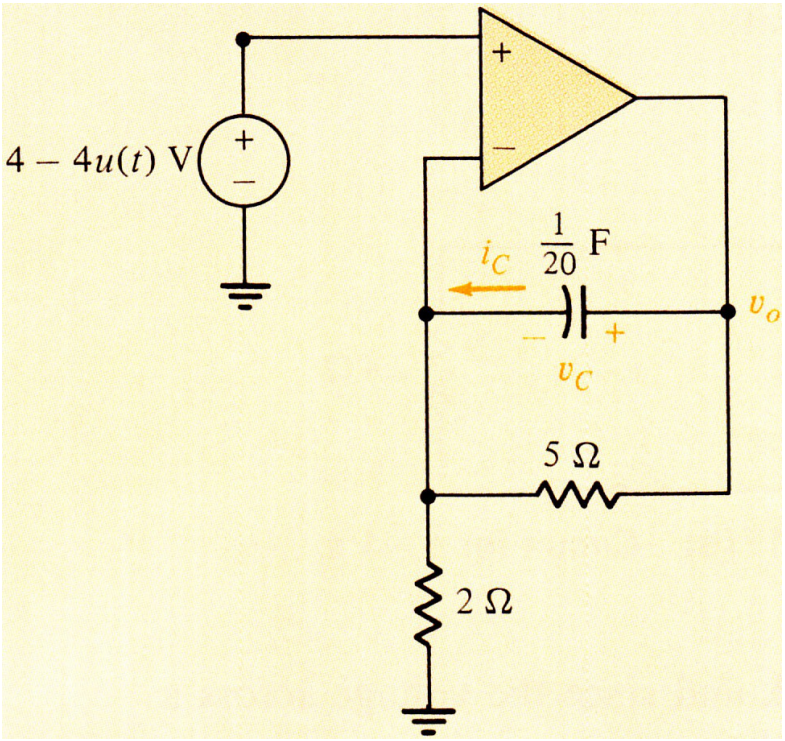
type

```
s\tr("r2,0,1,2:r5,1,o,5:c,1,o,1/20,10:o,0,1,o"):{vc,ic,vo}
```

$$\{10e^{-4t}, -2e^{-4t}, -10e^{-4t}\}$$

Bo2's Drill Exercise 5.5 (Op Amp)

Determine $v_C(t)$, $i_C(t)$ and $v_o(t)$ for all t.



Solution. For **t < 0s**, simulate in DC. This gives us the initial condition.

type

```
s\dc("e,2,0,4:o,2,1,o:c,o,1,1/20:r5,o,1,5:r2,1,0,2"):{vc,ic,vo}
```

returns

```
{ 10 , 0 , 14 }
```

For **t ≥ 0s**, simulate in TR, giving the capacitor its initial condition.

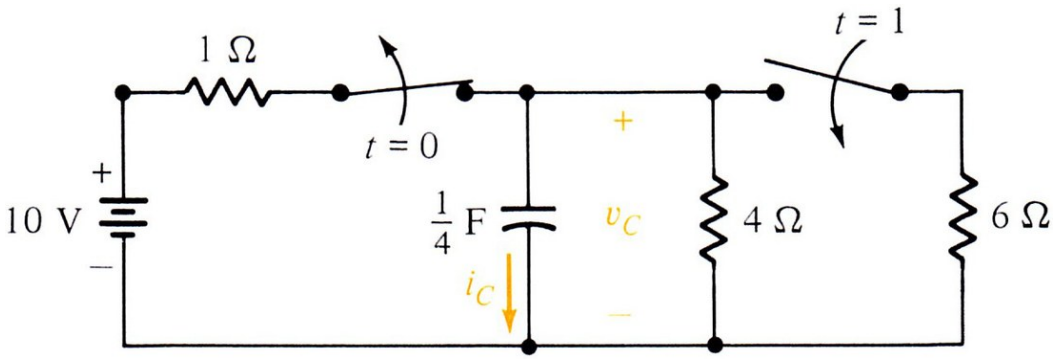
type

```
s\tr("o,0,1,o:c,o,1,1/20,10:r5,o,1,5:r2,1,0,2"):{vc,ic,vo}
```

$$\{10e^{-4t}, -2e^{-4t}, 10e^{-4t}\}$$

Bo2's Example 5.6

In the circuit shown, there are two switches: one that opens at time t=0 and one that closes at time t=1 second. Determine v_C(t) and i_C(t) for all t.



As we mentioned before, every time a switch moves marks the end of one interval and the beginning of another interval for Symbulator.

For the first interval, which corresponds to t<0, we simulate in DC.

type
```
s\dc("e,2,0,10:r1,2,1,1:c,1,0,1/4:r4,1,0,4"):{vc,ic}
```

returns
```
{ 8 , 0 }
```

The voltage in the capacitor at the end of this interval will serve as the initial condition of the capacitor for the next interval. The second interval runs between 0 and 1 second, i.e. $0 < t \leq 1$ second. We simulate in TR.

type
```
s\tr("c,1,0,1/4,8:r4,1,0,4"):{vc,ic}
```

$$\{8e^{-t}, -2e^{-t}\}$$

The capacitor's voltage at the end of this second interval is the initial condition for the third. We can use its exact value, `8*e^-1`, but the textbook prefers using its approximate value, i.e. `2.943`.

The third interval corresponds to t > 1 second. We simulate in TR. To make things easier for Symbulator, we replace the resistors by their equivalent.

type
```
s\tr("c,1,0,1/4,2.943:re,1,0,[4,6]")
```

Ask for `{vc,ic}`. The expressions we get are equivalent to:

$$\{2.943e^{(-5(t-1))/3}, -1.226e^{(-5(t-1))/3}\}$$

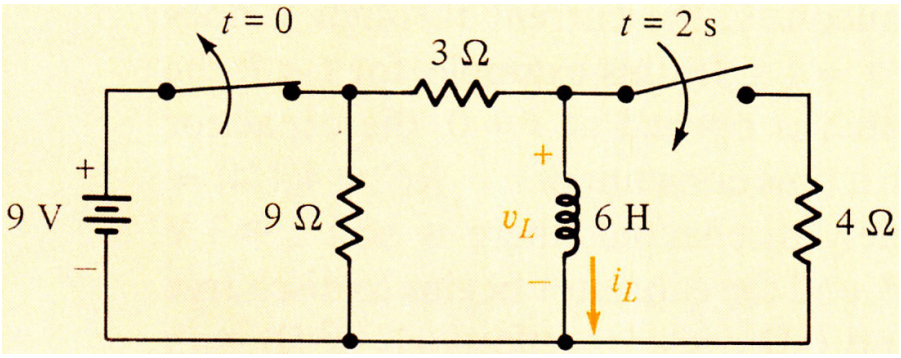
These are the right answers, as can be seen by checking the book's answers.

$$v_C(t) = \begin{cases} 8\ \text{V} & \text{for} \quad t < 0 \\ 8e^{-t}\ \text{V} & \text{for} \quad 0 \le t < 1\ \text{s} \\ 2.943e^{-5(t-1)/3}\ \text{V} & \text{for} \quad t \ge 1\ \text{s} \end{cases}$$

$$i_C(t) = \begin{cases} 0\ \text{A} & \text{for} \quad t < 0 \\ -2e^{-t}\ \text{A} & \text{for} \quad 0 \le t < 1\ \text{s} \\ -1.226e^{-5(t-1)/3}\ \text{A} & \text{for} \quad t \ge 1\ \text{s} \end{cases}$$

Bo2's Drill Exercise 5.6

Let's see another example where a TR simulation is done for an interval that starts at a time other than t=0. Determine $i_L(t)$ and $v_L(t)$ for all t.



For the first interval, which corresponds to t<0, we simulate in DC.

type
```
s\dc("e,1,0,9:r9,1,0,9:r3,1,2,3:l,2,0,6"):{il,vl}
```

returns
```
{ 3 , 0 }
```

The 3A current in the inductor is its initial condition for the second interval, which goes from 0 to 2 seconds, i.e. $0 < t \le 2$ seconds. We simulate in TR.

type
```
s\tr("r,1,0,9+3:l,1,0,6,3"):{il,vl}
```

$$\{3e^{-2t}, -36e^{-2t}\}$$

The inductor's current at the end of this second interval is the initial condition for the third. Find its approximate value thus:

type
```
iL|t=2.
```

.055

The third interval corresponds to t > 2 second. We simulate in TR. To make things easier for Symbulator, we replace the resistors by their equivalent.

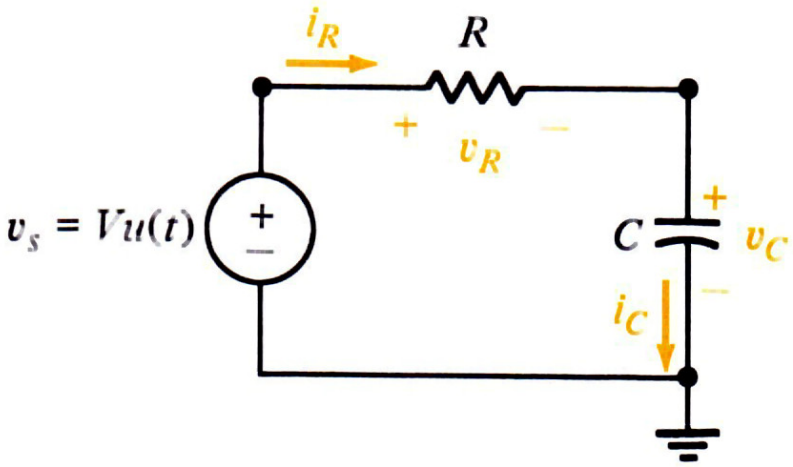
type
```
s\tr("r,1,0,[9+3,4]:l,1,0,6,.055"):{il,vl}
```

$$\{.055e^{(-(t-2)/2)}, -.165e^{(-(t-2)/2)}\}$$

These are the kind of expressions your book or professor are looking for.

Bo2's Figure 5.19

In the following circuit, assume that all the initial conditions are zero. The source is a step function **u(t)** with value **V** volts. Find i_R, v_R, i_C and v_C.



As discussed before, a step source with symbolic value requires that we use the `u(t)` nomenclature to describe it. Since the value is V volts, we describe the source as `V*u(t)`.

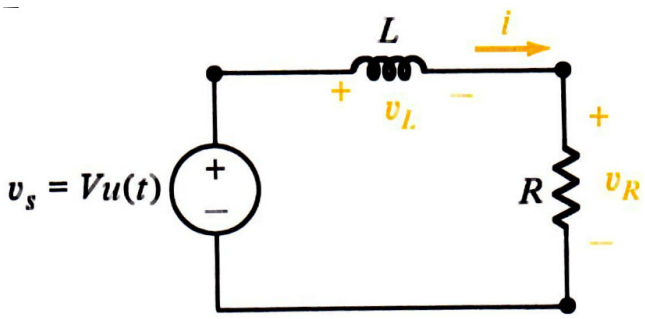
type
```
s\tr("e,1,0,V*u(t):r,1,2,r:c,2,0,c,0"):{vc,vr,ic}
```

There is no need to ask for i_R, since i_R=i_C. We get the following expressions:

$$\{v - ve^{(-t/(cr))}, ve^{(-t/(cr))}, (v/r)e^{(-t/(cr))}\}$$

Bo2's Figure 5.24

In the following circuit, assume that all the initial conditions are zero. The source is a step function **u(t)** with value **V** volts. Find i_R, v_R, i_L and v_L.



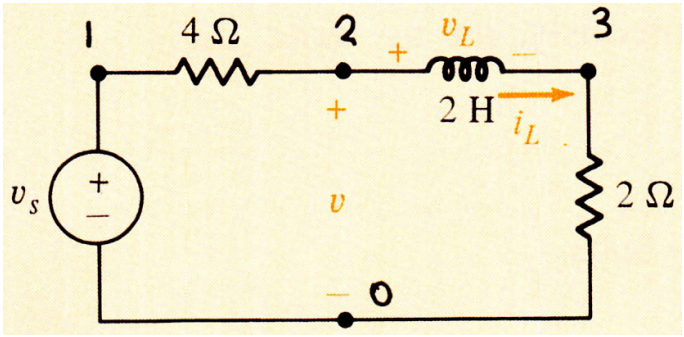

```
type
s\tr("e,1,0,V*u(t):l,1,2,l,0:r,2,0,r"):{vl,vr,il}
```

There is no need to ask for i_R, since in a series circuit it will be identical to i_L.

$$\{ve^{(-rt)/l}, v(1-e^{(-rt)/l}), (v/r)(1-e^{(-rt)/l})\}$$

Bo2's Drill Exercise 5.7

For the circuit, with a step source of 12 volts, find i_L, v_L, i_L and v.



As discussed before, a step source with numerical value can be described without the `u(t)` nomenclature. Thus, we describe the source as `12`.

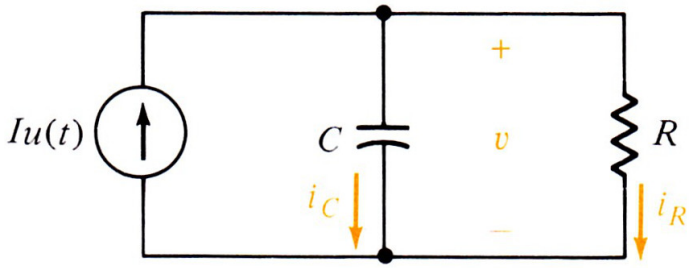
```
type
s\tr("e,1,0,12:r4,1,2,4:l,2,3,2,0:r2,3,0,2"):{il,vl,v2}
```

$$\{2-2e^{-3t}, 12e^{-3t}, 8e^{-3t}+4\}$$

Bo2's Figure 5.26

In the following circuit, assume that all the initial conditions are zero. The source is a step function **u(t)** with value **I** amperes. Find i_C, i_R, and v.

As discussed before, a step source with symbolic value requires that we use the `u(t)` nomenclature to describe it. Since the value is I amperes, we describe the source as `I*u(t)`. Thus:

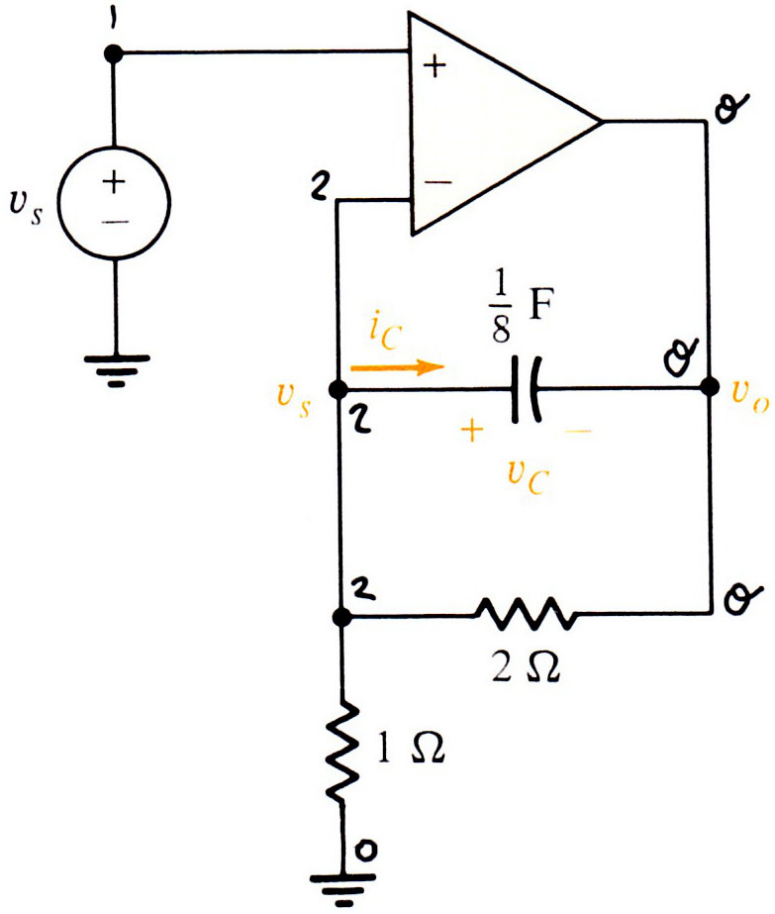


type
```
s\tr("j,0,1,i*u(t):c,1,0,c,0:r,1,0,r"):{v1,ir,ic}
```

$$\{(i - ie^{-t/(cr)}) \cdot r, i - ie^{-t/(cr)}, ie^{-t/(cr)}\}$$

Bo2's Example 5.7 (Op Amp)

Suppose that $v_s(t)=u(t)$. Find v_C, i_C, and v_o.



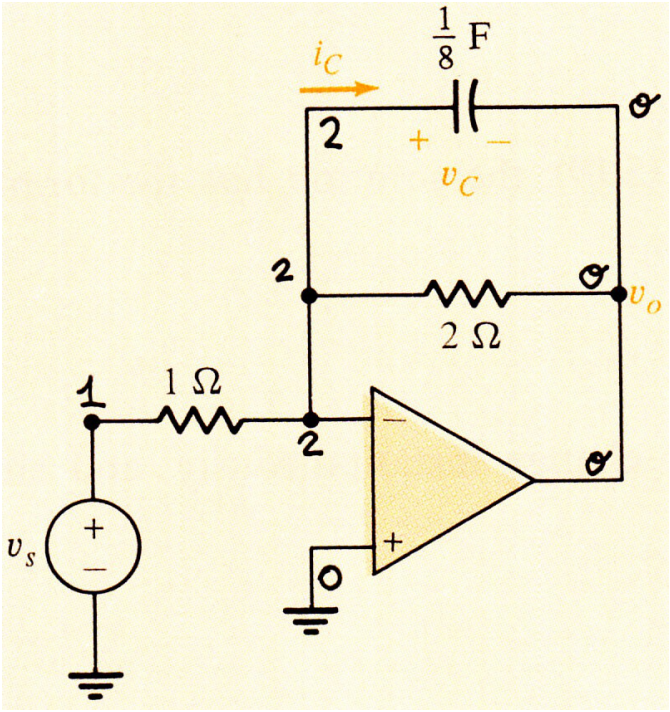
This source is a step source with a value of 1 volt. Since its value is numerical, the `u(t)` nomenclature is not needed. We describe the source plainly as `1`.

type
```
s\tr("e,1,0,1:o,1,2,o:c,2,o,1/8,0:r2,2,o,2:r1,2,0,1"):{vc,ic,vo}
```

$$\{2e^{-4t} - 2, -e^{-4t}, 3 - 2e^{-4t}\}$$

Bo2's Drill Exercise 5.8 (Op Amp)

Suppose that $v_s(t)=u(t)$. Find v_C, i_C, and v_o.



Again, following the same logic as above, we describe the source plainly as `1`.

```
type
s\tr("e,1,0,1:r1,1,2,1:r2,2,o,2:c,2,o,1/8,0:o,0,2,o"):{vc,ic,vo}
```

$$\{2 - 2e^{-4t}, e^{-4t}, 2e^{-4t} - 2\}$$

Bo2's p249 F5.28

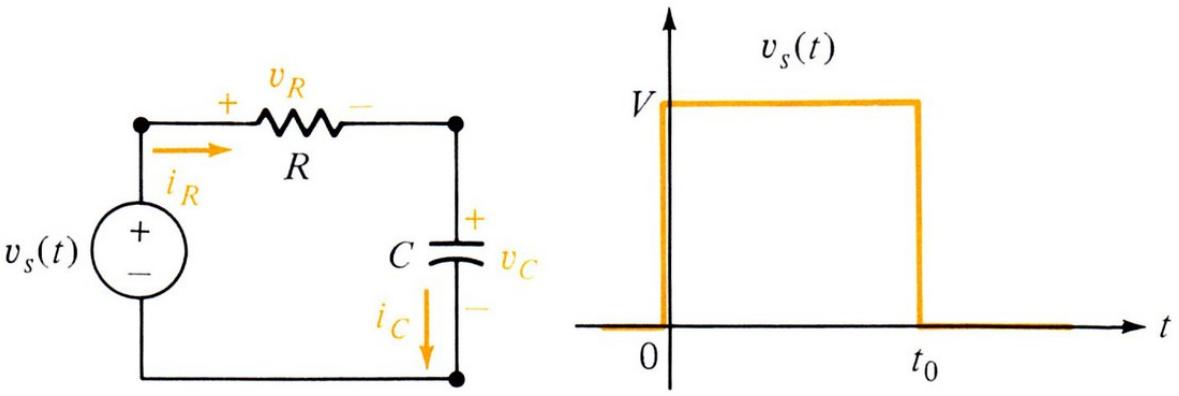
For the circuit left, the source is $v_s(t) = V\,u(t) - V\,u(t\text{-}t_0)$. This is shown in the plot to the right. Given this, find the voltage drop in the capacitor for all t.

Because the source has two steps, this problem has to be analyzed in two intervals. For the first interval, the initial conditions are zero and the source value is V*u(t). Our circuit description for the first interval is given below.

```
type
s\tr("e,1,0,V*u(t):r,1,2,r:c,2,0,c,0"):vc
```

The expression for the capacitor's voltage drop in the first interval is:

$$v - ve^{(-t/(cr))}$$



The second interval starts at t_0. In it, the initial condition of the capacitor is given by the value of the expression above when t=t_0. Here is how we find it:

```
type
vc|t=to
```

The circuit description for the second interval uses that expression as the capacitor's initial condition. The source is now 0 volts — another way of saying it becomes a short — so it need not appear in the description.

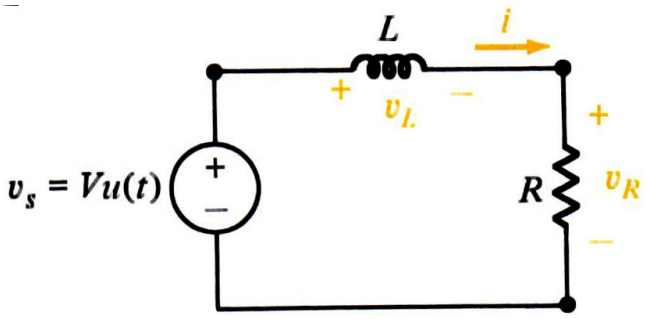
$$(v - ve^{(-to/(cr))})e^{((-t+to)/(cr))}$$

```
type
s\tr("r,0,2,r:c,2,0,c,v-e^(-to/(c*r))*v"):vc
```

This is the expression for the capacitor's voltage drop in the second interval.

Bo2's Drill Exercise 5.9

Find the current through the inductor if the source is $v_s(t) = V\,u(t) - V\,u(t\text{-}t_o)$.



Because the source has two steps, this problem has to be analyzed in two intervals. For the first interval, the initial conditions are zero and the source value is V*u(t). Our circuit description for the first interval is given below.

```
type
s\tr("e,1,0,V*u(t):l,1,2,l,0:r,2,0,r"):il
```

The expression for the inductor's current in the first interval is:

$$(v/r)(1 - e^{((-rt)/l)})$$

The second interval starts at t_0. In it, the initial condition of the inductor is given by the value of the expression above when t=t_0. Here is how we find it:

```
type
il|t=to
```

The circuit description for the second interval uses that expression as the inductor's initial condition. The source becomes a short, so it need not appear in the description.

$$(v/r)(1 - e^{((-tor)/l)})e^{((-rt)/l)}$$

```
type
s\tr("l,0,2,l,v/r-e^(-to*r/l)*v/r:r,2,0,r"):il
```

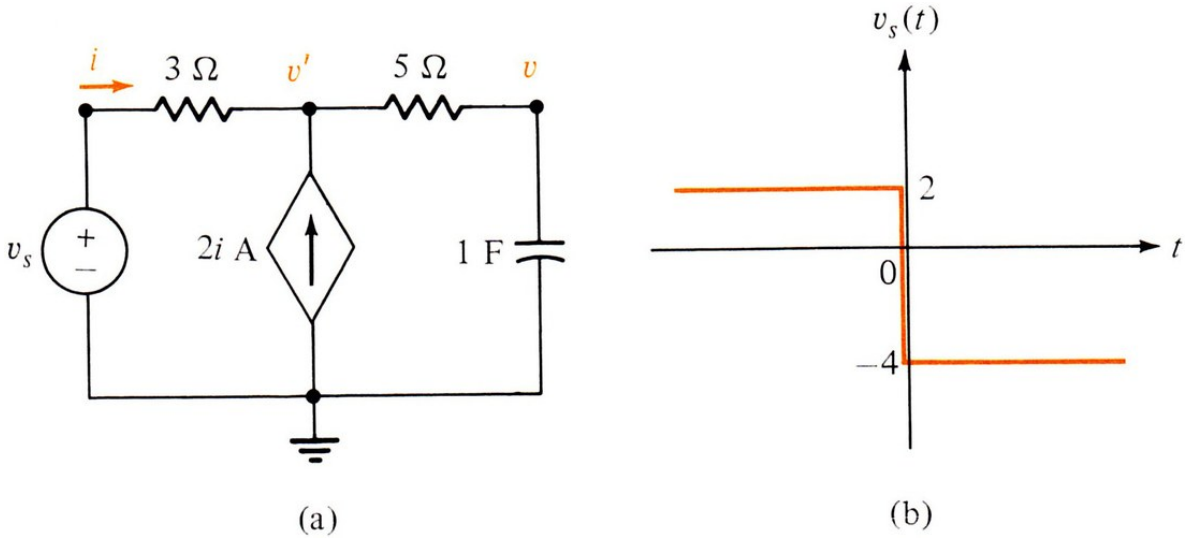
To put this in terms of the same t as the first interval, we replace **t** with **t-t**$_0$:

$$(v/r)(1 - e^{((-tor)/l)})e^{((-r(t-to))/l)}$$

This is the expression for the inductor's current in the second interval.

Bo2's Figure 5.36

For the circuit shown in (a), find the voltage drop in the capacitor, given that the voltage source is as shown in (b).



This problem requires two intervals. In the first interval, a DC analysis of the circuit with a source of 2 V gives us the voltage in the capacitor.

type
```
s\dc("e,1,0,2:r3,1,2,3:r5,2,3,5:c,3,0,1:j,0,2,2ir3"):vc
```

2

This is the initial condition for the capacitor in the second interval, which we analyze using TR and a source value of -4 V.

type
```
s\tr("e,1,0,-4:r3,1,2,3:r5,2,3,5:c,3,0,1,2:j,0,2,2ir3"):vc
```

$$6e^{(-t/6)} - 4$$

This is the expression for the voltage drop in the capacitor after t=0.

The only tool

Bo2's Drill Exercise 5.11

For the circuit of **Bo2's Example 5.7 (page 139)**, change the value of the capacitor to ¼ F. Find v_C(t) for the case that the source is v_S(t)=1 V for t<0 and 3V for t≥0.

We let Symbulator know that we are only interested in one variable: vc.

type
```
s\only("vc")
```

Then we run the simulation for the first interval, just as we did before, but with the new capacitor and source values.

type
```
s\dc("e,1,0,1:o,1,2,o:c,2,o,1/4:r2,2,o,2:r1,2,0,1"):vc
```

-2

This is the initial condition for the next interval. Before we run the simulation for the second interval, we let Symbulator know, again, that we are only interested in one variable: vc.

type
```
s\only("vc")
```

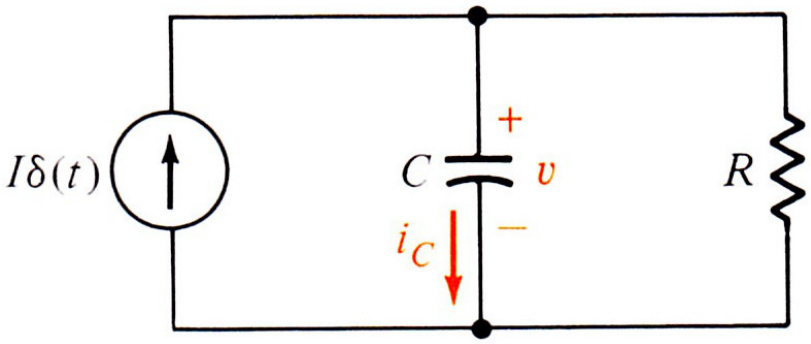
Then we run the simulation for the second interval, just as we did before, but with the new capacitor and source values.

type
```
s\tr("e,1,0,3:o,1,2,o:c,2,o,1/4,-2:r2,2,o,2:r1,2,0,1"):vc
```

$$4e^{(-2t)} - 6$$

Bo2's Figure 5.38 (Impulse)

Find the voltage drop and current in the capacitor, given a source of I δ(t) A, where δ(t) is the impulse function and I is a constant.



As we mentioned before, impulse sources must be described using the δ(t) nomenclature. To save some typing time, you find it in the menu.

type
```
s\tr("j,0,1,i*δ(t):c,1,0,c,0:r,1,0,r")
```

Once the simulation completes, we ask for the variables of interest:

$$(i/c)e^{(-t/(cr))}$$

$$(-i/(cr))e^{(-t/(cr))}$$

type
```
vc
```

type
```
ic
```

Bo2's Drill Exercise 5.13 (Impulse)

For the circuit of **Bo2's Example 5.7 (page 139)**, find v_C(t), i_C(t) and v_o(t) due to an impulse voltage source of v_S(t)= δ(t) (that is to say, a 1V impulse in t=0.)

Nothing new here. Again we use the δ(t) nomenclature for the source.

type
```
s\tr("e,1,0,δ(t):o,1,2,o:c,2,o,1/8,0:r2,2,o,2:r1,2,0,1"):{vc,ic,vo}
```

$$\{-8e^{-4t}, 4e^{-4t}, 8e^{-4t}\}$$

Bo2's Figure 5.39 (Ramp)

Find i_L(t) and v_L(t) for a circuit that has in parallel a resistor R, an inductor L and a current source with a ramp value I r(t). Initial conditions are zero.

A ramp value I r(t) means a value of I t. We describe this source simply as `i*t`.

type
```
s\tr("j,0,1,i*t:r,1,0,r:l,1,0,l,0"):{il,vl}
```

$$\{((il)/r)(e^{((-rt)/l)} - 1) + it, il(1 - e^{((-rt)/l)})\}$$

Bo2's Drill Exercise 5.14 (Ramp)

For the circuit of **Bo2's Example 5.7 (page 139)**, find v_C(t) and i_C(t) due to a ramp input voltage of v_S(t)= r(t).

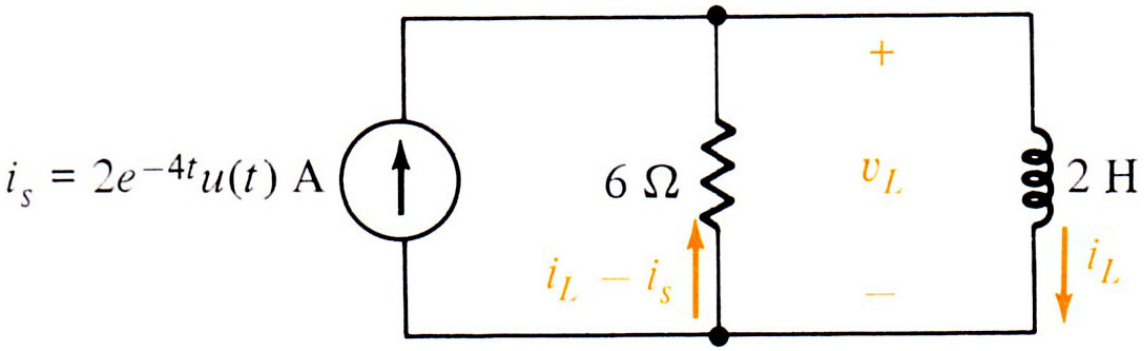
Nothing new here. The command below should be clear to you by now.

type
```
s\tr("e,1,0,t:o,1,2,o:c,2,o,1/8,0:r2,2,o,2:r1,2,0,1"):{vc,ic}
```

$$\{(1/2)(1 - e^{-4t}) - 2t, (1/4)(e^{-4t} - 1)\}$$

Bo2's Example 5.12 (Exponential)

Assume initial conditions zero. Find i_L(t) and v_L(t), given that i_S(t) = 2 e^{-4t} A.



Describe the source as given to you by the problem, using an exponential.

type
```
s\tr("j,0,1,2e^(-4t):r,0,1,6:l,1,0,2,0"):{il,vl}
```

$$\{6e^{-3t} - 6e^{-4t}, 48e^{-4t} - 36e^{-3t}\}$$

Bo2's p265 (Exponential)

Repeat **Bo2's Example 5.12 (page 145)** if the current source is now $i_S(t) = 2\,e^{-3t}$ A.

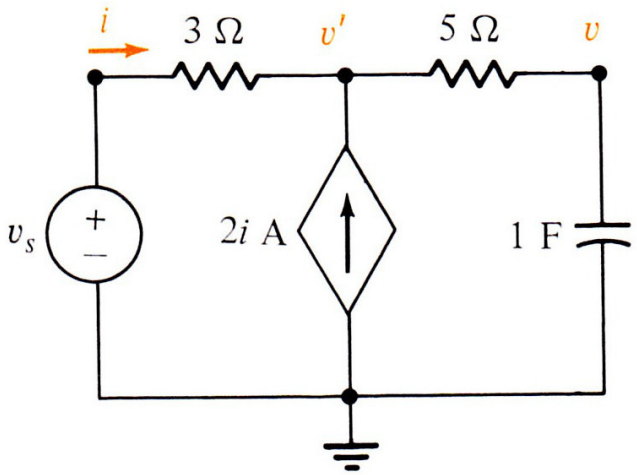
All we do is change one digit in the circuit description: replace 4 with 3.

```
type
s\tr("j,0,1,2e^(-3t):r,0,1,6:l,1,0,2,0"):{il,vl}
```

$$\{6te^{-3t}, (12 - 36t)e^{-3t}\}$$

Bo2's Example 5.13 (Exponential)

Find the voltage drop in the capacitor and the current in the 3 Ω resistor, if the source value is $v_S(t)=18e^{-t/2}$.




```
type
s\tr("e,1,0,18e^(-t/2):r3,1,2,3:r5,2,3,5:c,3,0,1,0:j,0,2,2ir3"):{vc,ir3}
```

$$\{9e^{(-t/6)} - 9e^{(-t/2)}, (3/2)e^{(-t/2)} - (1/2)e^{(-t/6)}\}$$

Bo2's Drill Exercise 5.16 (Exponential)

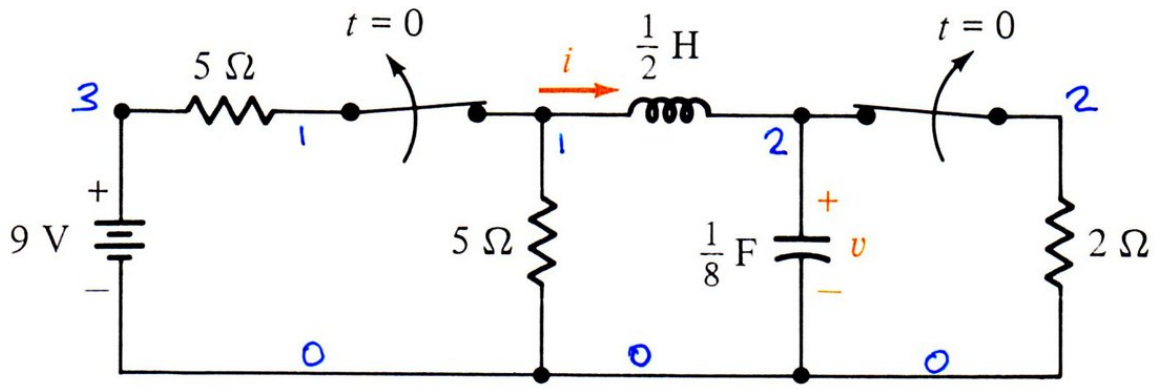
For the circuit of **Bo2's Example 5.7 (page 139)**, find $v_C(t)$ and $i_C(t)$ for the case when the source's value is $v_S(t)= 2\,e^{-4t}$. As before, assume initial conditions are zero.

```
type
s\tr("e,1,0,2e^(-4t):o,1,2,o:c,2,o,1/8,0:r2,2,o,2:r1,2,0,1"):{vc,ic}
```

$$\{-16te^{-4t}, (8t - 2)e^{-4t}\}$$

Bo2's Example 6.1

Find the current in the inductor ***i***(t) and the voltage drop in the capacitor ***v***(t).



Since we have switches changing in time t=0, this is a two interval problem. The first interval is for t<0. We can analyze the first interval using DC.

type
```
s\dc("e,3,0,9:r2,3,1,5:r1,1,0,5:l,1,2,1/2:c,2,0,1/8:r3,2,0,2"):{vc,il}
```

returns
```
{2,1}
```

These are the initial conditions for the second interval. The second is for t≥0. We can analyze the second interval using TR. We only care about `vc` and `il`.

type
```
s\only("vc,il"):s\tr("r1,1,0,5:l,1,2,1/2,1:c,2,0,1/8,2"):{vc,il}
```

$$\{4e^{-2t} - 2e^{-8t}, 2e^{-8t} - e^{-2t}\}$$

Notice that the use of the only tool saves approximately 10 seconds here.

The plot tool

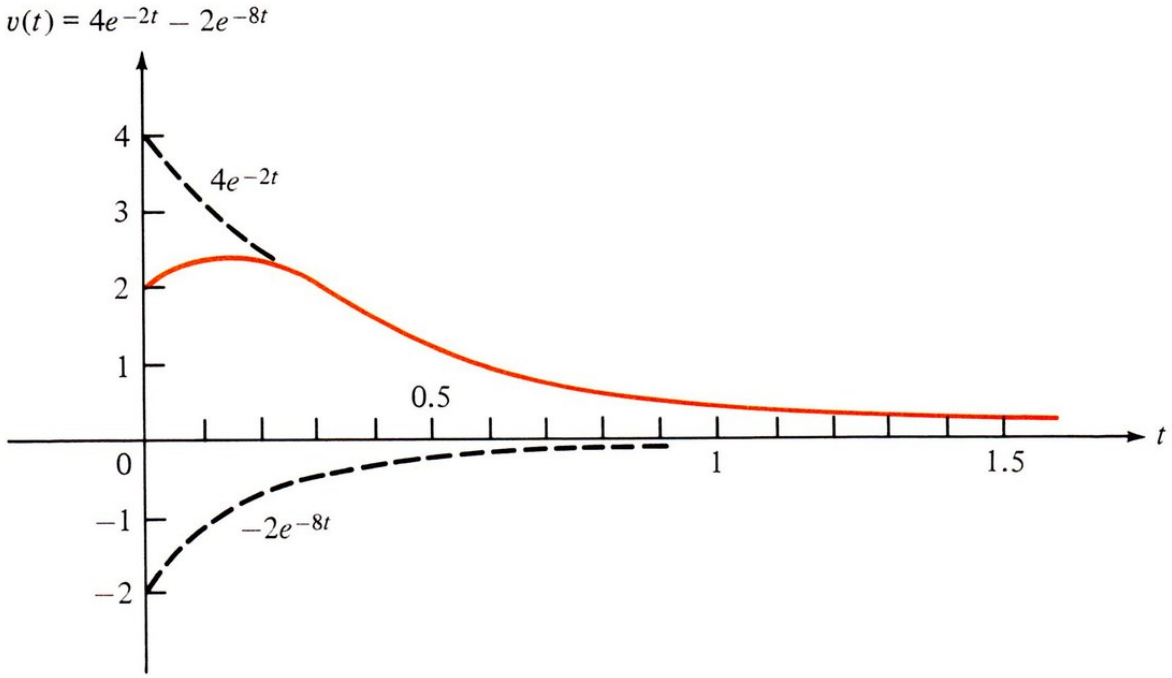
Imagine, for example, that in **Bo2's Example 6.1 (page 147)** we are asked to plot ***i***(t) and ***v***(t) for times between 0 s and 1.5 s. The first step is to run the `plot` tool:

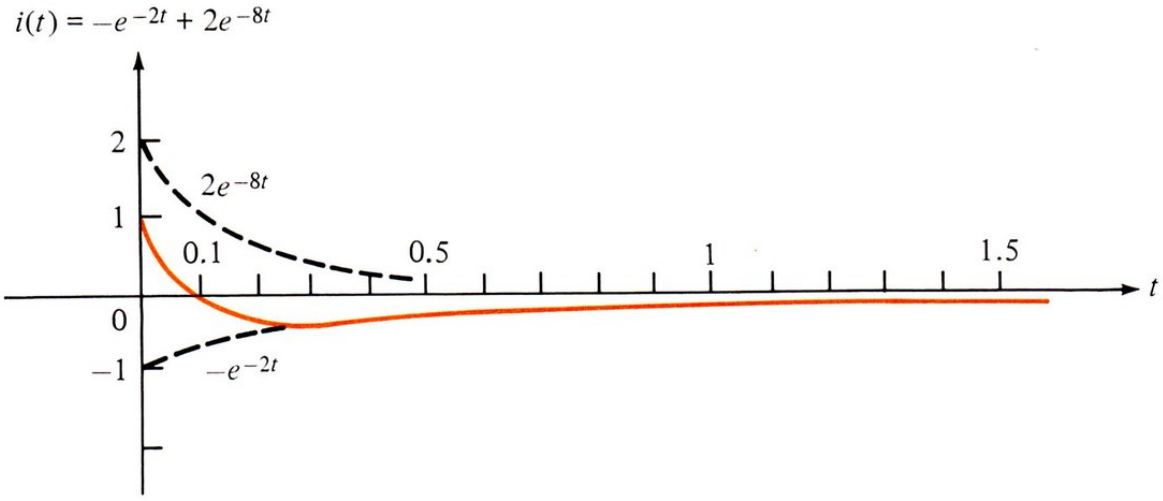
type
```
s\plot()
```

A window opens, asking you to enter three things: first, a function of time; second, a minimal time; and third, a maximal time. Let's plot $v_C(t)$ first. Unless you have deleted the value of the variable **vc**, you should have the answer to the simulation above stored in it. So enter `vc` as the function, `0` as the minimal time and `1.5` as the maximal time.

Press Enter, and wait a little. The graph of the voltage drop in the capacitor between time 0 and 1.5 seconds should appear in the screen.

Compare it to the graph given by the textbook. Repeat this procedure giving `il` as the function. Compare the resulting graph to that given by the textbook.

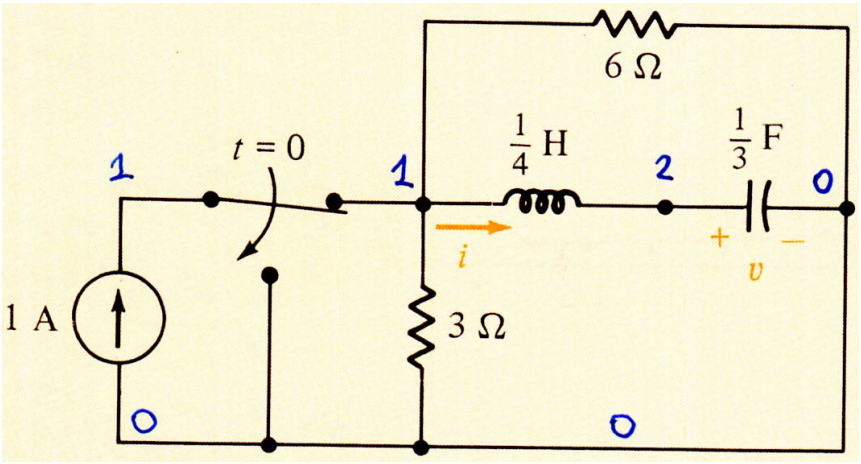




Bo2's Drill Exercise 6.1

For the circuit shown, find ***v***(t) and ***i***(t) for t≥0.

Nothing new here. This is a two interval problem, as others seen before. The first interval, for t<0, is analyzed in DC to find the initial conditions.



type
```
s\dc("j,0,1,1:r3,1,0,3:r6,1,0,6:l,1,2,1/4:c,2,0,1/3"):{vc,il}
```

returns
```
{2,0}
```

These conditions are used in the TR analysis of the second interval, for t≥0.

type
```
s\only("vc,il"):s\tr("r3,1,0,3:r6,1,0,6:l,1,2,1/4,0:c,2,0,1/3,2")
```

type
```
vc
```

type
```
il
```

$$3e^{-2t} - e^{-6t}$$

$$2e^{-6t} - 2e^{-2t}$$

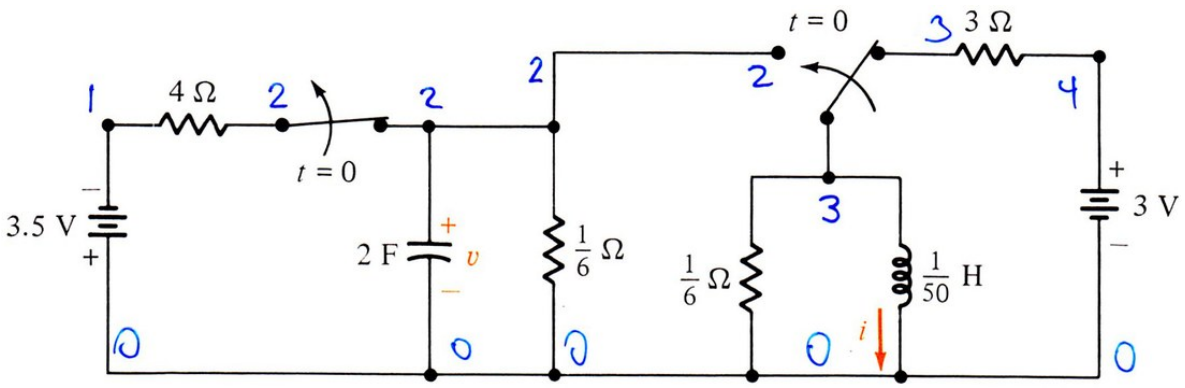
Using `only` saves about 13 seconds, or about a third of the calculation time, because only about 8 seconds are used finding the inverse Laplace of the selected answers, instead of 21 seconds finding it for all the answers.

Bo2's Example 6.2

Find ***i***(t) and ***v***(t) for the circuit shown.

I decided to simulate this using only fractions, not decimals.

So I convert the 3.5V value to its exact fractional equivalent, thus:



type
```
exact(3.5)
```

7/2

Because of the switches, this is a two interval problem. The first interval is to be analyzed using DC. Notice that for t<0 the circuit is such that we basically have two separate circuits. We simulate each one of them separately:

type
```
s\dc("e1,0,1,7/2:r1,1,2,4:c,2,0,2:r2,2,0,1/6"):vc
```

-7/50

type
```
s\dc("r3,3,0,1/6:l,3,0,1/50:r4,3,4,3:e2,4,0,3"):il
```

1

The second interval, for t≥0, is analyzed using TR. I could have described the circuit again from scratch, but out of laziness I copied the descriptions from the DC simulations. To avoid renaming nodes, I simulated the right switch as a short between nodes **2** and **3**.

type
```
s\only("vc,il"):s\tr("c,1,0,2,-7/50:r,1,0,[1/6,1/6]:l,1,0,1/50,1")
```

type
```
vc
```

type
```
il
```

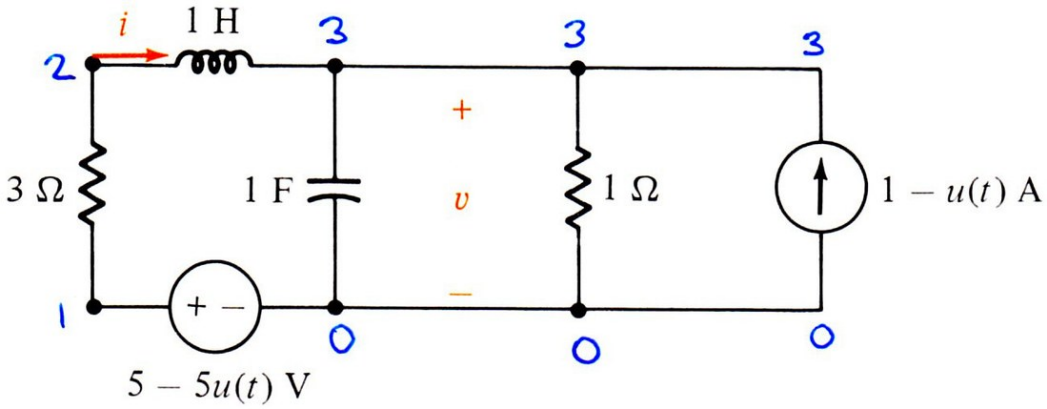
$$(-7/50)e^{-3t}\cos(4t) - (1/50)e^{-3t}\sin(4t)$$

$$e^{-3t}\cos(4t) - e^{-3t}\sin(4t)$$

The TR simulation took 30 s in my calculator. These are the right answers.

Bo2's Example 6.3

Find ***i***(t) and ***v***(t) for the circuit shown.



Since the sources change value, this is a two interval problem. For t<0, in DC:

type
```
s\dc("e,1,0,5:r3,1,2,3:l,2,3,1:c,3,0,1:r1,3,0,1:j,0,3,1"):{vc,il}
```

returns
```
{2,1}
```

Using these as initial conditions, now we use TR for t≥0:

type
```
s\only("vc,il"):s\tr("r3,0,2,3:l,2,3,1,1:c,3,0,1,2:r1,3,0,1"):{vc,il}
```

$$\{(3t+2)e^{-2t}, (1-3t)e^{-2t}\}$$

Is the textbook wrong?

Anybody who has written a textbook, or read one for that matter, can attest that errors in the problems or the answers come with the territory. (I am sure this book has many.) We are all only human. But errors in an engineering textbook can be *very* frustrating to a student: confusion and hair-pulling ensue. If you think you have found one in your circuit analysis textbook, Symbulator can help you confirm whether your guess is right.

Below is an example of using Symbulator to confirm an error in a textbook.

Bo2's Drill Exercise 6.4

My copy of Bo2 is second-hand, and this problem had scribbles on it. The previous owner evidently struggled with it and concluded that the inductor had to be ½ H, not the 1 H the schematic says. Symbulator can show he is right and the book wrong. First, solve the circuit as printed, with a 1 H inductor.

DRILL EXERCISE 6.4

For the *RLC* circuit shown in Fig. DE6.4, suppose that $v_s(t) = 2 - 2u(t)$ V. Find $i(t)$ and $v(t)$ for $t \geq 0$.

Answer: $8te^{-4t} + 2e^{-4t}$ A; $-16te^{-4t}$ V

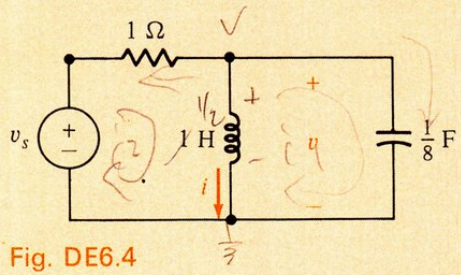


Fig. DE6.4

```
type
s\dc("e,2,0,2:r,1,2,1:l,1,0,1:c,1,0,1/8"):{il,vc}
```

```
returns
{2,0}
```

```
type
s\only("il,vc"):s\tr("r,1,0,1:l,1,0,1,2:c,1,0,1/8,0"):{il,vc}
```

The answer we get doesn't match the answer provided by the book. Now let's repeat the TR simulation using the ½ H value for the inductor:

```
type
s\only("il,vc"):s\tr("r,1,0,1:l,1,0,1/2,2:c,1,0,1/8,0"):{il,vc}
```

Now the answer matches the book's. An expert user can reach for the `fd` gate and the `t2s` tool, which we have not discussed yet, to find the inductor's value from the book's answer. This is how I found the ½ H myself, working backwards:

```
type
s\t2s(-16*t*e^(-4*t)):
```

-16/(s+4)^2

```
type
s\t2s((8*t+2)*e^(-4*t))
```

2*(s+8)/(s+4)^2

```
type
s\fd("r,1,0,1:l,1,0,l,2:c,1,0,1/8,0"):vc
```

-16*l/(l*s^2+8*l*s+8)

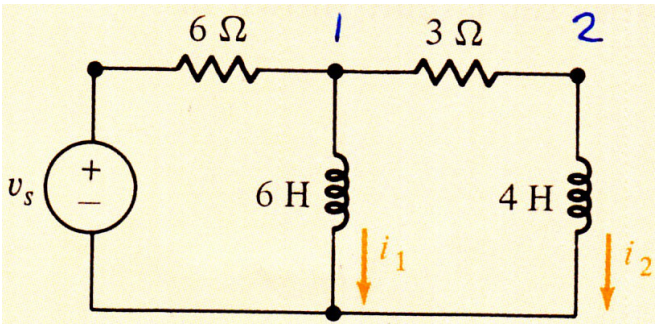
type
```
solve(-16/(s+4)^2=-16*l/(l*s^2+8*l*s+8),l)
```

l=1/2

The `fd` gate and `t2s` tool are used for frequency domain analysis, and will be discussed separately in a subsequent part of Symbulator's documentation.

Bo2's Drill Exercise 6.5

In the circuit, suppose that $v_S(t) = 12 - 12\ u(t)$ V. Find $i_2(t)$ for $t \geq 0$.



type
```
s\dc("e,3,0,12:r6,3,1,6:l1,1,0,6:r3,1,2,3:l2,2,0,4"):{il1,il2}
```

returns
```
{2,0}
```

type
```
s\only("il2"):s\tr("r6,0,1,6:l1,1,0,6,2:r3,1,2,3:l2,2,0,4,0"):il2
```

$$(12/11)(e^{-3t} - e^{-t/4})$$

Bo2's Drill Exercise 6.6 (Op Amp)

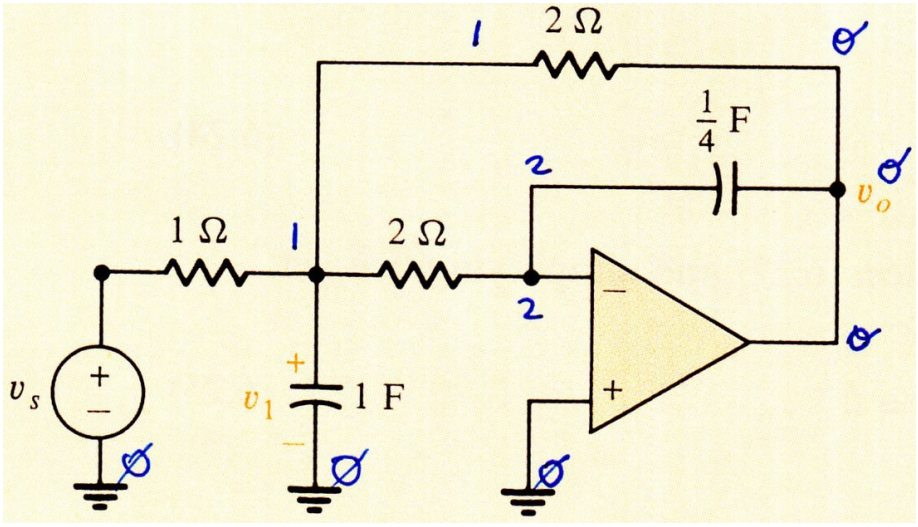
In the circuit, suppose that $v_S(t) = 2 - 2\ u(t)$ V. Find $v_o(t)$ for $t \geq 0$.

Simulate the first interval in DC to find the initial conditions of the capacitors.

type
```
s\dc("e,3,0,2:r1,3,1,1:r2,1,2,2:r3,1,o,2:ca,1,0,1:
cb,2,o,1/4:o,0,2,o"):{vca,vcb}
```

returns
```
{0,4}
```

Then simulate the second interval in TR, to find the voltage in node *o*.

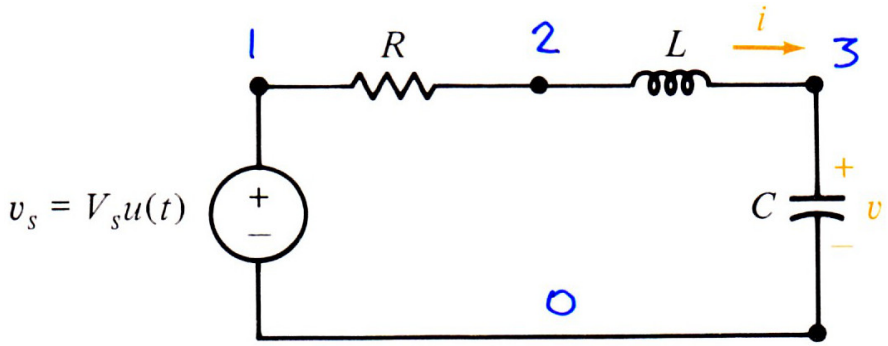


$$(-4t-4)e^{-t}$$

```
type
s\only("vo"):s\tr("r1,0,1,1:r2,1,2,2:r3,1,o,2:
ca,1,0,1,0:cb,2,o,1/4,4:o,0,2,o"):vo
```

Bo2's Example 6.5 (Plot)

In the circuit, use v_S=2/5V, R=12Ω, L=2H and C=1/50F. Find v(t) and i(t), and plot them for time 0<t<1.5 seconds.




```
type
s\only("vc,il"):s\tr("e,1,0,2/5:r,1,2,12:l,2,3,2,0:c,3,0,1/50,0")
```

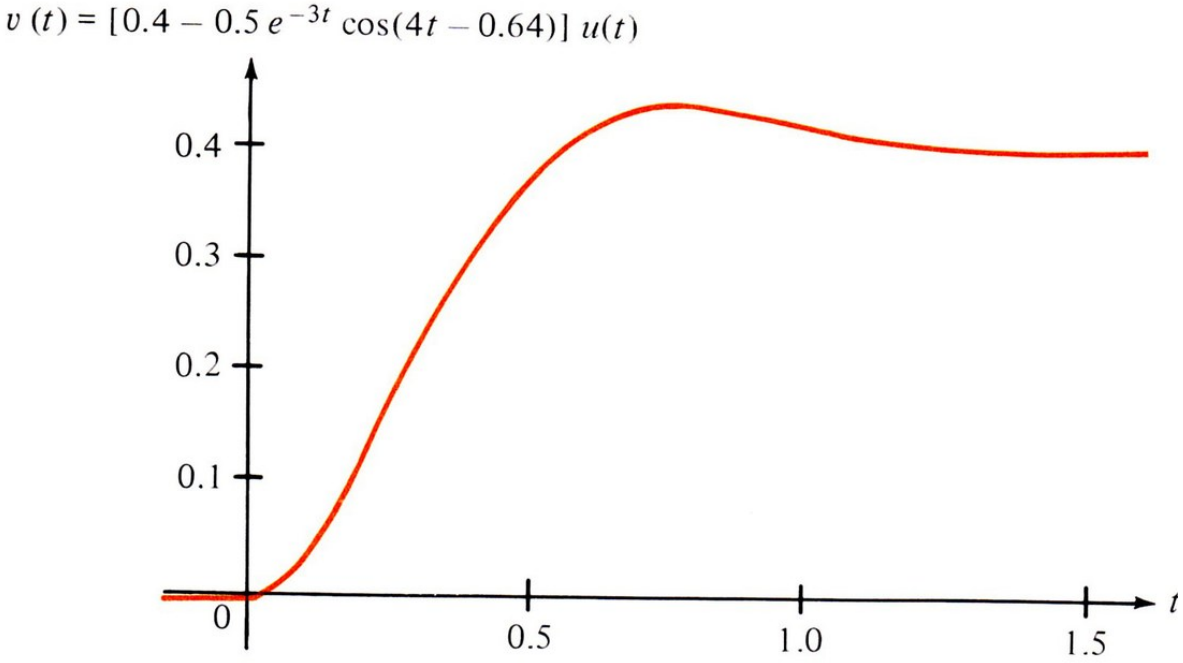
```
type
vc
```

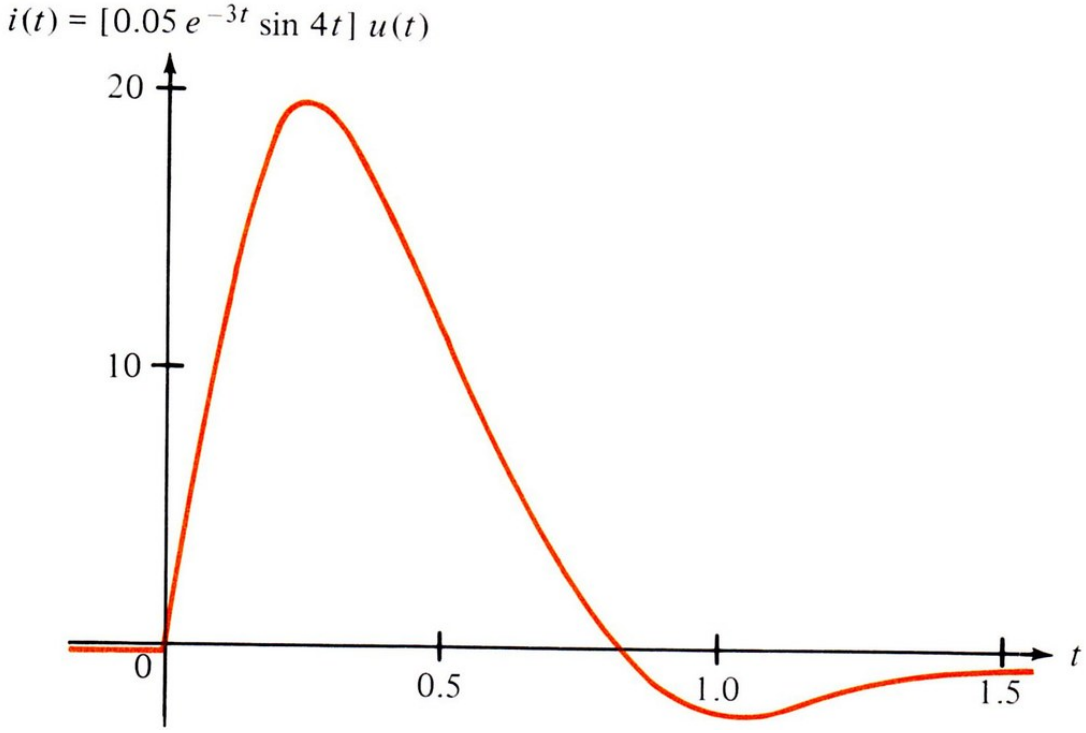
$-(2/5)e^{-3t}\cos(4t)-(3/10)e^{-3t}\sin(4t)+2/5$

```
type
il
```

$$(1/20)e^{-3t}\sin(4t)$$

These are the right answers. To plot them, run the `plot` tool, thus: s\plot(). Once the plot window opens, enter `vc` as function, `0` as minimal time and `1.5` as maximal time. Compare to the graph from the book. Repeat the plot for `il`, and compare that one to the book's graph too.





Bo2's Drill Exercise 6.7

For the circuit in **Bo2's Example 6.5 (page 154)**, use v_S=3V, R=5Ω, L=1/2H and C=1/8F. Find v(t) and i(t).

type
```
s\only("vc,il"):s\tr("e,1,0,3:r,1,2,5:l,2,3,1/2,0:c,3,0,1/8,0"):{vc,il}
```

$$\{-4e^{-2t}+e^{-8t}+3, e^{-2t}-e^{-8t}\}$$

Bo2's Drill Exercise 6.8 (Ramp)

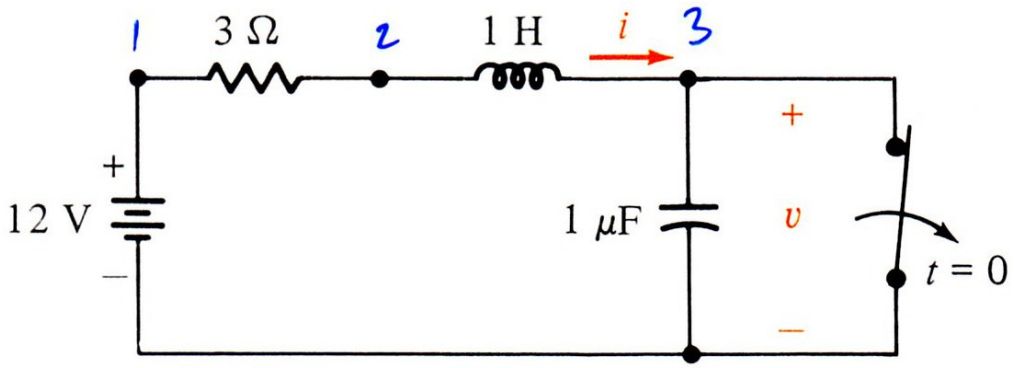
For the circuit in **Bo2's Drill Exercise 6.7 (page 156)**, find the voltage drop in the capacitor v(t) if the source has a value v_S(t) = 3 r(t).

type
```
s\only("vc"):s\tr("e,1,0,3t:r,1,2,5:l,2,3,1/2,0:c,3,0,1/8,0"):vc
```

$$2e^{-2t}-(e^{-8t})/8+3t-15/8$$

Bo2's Example 6.6 (Plot)

For the circuit, plot the voltage drop in the inductor for 0 < t < 5ms.



First, find the initial conditions for t<0 by running a DC simulation.

type
```
s\dc("e,1,0,12:r,1,2,3:l,2,3,1:c,3,0,1'μ:s,3,0"):{il,vc}
```

returns
```
{4,0}
```

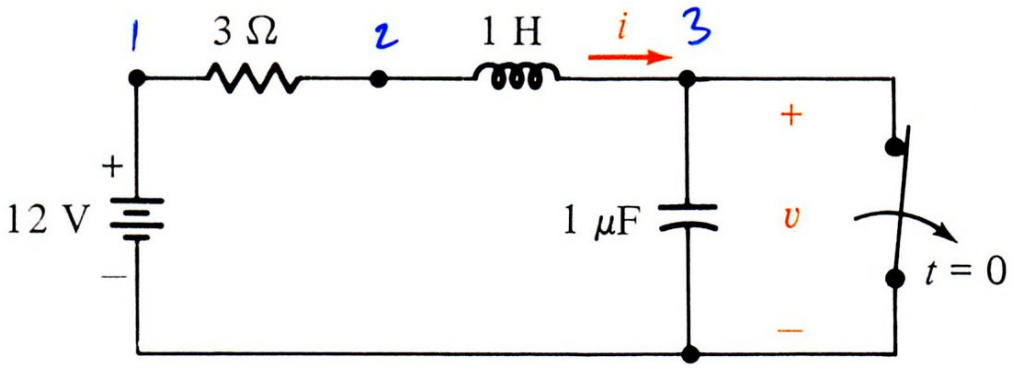
Then, find the `v1` for t≥0 by running a TR simulation.

type
```
s\only("vl"):s\tr("e,1,0,12.:r,1,2,3:l,2,3,1,4:c,3,0,1'μ,0")
```

Finally, run `s\plot()` and enter `vl` as function, `0` as minimal time and `0.005` as maximal time. In the resulting plot you will see something outstanding: around t = 1.57 ms, the voltage drop across the inductor is 3991 volts!

Bo2's Drill Exercise 6.9

For the circuit, R=1Ω, L=2H, C=1/2F and i_S(t)=u(t). Find i(t) and v(t).

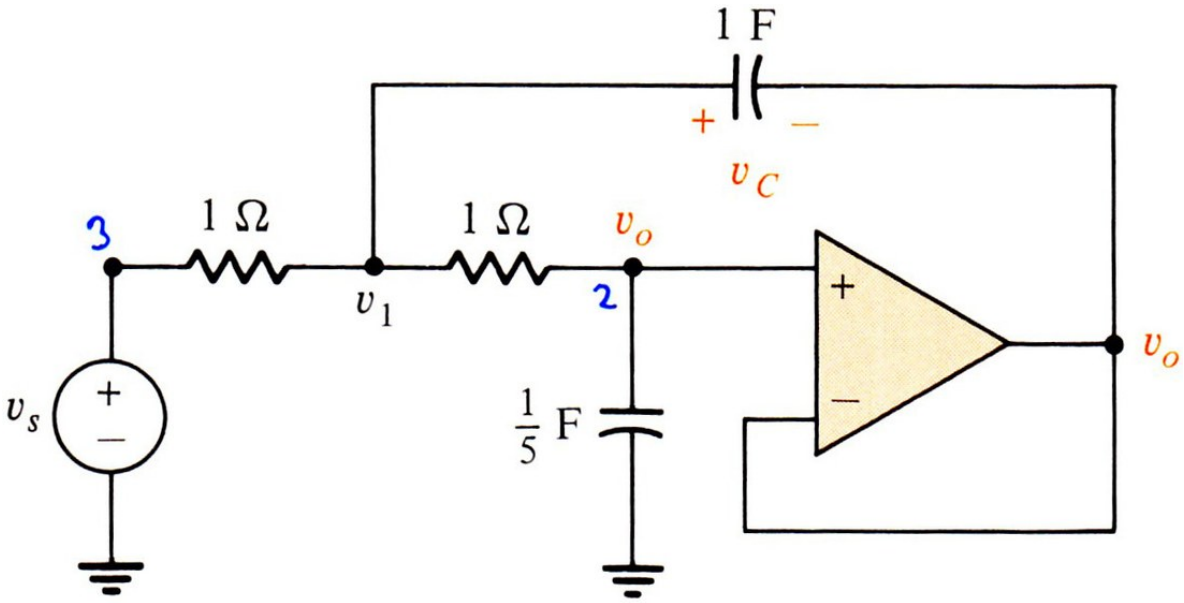


type
```
s\only("il,vc"):s\tr("j,0,1,1:r,1,0,1:l,1,0,2,0:c,1,0,1/2,0"):{il,vc}
```

$$\{(-t-1)e^{-t}+1, 2te^{-t}\}$$

Bo2's Figure 6.23 (Op Amp)

For the circuit, find v_o(t) if v_S(t) = u(t).



type
```
s\only("vo"):s\tr("e,3,0,1:r1,3,1,1:r2,1,2,1:
ca,2,0,1/5,0:cb,1,o,1,0:o,2,o,o"):vo
```

$$1 - e^{-t}\cos(2t) - (1/2)e^{-t}\sin(2t)$$

Bo2's Drill Exercise 6.10 (Op Amp)

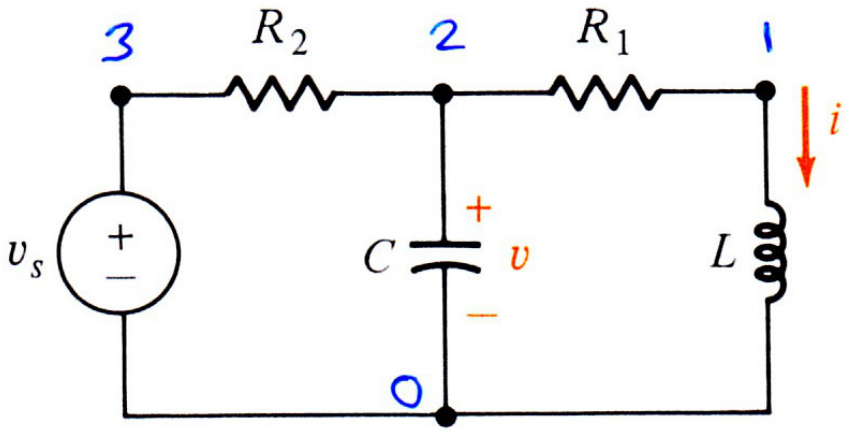
In the previous circuit, change the capacitor's value from 1/5 F to 25/16 F. Find v_o(t) if v_S(t) = 3 u(t).

type
```
s\only("vo"):s\tr("e,3,0,3:r1,3,1,1:r2,1,2,1:
ca,2,0,25/16,0:cb,1,o,1,0:o,2,o,o"):vo
```

$-4e^{-2t/5}+e^{-8t/5}+3$

Bo2's Drill Exercise 6.11 (p307)

For the circuit, suppose that $R_1 = R_2 = 1\Omega$, $L = 1$H, $C = 1$F and v_S(t)= $2e^{-2t}$ u(t). Find i(t) if all initial conditions are zero.



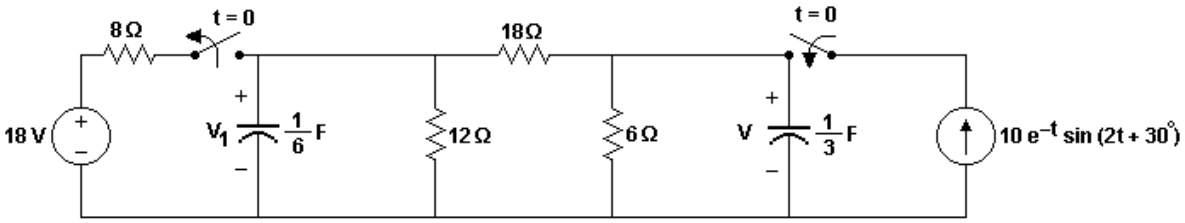
type
```
s\only("il"):s\tr("e,3,0,2e^(-2t):r2,3,2,1:r1,2,1,1:c,2,0,1,0:l,1,0,1,0"):
il
```

$-(e^{t}\cos(t)-e^{t}\sin(t)-1)\,e^{-2t}$

A more complex problem

The circuit comes from the circuits analysis class of Professor Eliane Boulet de Cabrera, from Universidad Tecnologica de Panama. Find v_1 and v.

Because of the switches, this is a two interval problem. The first is run in DC.



type
```
s\dc("e,3,0,18:r8,3,1,8:ca,1,0,1/6:r12,1,0,12:
r18,1,2,18:r6,2,0,6:cb,2,0,1/3"):{vca,vcb}
```

returns
```
{9,9/4}
```

These serve as initial conditions for the second interval, which is run in TR. Since we only want v1 and v2, it pays off to let Symbulator know this:

type
```
s\only("v1,v2"):
```

type
```
s\tr("ca,1,0,1/6,9:r12,1,0,12:r18,1,2,18:r6,2,0,6:
cb,2,0,1/3,9/4:j,0,2,10e^(-t)sin(2t+30°)")
```

Be patient. This took 78 seconds in my calculator, including over half a minute just to find the inverse Laplace of the two desired answers.

$$
\begin{aligned}
&(5\sqrt{3}/17 - 20/17)e^{-t}\cos(2t) \\
&+ (-20\sqrt{3}/17 - 5/17)e^{-t}\sin(2t) \\
&+ (80\sqrt{3}/17 + 193/34)e^{-t/2} + (9/2 - 5\sqrt{3})e^{-t}
\end{aligned}
$$

$$
\begin{aligned}
&(-245\sqrt{3}/34 - 20/17)e^{-t}\cos(2t) \\
&+ (245/34 - 20\sqrt{3}/17)e^{-t}\sin(2t) \\
&+ (80\sqrt{3}/17 + 193/34)e^{-t/2} + (5\sqrt{3}/2 - 9/4)e^{-t}
\end{aligned}
$$

type
```
v1
```

type
```
v2
```

These are the correct answers, which explains why Professor Boulet – who made the problem up - is a living legend among circuit students at UTP.

Advanced use of Expert in TR

The **expert** tool can be very useful in transient analysis. Using it, however, requires some knowledge. Here's two things you need to know in order to use `ex` like a boss:

First, when Symbulator solves a problem using TR, it follows these general steps:

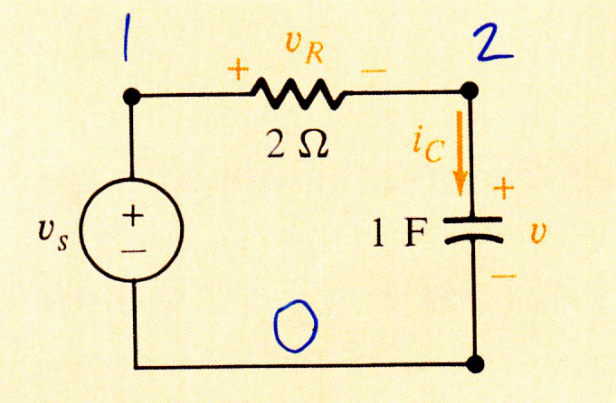
- Generate a set of equations and unknowns for the system, in the frequency domain. Any time-dependent source is replaced with a dummy variable.
- Solve these frequency domain equations.
- Replace any dummy variable with the original source value, and convert the answers to the time domain.

Second, when Symbulator solves a problem using the expert tool, it freezes this process halfway between steps 1 and 2, so that you can tinker with the equations and unknowns before they are solved.

In the problems below, you will see that, when we tinker with the equations, we have to do so in the frequency domain.

Bo2's Drill Exercise 4.5 (Expert)

For this circuit, we know that the capacitor's initial condition is zero, that v_s is an unknown step source (i.e. of the form $A\ u(t)$, where A is a constant value in volts) and that the voltage drop in the capacitor for t>0 is found to be 1-e^(-t/2). Find the voltage drop in the resistor, the current through the capacitor and the value of the source.



This problem is a match made in heaven for the **expert** tool, because we have one unknown value in the circuit (i.e. the value of the step source) and we have one known answer (i.e. the voltage drop in the capacitor.) So, the game plan here is to run this circuit through Symbulator's expert mode, add one new equation and one new unknown, and then solve. First, let's generate the new equation.

A non-expert user would think that the new equation is 1-e^(-t/2)=vc. But you know better. You know that Symbulator solves the equations of TR problems in the frequency domain. So we have to convert this expression from the time domain to the frequency domain. Symbulator has a shortcut to invoke DiffEq's Laplace Transform to do so: `s\t2s`.

```
type
s\t2s(1-e^(-t/2))=vc
```

We have our new equation. Copy this equation into the clipboard, since we will want to paste it in the Expert window. Now let's run the Expert simulation of the circuit. Let's define the value of the source as `a*u(t)`, since we know it's a step source; the variable `a` will serve as the unknown value, for which we will solve in the Expert mode.

```
type
s\ex("e,1,0,a*u(t):r,1,2,2:c,2,0,1,0")
```

When prompted, select TR as the desired option. Then you will see the typical Expert prompt. In the equations field, paste the new equation, adding the word `and` first:

```
type
and 1/s-2/(2*s+1)=vc
```

In the unknown field, add the variable a, preceded by a comma:

```
type
,a
```

Let Symbulator solve the equations now. Once it is done, we want to do a quick sanity check. If the system solved correctly, we should see that Symbulator has as voltage drop in the capacitor the expression we already know from the problem statement:

```
type
vc
```

$1-e^{-t/2}$

This is correct, so we proceed to ask for the answers we want:

```
type
ic
```

$(1/2)e^{-t/2}$

```
type
vr
```

$e^{-t/2}$

```
type
a
```

These are the correct answers. The value of the step source is 1 volt.

For the circuit shown in Fig. DE4.13, the voltage across the 1-Ω resistor is $v_R(t) = e^{-t}$ V for $t \geq 0$. Given that $v_C(0) = 0$ V, find the following functions for $t \geq 0$: (a) $i_R(t)$; (b) $v_C(t)$; and (c) $v_s(t)$.
Answer: (a) e^{-t} A; (b) $1 - e^{-t}$ V; (c) 1 V

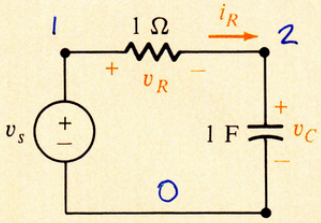


Fig. DE4.13

Bo2's Drill Exercise 4.13 (Expert)

This problem is solved just like the one we already saw. First, find the new equation:

type
```
vr=s\t2s(e^(-t))
```

Then run the Expert simulation. I decided to use `vs` as the step value of the source:

type
```
s\ex("e,1,0,vs*u(t):r,1,2,1:c,2,0,1,0"):
```

When prompted, select TR. In the equations field, add the new equation you found:

type
```
and vr=1/(s+1)
```

In the unknowns field, add the new variable:

type
```
,vs
```

Let Symbulator rip. Once it finishes solving, ask for the sanity check:

type
```
vr
```

e^{-t}

Looks good, so go ahead and ask for the answers:

type
```
ir
```

e^{-t}

type
```
vc
```

$1-e^{-t}$

type
```
vs
```

Ain't that a beauty? Yes, ma'am! Atta boy, Symbulator! Atta boy!

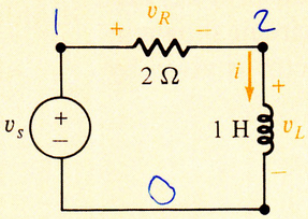
Bo2's 164 (Expert)

For the circuit shown in Fig. DE4.2, the current through the inductor is

$$i(t) = \begin{cases} 0 & \text{for } -\infty < t < 0 \\ 1 - e^{-2t} & \text{for } 0 \le t < \infty \end{cases}$$

Find, for $t \ge 0$, (a) $v_L(t)$, (b) $v_R(t)$, (c) $v_s(t)$, and (d) the power absorbed by the inductor.

Answer: (a) $2e^{-2t}$ V; (b) $2(1 - e^{-2t})$ V; (c) 2 V; (d) $2(e^{-2t} - e^{-4t})$ W



This is the same thing as the previous two, so there is no need for commentary.

type
```
il=s\t2s(1-e^(-2*t))
```

type
```
s\ex("e,1,0,vs*u(t):r,1,2,2:l,2,0,1,0")
```

Choose TR, and add `,vs` to the unknowns and the following to the equations

type
```
and il=1/s-1/(s+2)
```

And solve. When done, ask for the sanity check:

type
```
il
```

$1-e^{-2t}$

Since that adds up, ask for the answers:

type
`vl`

$2e^{-2t}$

type
`vr`

$2-2e^{-2t}$

type
`vs`

2

type
`vl*il`

$2e^{-4t}(e^{2t}-1)$

This is the end of the TR section.

LESSON 7

Alternating current analysis

Learn how to run an *alternating current* analysis (via phasor analysis) using **ac**. Learn how to describe capacitors, inductors, *impedances and admittances*, and sources for AC analysis.

In this lesson you will learn how to use the **ac** program to solve AC circuits in Symbulator. You will also learn how to describe the elements you already know so they suit AC analysis, and how to use **er** and **th** in their AC mode.

7.1 AC analysis in Symbulator

The main thing to know is that in AC mode Symbulator understands phasors — complex numbers. It uses them in the input, in the analysis and in the output.

> CAUTION
>
> **Symbulator does not understand sinusoids**
>
> You should not feed Symbulator any sinusoids, or expect any sinusoid answers from it. Any conversion to and from sinusoids has to be done by you.

The ac program

Symbulator has a program for analysing AC circuits, called **s\ac**. Unlike s\dc and s\tr, which take only the circuit description, s\ac takes two inputs: the description, in a string, and the frequency of the circuit in radians per second.

The AC mode

Besides s\ac itself, you can use s\er, s\th and s\ex in their AC mode: select AC when asked what type of analysis you want. This lesson has examples of all of them.

Describing elements for AC

When describing a circuit for AC analysis, element values have to be written the way this analysis expects. Valueless elements, like the short circuit and the op amp, are unchanged. Here's how to describe the ones with values:

- Current sources **j** and voltage sources **e** accept complex values. You can declare their value in rectangular form, e.g. `10-3i`, or in angular form, e.g. `(100∠120°)`.
- The element **r** no longer represents just a resistor: it now represents an **impedance**.

If you are wondering why we didn't use the letter z, wait until we get to two-ports. In AC analysis, r accepts complex numbers as its value, such as `10-3i`. We use r to describe resistors (real values) and conductances (real values, inverted), but also impedances (complex values) and admittances (the inverse of their complex value).

- The elements **c** and **l** are used to describe capacitors and inductors *only* when their values are given in farads and henries, respectively. For AC analysis their description no longer requires the fifth field for the initial condition, since alternating current analysis focuses on the steady state after transient effects are long gone. So we only use the name, the two nodes, and the value.

> ⚠ WARNING
>
> **If it's in Ω, it's an r**
>
> When the problem gives us the values of capacitors and inductors in ohms, we **must** describe them as impedances. If a capacitor has its value in farads, describe it with **c**. If an inductor has its value in henries, describe it with **l**. But anything that has its value in ohms — resistor, impedance, capacitor or inductor — must be described using **r**.

The values of all of these elements can use SI prefixes.

What answers do you get

You get the same answers you did in DC, except that now they are phasors. The one difference is that now, besides the real power consumed, you also get the complex power consumed. We will discuss this in Lesson 8 (page 179).

Rectangular or polar

A phasor can be written two ways, and circuit problems use both: in *rectangular* form as `3 + 4j`, or in *polar* form as an amplitude and an angle, $5\angle 53.13°$. They are the same number. Which one you want depends on the question — rectangular adds and subtracts easily, polar multiplies and divides easily, and textbook answers are usually quoted in polar.

Symbulator answers in rectangular form, and the **aa** tool converts one answer at a time. You will see it used throughout this lesson.

7.2 Solved numerical examples

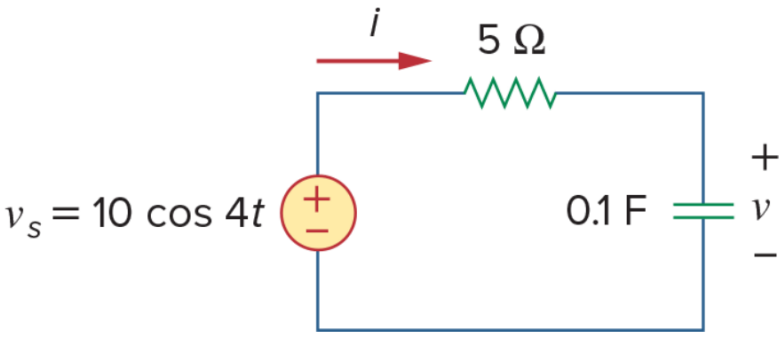
With values in F and H

AS7's Example 9.9

Find $v(t)$ and $i(t)$.

SOLUTION

Since this is the first AC problem we will solve, I will explain every step, including the



AS7's Example 9.9

manual ones, in full detail.

Step 1: sinusoids to phasors. We have to express the value of the source in terms Symbulator can understand, so we convert it from a sinusoid to a phasor: $10\angle 0°$. We can pass this as `(10∠0°)` or, since the angle is zero, simply as `10`.

Step 2: note the frequency. The circuit has a capacitor with a value given in farads. Because of this, Symbulator will need the frequency, given by the problem as 4 rad/s, to convert that value into an impedance in ohms.

Step 3: describe the circuit.

```
type
"e,1,0,10:r1,1,2,5:c,2,0,.1"→cir
s\ac(cir,4)
```

Notice that, in our circuit description, the order we give to the nodes of the resistor and the capacitor is chosen to be convenient for the answers we will ask for.

Step 4: read the answers. Once Symbulator is done, you should take a look at what it found:

- the usual voltages in the nodes, voltage drops in the elements, and currents through the elements, in variables that should be familiar by now
- the *average power consumed* in the source and the resistor, in `ape` and `apr1`. None is given for the capacitor, since capacitors and inductors do not consume real power.
- the *complex power consumed* in all elements, in `sc`, `se` and `sr1`
- the *equivalent impedance* of the rest of the circuit as seen by the source, in `ze`

To get the answers we need for this problem in particular, we ask for `ir1` and `vc`. It is likely that the calculator will give you the answers in rectangular form, with a real part and an imaginary part.

If we want to see them as an amplitude and angle, we can use Symbulator's **aa** tool:

```
type
s\aa(ir1)
```

returns

```
1.789∠26.57°
```

This is correct. You manually convert it to a sinusoid, by putting it back in the same terms the input was given in:

$$i(t) = 1.789\cos(4t + 26.57°)\ \text{A}$$

Asking for the capacitor voltage the same way gives 4.472∠-63.43°, that is,

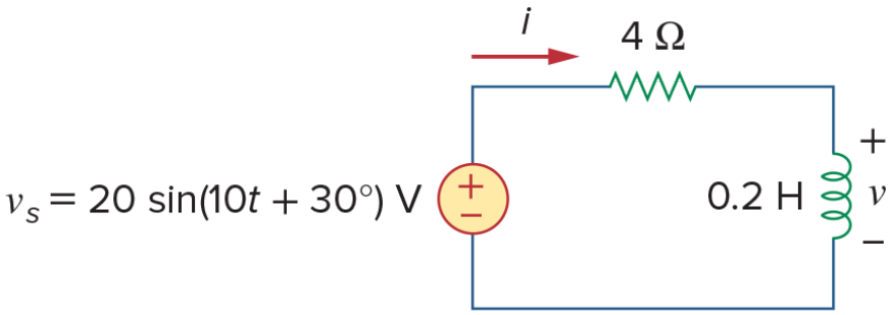
$$v(t) = 4.472\cos(4t - 63.43°)\ \text{V}$$

which is correct.

Future solved problems will not include this level of detail in the solution, only the circuit description and the commands we give.

AS7's Practice Problem 9.9

Determine ***v***(t) and ***i***(t).



AS7's Practice Problem 9.9

SOLUTION

type

```
"e,1,0,(20∠30°):r1,1,2,4:l,2,0,.2"→cir
s\ac(cir,10)
```

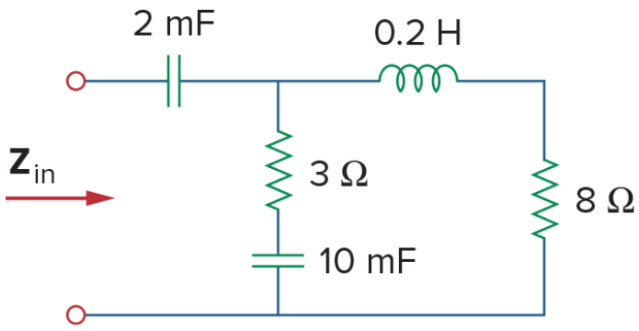
We ask for the current in the resistor and get 4.472∠3.43°, and for the voltage in the inductor and get 8.944∠93.43°. Both are correct.

AS7's Example 9.10

Find the input impedance of the circuit. Assume that the circuit operates at ω = 50 rad/s.

SOLUTION

Since this is a passive circuit, to reduce it we will use the **er** tool.



AS7's Example 9.10

type
```
"ca,1,2,2'm:r1,2,3,3:cb,3,0,10'm:l1,2,4,.2:r2,4,0,8"→cir
s\er(cir,1,0)
```

When asked what type of analysis, choose AC. Since there are capacitors in farads and an inductor in henries, Symbulator will ask you for the frequency; enter 50. Once the program is done, ask for the equivalent impedance, `zeq`.

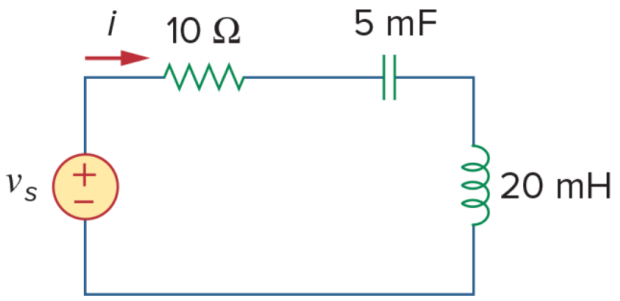
returns
```
3.22-i11.07
```

That is 3.22 – j11.07 Ω, which is correct.

> **TIP**
>
> **The pr tool only reduces impedances**
>
> In case you were tempted to reduce this circuit by hand instead of simulating it, remember that parallel reduction works on resistors and impedances only. You would need to convert the capacitors and the inductor to impedances in ohms before you do that.

AS7's Problem 9.35

Find the steady-state current i in the circuit when $v_s(t) = 50 \cos 200t$ V.



AS7's Problem 9.35

SOLUTION

The capacitor is in farads and the inductor in henries, so Symbulator needs the frequency, and the source is entered as its phasor: 50 cos 200t is `50` at an angular frequency

of 200.

type
```
"e,1,0,50:r1,1,2,10:c,2,3,5'm:l,3,0,20'm"→cir
s\ac(cir,200)
```

Choose AC. Symbulator asks for the frequency because of the farads and henries; enter 200. Then ask for `s\aa(ir1)`.

type
```
s\aa(ir1)
```

We get **4.789∠-16.7°** A.

That is 4.789 A at an angle of −16.7°, which is correct.

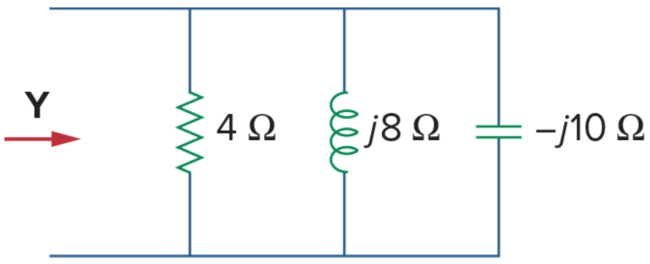
> TIP
> **Why the resistor is r on one calculator and r1 on the other**
> The version 7 description above names its resistor `r`; the version 8 one cannot, because the Nspire reserves `r` (and `rr`), as Lesson 1 (page 8) notes — `r1` is the usual fix there.

With values in Ω only

When the values of all the capacitors and inductors in the circuit are given in imaginary ohms, they must be entered in the circuit description as impedances, **r**. In these cases, Symbulator will not need a frequency.

AS7's Problem 9.37

Determine the admittance Y for the circuit.



AS7's Problem 9.37

SOLUTION

Since the values are in ohms, these are all impedances. This is a passive circuit, so we could use the er tool. However, the structure of this circuit is so simple that we can reduce it by hand. The problem asks for the equivalent admittance, so our answer will be the inverse of the equivalent impedance:

type
```
1/(s\pr({4,i8,-i10}))
```

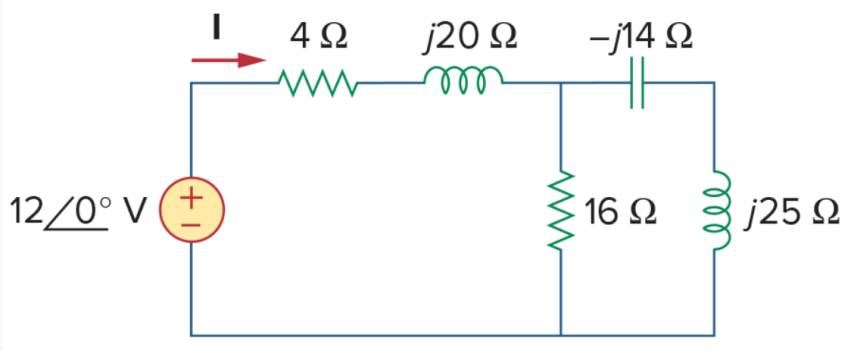
returns

```
0.25-0.025i
```

That is 0.25 – j0.025 S, which is correct.

AS7's Problem 9.39

For the circuit shown, find the equivalent impedance, and use that to find the current I. Let omega = 10 rad/s.



AS7's Problem 9.39

SOLUTION

The frequency the question gives is superfluous: every value is already in ohms. Nothing here needs a full simulation either; the equivalent impedance is one expression, using the parallel shorthand:

type

```
4+i20+s\pr({16,-i14+i25})→zeq
```

returns

```
9.135+i27.47
```

That is 9.135 + j27.47 Ω. The current is the source voltage divided by it:

type

```
s\aa(12/zeq)
```

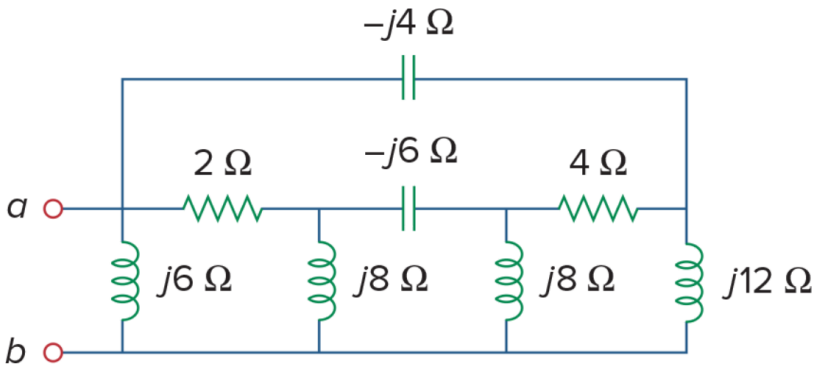
We get **414.5∠-71.6°** mA.

That is 414.5 mA at an angle of –71.6°, which is correct.

> **TIP**
>
> **The same answer from a simulation**
>
> You can also let Symbulator do the reduction, using the square-bracket shorthand for parallel elements inside a value:
> AC asks for a frequency anyway; put in anything, since every value is already an impedance. Then `aa(ir1)` gives the same **0.4145∠-71.60°** A.

AS7's Problem 9.73

Determine the equivalent impedance for the circuit.



AS7's Problem 9.73

SOLUTION

Eight impedances and no source: a job for *Find equivalent* rather than a solve. The names carry the two nodes each element bridges, a convenience only.

type
```
"r10,1,0,i6:r20,2,0,i8:r30,3,0,i8:r40,4,0,i12:r12,1,2,2:r23,2,3,-i6:r34,3,
4,4:r14,1,4,-i4"→cir
s\er(cir,1,0)
```

returns
```
0.3794+i1.46
```

That is 0.3794 + j1.46 Ω, which is correct.

With dependent sources

A dependent source is described exactly like an independent one — the value is just an expression naming another element's answer instead of a number.

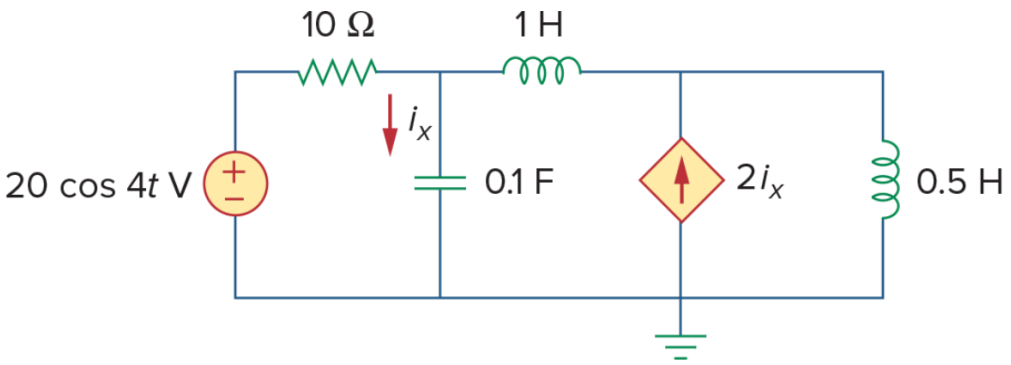
Version 7 runs the quantity and the element together, as in `2icx`.

AS7's Example 10.1

Find i_x in the circuit.

SOLUTION

The current source is controlled by the current through the capacitor **cx**, which is what the last field of **j1** says.



AS7's Example 10.1

```
type
"e1,1,0,20:r1,1,2,10:cx,2,0,.1:l1,2,3,1:j1,0,3,2icx:l2,3,0,.5"→cir
s\ac(cir,4)
```

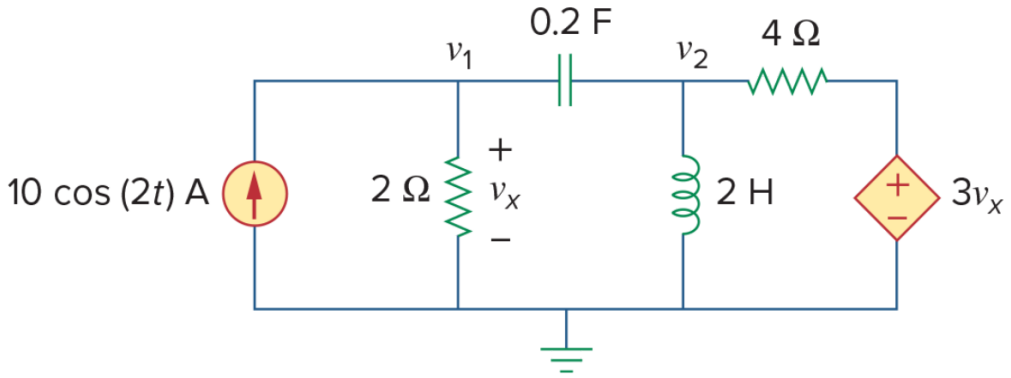
```
type
s\aa(icx)
```

We get **7.59∠108.4°** A.

That is 7.59 A at an angle of 108.4°, which is correct.

AS7's Practice Problem 10.1

Find v_1 and v_2 in the circuit.



AS7's Practice Problem 10.1

SOLUTION

This one is controlled by a *voltage* rather than a current — three times the voltage across **rx**.

```
type
"j,0,1,10:rx,1,0,2:c,1,2,.2:l,2,0,2:r1,2,3,4:e,3,0,3vrx"→cir
s\ac(cir,2)
```

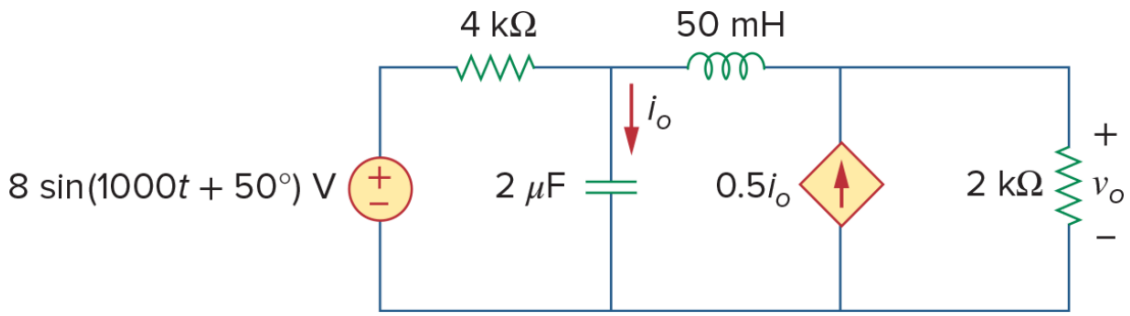
type
```
{s\aa(v1),s\aa(v2)}
```

We get **11.33∠60.02°** V and **33.02∠57.13°** V.

Both are correct.

AS7's Example 10.13

Obtain v_o and i_o in the circuit.



AS7's Example 10.13

SOLUTION

The book wants its answers in terms of cosine rather than sine, so the source is taken as 8 cos(1000t − 40°) — as a phasor, `(8∠-40°)`.

Note the decimal points on the values. They make the arithmetic approximate, which here is what you want: an exact solve of this circuit carries surds through every step for no benefit.

type
```
"e1,1,0,(8.∠-40°):r1,1,2,4.'k:co,2,0,2.'μ:l1,2,3,50.'m:j1,0,3,.5ico:ro,3,
0,2.'k"→cir
s\ac(cir,1000.)
```

type
```
{s\aa(vro),s\aa(ico)}
```

We get **1.55∠-95.18°** V and **3.26∠-3.74°** mA.

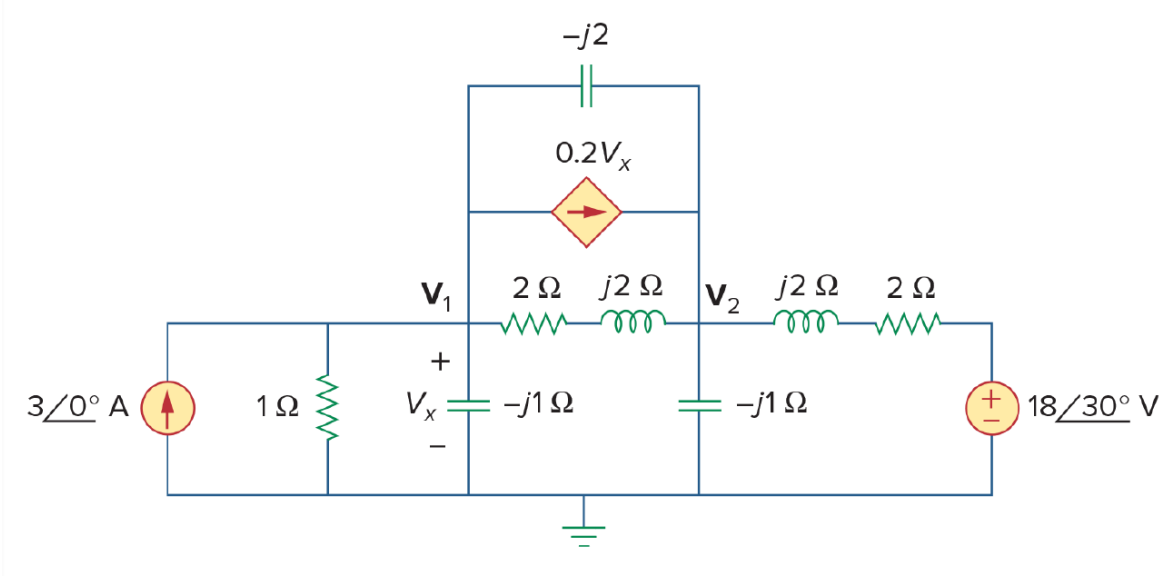
That is 1.55 V at −95.18° and 3.26 mA at −3.74°, both correct.

AS7's Example 10.14

Find V_1 and V_2 in the circuit.

SOLUTION

Every value here is already in ohms, so no frequency matters. Again the decimal points keep the arithmetic approximate.



AS7's Example 10.14

type
```
"j1,0,1,3.:r1,1,0,1.:rx,1,0,-i1.:r3,1,2,-i2:j2,1,2,.2vrx:r4,1,2,2.+i2.:r5,
2,0,-i1.:r6,2,3,2.+i2.:e1,3,0,(18.∠30°)"→cir
s\ac(cir,ω)
```

type
```
{s\aa(v1),s\aa(v2)}
```

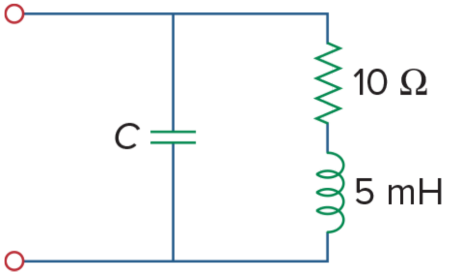
We get **2.708∠-56.73°** V and **6.914∠-80.70°** V.

That is 2.708 V at −56.73° and 6.914 V at −80.70°, both correct.

7.3 Solved numerical-from-symbolic examples

AS7's Problem 9.89

Calculate the value of C so that the net impedance is purely resistive at 2 kHz.



AS7's Problem 9.89

SOLUTION

The answer given by the book is 25 μF. But I think this is one of those unfortunately common instances where there is a mistake in the book. I believe they meant to ask "at

2k rad/s", because that is the frequency at which the answer they give is right. Let me tell you how I solved this problem and I will let you be the judge.

```
type
"c,1,0,c:r1,1,2,10:l,2,0,5'm"→cir
s\er(cir,1,0)
```

When asked, specify AC. Then we are asked for the frequency. The problem gives 2000 Hz; Symbulator asks for radians per second, so I enter `2π2E3`. I could also have entered 12566.37.

Notice two things. First, I did not use an SI prefix: the shorthand only works in the circuit description string. Second, I used 2E3 rather than 2000 because I wanted an approximate value — with exact values, cSolve took too long.

> ⚠ WARNING
>
> **If exact fails in AC, try approximate**
>
> It is a known bug of the TI CAS that `cSolve` often struggles with some exact equations. You may see that it takes forever to solve, or the calculator runs out of memory. In those cases, try again using approximate values — add a decimal point to the end of the values, or use engineering notation — or with symbolic values. More often than not, this bypasses the bug and you get a solution surprisingly fast.

Once the simulation finishes, solve for the value of c that makes the imaginary part of the equivalent impedance zero:

```
type
solve(imag(zeq)=0,c)
```

The answer is **c** = **0.000001235**, or 1.235 μF. This is not the answer the book gives.

I first used cSolve here, which also works, but my friend Qifan Wang — who verified my answers — pointed out, correctly, that it is not necessary: focusing on the imaginary part removes every reference to the complex operator, so we are solving an equation in real terms only.

Let's now repeat the process, using what I suspect is the frequency they meant: 2000 rad/s. I also want to show you something cool. Now that we know the answer is in the range of μF, we can declare the value of the capacitor as `c'μ`, so that the value of c we get will be in that scale:

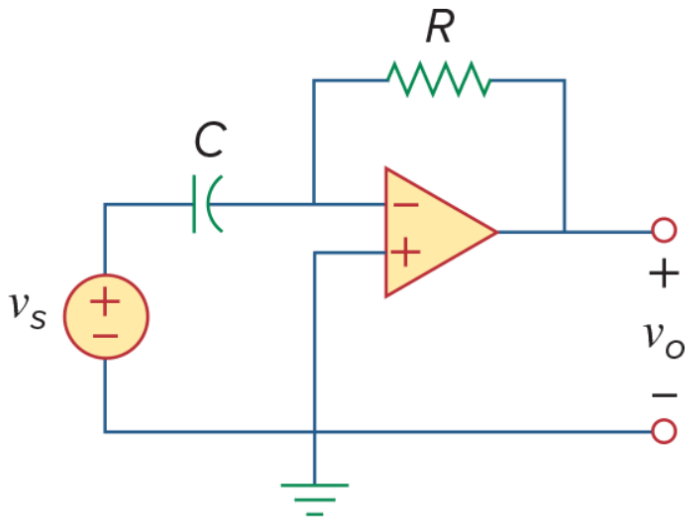
```
type
"c,1,0,c'μ:r1,1,2,10:l,2,0,5'm"→cir
s\er(cir,1,0)
solve(imag(zeq)=0,c)
```

The answer is **c** = **25**, that is to say 25 μF. This is the right answer in the book. So they asked the wrong question — they asked it using the wrong frequency unit.

7.4 Solved symbolic examples

AS7's Problem 10.69

Find V_o/V_s.



AS7's Problem 10.69

SOLUTION

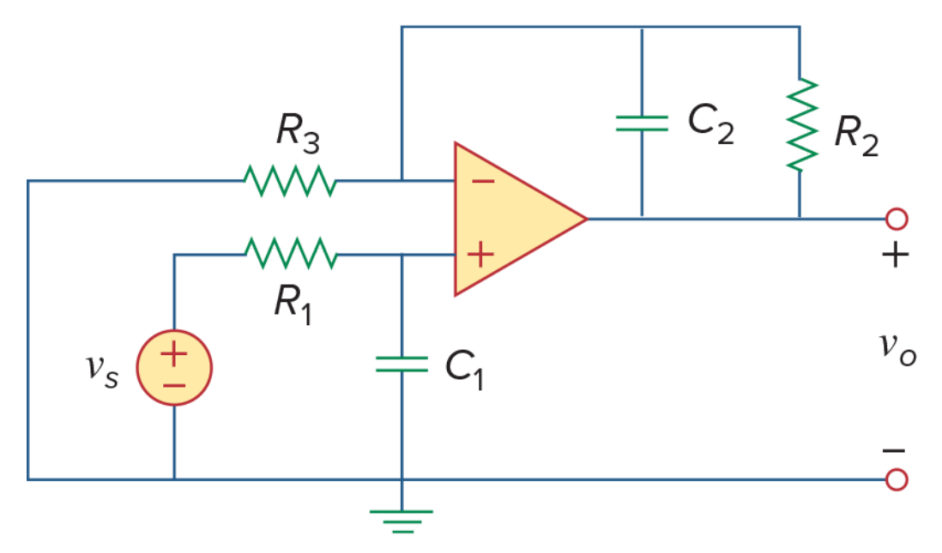
type
```
"e,1,0,vs:c,1,2,c:r1,2,o,r:o,0,2,o"→cir
s\ac(cir,ω)
```

We ask for `vo/vs` and get $-c\cdot\omega\cdot r\cdot\mathbf{i}$, which is correct.

> **NOTE**
> **Know Symbulator's limits**
> Symbulator is a calculator program. It runs on limited hardware, supported by a limited CAS. It can solve most of the problems you will find in the textbooks used in Circuits I and II courses, and it particularly shines when solving symbolic problems. However, as you move to larger and more complex problems, it will struggle. In a sense this is not a problem. First, solving large circuits symbolically makes no sense: the expressions are beyond our understanding. Second, there are already many established tools for solving them numerically.
> What matters is that you know the limits, so that they don't take you by surprise in an exam. Think of Symbulator as a wonderful Swiss knife of circuit simulators: no matter how good it is, it is still a pocket knife. You would not try to chop a tree down with it. SPICE and its children — LTspice, PSpice, MultiSim — are the chainsaws of circuit simulation. They won't give you symbolic answers, but when you need to chop a tree down, they are the way to go.

AS7's Problem 10.77

Compute the closed-loop gain V_o/V_s for the op-amp circuit.



AS7's Problem 10.77

SOLUTION

Here is an example that takes Symbulator to the limit of what it can solve symbolically — everything in it is a symbol, including the frequency.

type

```
"e,1,0,vs:r1,1,2,r1:ca,2,0,ca:r3,0,3,r3:o,2,3,o:cb,3,o,cb:r2,3,o,
r2"→cir:s\ac(cir,ω)
```

Notice `ca` and `cb`: `c1` and `c2` are reserved variables on the calculator.
Then ask for the ratio:

type

```
vo/vs
```

$$\frac{V_o}{V_s} = \frac{j\,c_b r_2 r_3 \omega + r_2 + r_3}{r_3\left(-c_a c_b r_1 r_2 \omega^2 + j\,c_a r_1 \omega + j\,c_b r_2 \omega + 1\right)}$$

It looks different from the answer in the book, but evaluating both shows they are the same expression arranged differently.

LESSON 8

Power in AC circuits

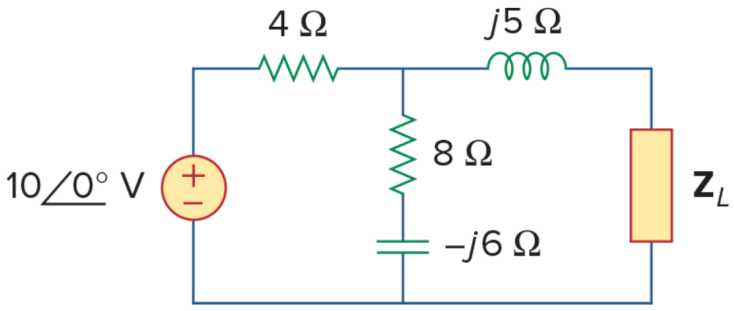
Learn to solve *average power* and *complex power* problems in AC. Learn how you can use RMS values in Symbulator with the **rms** flag. Solve power factor problems using the **pf** tool.

In this lesson you will learn how to run more advanced *alternating current* simulations. You will learn how to solve *average power* problems, when and how to use the **rms** flag, and how to solve *complex power* problems.

8.1 Average power

AS7's Example 11.5

Determine the load impedance ZL that maximises the average power drawn from the circuit. What is the maximum average power?



AS7's Example 11.5

SOLUTION

Symbulator's **th** tool helps us find all the answers we need. Notice this circuit description does not include the load, because it is not needed as input to **th**.

```
type
"e,1,0,10:r1,1,2,4:r2,2,0,8-6i:r3,2,3,5i"→cir
s\th(cir,3,0)
```

When asked a type of analysis, specify AC. Then you will be asked if a load problem is next. You don't have to answer Yes, but I invite you to, because I want to show you something cool.

The **s\th** tool saves the equivalent impedance in `zeq`, which evaluates approximately to

2.933 + j4.467. The load that will deliver the maximum power is its conjugate, 2.933 – j4.467.

The average maximum power delivered by the circuit is in `apmax`, which evaluates to 2.3674 W.

Now, here's the cool part: when you answer Yes to the load question, Symbulator saves expressions for the current through, voltage drop in, average power and complex power in a load connected to the equivalent circuit, as a function of the complex value of that load — the variable `load_`, where the underscore tells the calculator to treat this variable as complex. This means you can verify that this is the maximum power delivered:

```
type
aprl|load_=conj(zeq)
```

It evaluates to the same answer.

```
returns
2.36742424242424
```

It is the same answer.

8.2 RMS and PF

In the previous lesson we mentioned that after an AC simulation you also get the complex power consumed. How to read these values depends on a flag called the RMS flag.

The userms flag

You will learn in your circuits course that phasor analysis can be done in RMS values. There are advantages to it, and you will have to solve AC circuits both in normal values (also called magnitude values) and in RMS. Working in RMS simply means every current and voltage in the analysis is taken to be RMS. That's it.

The flag that tells the program which you want is stored in a variable called **userms**, and it can be true or false. By default, it is *false*.

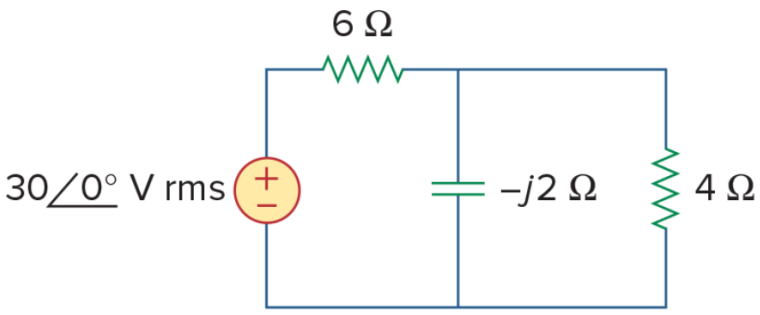
When the flag is false, AC analysis assumes normal values: Symbulator assumes all the currents and voltages you give it are magnitude values, gives you all the answers in magnitude values, and stores the average real power consumed in r, e, j and o elements in a variable called **ap** plus the element name.

When the flag is true, AC analysis assumes RMS values throughout, and the average real power consumed lands in a variable called **p** plus the element name.

AS7's Example 11.10

Determine the power factor of the entire circuit as seen by the source. Calculate the average power delivered by the source.

SOLUTION



AS7's Example 11.10

The source value in this problem is given in RMS values, so we must say so.

```
type
true→userms
"e,1,0,30.:r1,1,2,6.:r2,2,0,-i2.:r3,2,0,4."→cir
s\ac(cir,ω)
```

Or, if you want to simplify the impedances, the resistors can be collapsed into a single parallel combination.

The average power *consumed* by the source is in `pe`. To get the average power *delivered*, we ask for the negative of it, which gives us 125.4 W.

The pf tool

To find the power factor of an element answer, we can use the **s\pf** tool. We can use this tool in two ways.

With a complex expression as input. One way is to give it a complex expression, with real and imaginary parts. It can contain symbolic and/or numerical values, exact or approximate; it can be a complex number in rectangular or angular form; or it can be a variable holding such an expression, such as the complex power consumed in an element:

```
type
s\pf(se)
```

The pf tool takes this complex value and calculates the power factor as the absolute value of the real part divided by the magnitude of the whole thing.

Now, this is not the whole picture: we need to know whether this power factor is leading or lagging.

With an element name as input. The second way is to give it a string with the name of a single element that was part of an s\ac simulation that just finished. The tool recognises three elements in this form: e, j and r:

```
type
s\pf("e")
```

returns

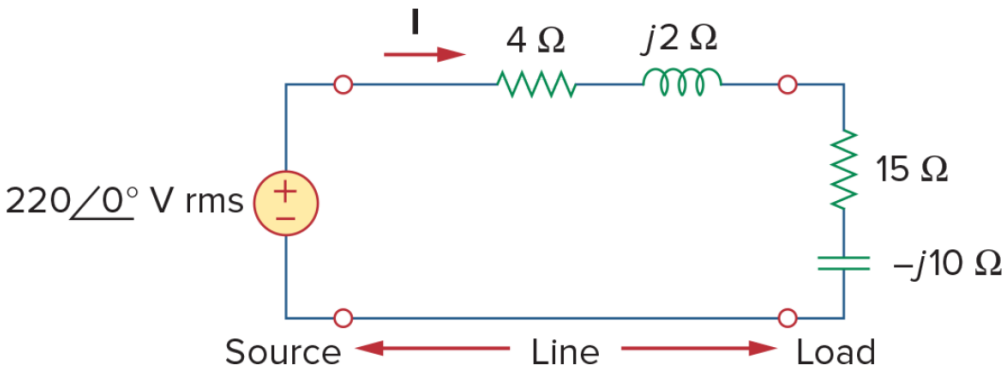
```
pf: 0.97342 leading
```

In a nutshell: given a complex number or expression, pf returns the power factor but says nothing about leading or lagging. Given the name of an element in a string, right after an s\ac simulation, it returns the value and says which. That second form works only if the expression can be evaluated numerically.

8.3 Complex power

AS7's Example 11.13

The figure shows a load being fed by a voltage source through a transmission line. The impedance of the line is represented by the (4+j2) Ω impedance and a return path. Find the real power and reactive power delivered by the source, and absorbed by the line and load.



AS7's Example 11.13

SOLUTION

The value of this circuit's source is given in RMS. Since the problem asks for answers regarding the line and the load, we should have separate impedances, one for each.

type

```
true→userms
"e1,1,0,220.:r1,1,2,4.+2.i:r2,2,0,15.−10.i"→cir
s\ac(cir,ω)
```

The complex power absorbed in the source, line and load are in `-se1`, `sr1` and `sr2`:

type

```
{-se1,sr1,sr2}
```

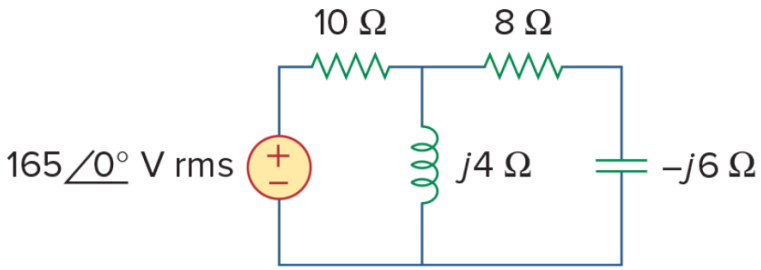
returns

```
{2163.8−911.1i, 455.5+227.8i, 1708.2−1138.8i}
```

These are correct.

AS7's Practice Problem 11.10

Calculate the power factor of the entire circuit as seen by the source. What is the average power supplied by the source?



AS7's Practice Problem 11.10

SOLUTION

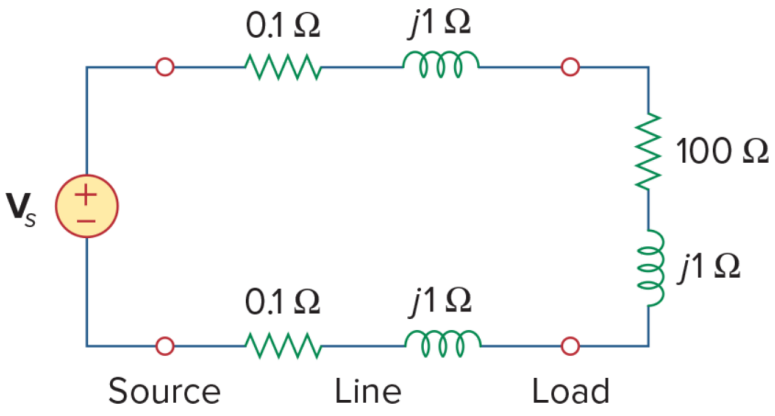
The values are RMS, so set `true→s\rms`.

```
type
true→userms:"e,1,0,165.:r,1,0,10.+[i4.,8.-6.i]"→cir:s\ac(cir,ω)
```

```
returns
-pe gives 2007.1 W; s\pf("e") gives pf: 0.93595 lagging
```

AS7's Problem 11.97

A power transmission system is modelled as shown. If V_s = 240 V rms, find the average power absorbed by the load.



AS7's Problem 11.97

SOLUTION

Three impedances in series: the outgoing line, the load, and the return path.

type

```
true→userms:"evs,1,0,240.:rl1,1,2,.1+i:rl,2,3,100+i:rl2,3,0,.1+i"→cir:
s\ac(cir,ω):prl
```

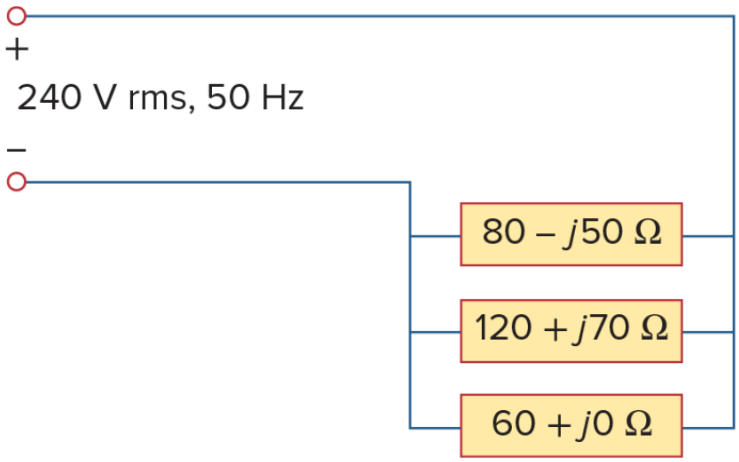
returns

```
573.2 W
```

Correct.

AS7's Problem 11.75

Consider the power system shown. Calculate the total complex power, the power factor, and the parallel capacitance needed for a unity power factor.



AS7's Problem 11.75

SOLUTION

Three loads in parallel across a 240 V rms source. The first two parts need no frequency at all, so leave omega as anything.

type

```
true→userms:"e,1,0,240:r1,1,0,80-i50:r2,1,0,120+i70:r3,1,0,60+i0"→cir:
s\ac(cir,ω)
```

(a) The complex power delivered is the opposite of the power the source consumes: `-se` gives **1835.9** – **114.7**j VA.

(b) The power factor, from `s\pf("e")`, is **0.99805** leading.

(c) This one needs a frequency, because it needs a capacitor. Add one in parallel with a symbolic value (let's say x), and run at the stated 50 Hz — (2π)(50) in rad/s.

type

```
"e,1,0,240:r1,1,0,80-i50:r2,1,0,120+i70:r3,1,0,60+i0:c,1,0,x"→cir
s\ac(cir,2π50.)
solve(s\pf(se)=1.,x)
```

returns

```
-6.3 μF
```

A negative capacitance, and no positive value satisfies the equation — which is probably why the textbook gives no number for this part. It is not a failure of the method: the load is *already* leading, as part (b) said, so no capacitor can bring it to unity. What it needs is the opposite.

Try an inductor instead:

type

```
"e,1,0,240:r1,1,0,80-i50:r2,1,0,120+i70:r3,1,0,60+i0:l,1,0,x"→cir
s\ac(cir,2π50.)
solve(s\pf(se)=1.,x)
```

returns

```
1.5987 H
```

Check it by putting that number back in as the inductor's value: the power factor comes out **1**. An inductor of about 1.6 H in parallel is what brings this system to unity.

LESSON 9

Three-phase circuits

Learn to solve simple *three-phase* circuits in Y-Y, Y-Δ, Δ-Δ and Δ-Y configurations, both balanced and unbalanced, to find line and phase currents, voltages and complex power.

In this lesson you will learn to use Symbulator to solve simple **three-phase** systems, in their four basic configurations of wye-wye, wye-delta, delta-delta and delta-wye. We will see both balanced and unbalanced examples, where you are asked to find currents, voltages and complex power in the source, line and load.

9.1 About solving three-phase circuits

Three-phase circuits can be tricky to solve in Symbulator, for several reasons. It is not always clear which node to use as ground. There are three times as many nodes as in an equivalent single-phase circuit. And sources in a delta array bring mathematical trouble: more unknowns than equations.

Symbulator does a great job on the simple three-phase circuits of a basic Circuits I / II textbook, and here we will see examples and one or two tricks. It would not be my tool of choice for anything larger: as circuits grow, the number of nodes quickly passes what the calculator can solve.

9.2 Wye-Wye

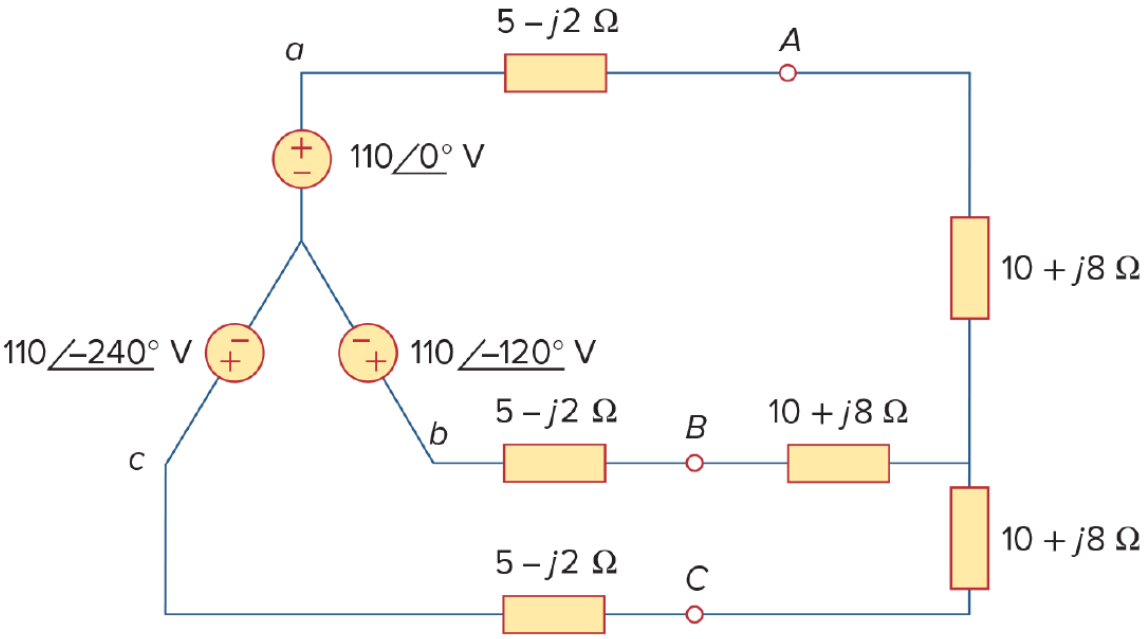
Balanced wye-wye system

AS7's Examples 12.2 & 12.6

Calculate the line currents in the three-wire Y-Y system. Determine the total average power, reactive power, and complex power absorbed at the source and at the load. Assume that the values given for the source are RMS.

SOLUTION

Although not specified in the textbook, to get the answers they give we need to assume the values are RMS. One must describe the circuit carefully in the case of three-phase circuits, because it is very easy to make a mistake. Below is how I describe this one:



AS7's Examples 12.2 and 12.6

type
```
true→userms
"ea0,ag,0,(110.∠0°):eb0,bg,0,(110.∠–120°):ec0,cg,0,(110.∠120°):rat,ag,ad,
5.–i2.:rbt,bg,bd,5.–i2.:rct,cg,cd,5.–i2.:ra0,ad,0,10.+8.i:rc0,cd,0,
10.+8.i:rb0,bd,0,10.+8.i"→cir
s\ac(cir,ω)
```

Some observations about my description:

- Notice that I have specified the node at the centre of both Y's as node **0**. This is something you can do only in the case of balanced Y-Y systems, not for other configurations and not for unbalanced systems.
- Since Symbulator does not differentiate between lower and upper case variables, nodes called **a** and **A** would be considered the same node. Instead, we use the names **ag** and **ad**, where the g reminds us a node is on the generation side and the d reminds us it is on the demand side.

Once the simulation is completed, we ask for the currents on the transmission lines and get:

type
```
{s\aa(irat),s\aa(irbt),s\aa(irct)}
```

returns
```
{"6.809e0∠-21.8°","6.809e0∠-141.8°","6.809e0∠98.2°"}
```

The complex power at the source is `sea0+seb0+sec0`, which gives −2086.2 − 834.5**i** VA. The real part is the average power absorbed by the source; since it is negative, the source is delivering an average power of 2086 W. The imaginary part is the reactive power: 834 var.

The complex power at the load is `sra0+srb0+src0`, which gives 1390.8 + 1112.6**i** VA. The load is consuming an average power of 1391 W and a reactive power of 1113 var.

Before we move on, try simulating this circuit with a different node as the centre of the load's Y array — call it d0, for the demand side. The voltage in that node evaluates to 0 V, because the system is balanced. That is why we could use 0 for both. In an unbalanced system we cannot.

Further wye-wye problems

AS7's Practice Problem 12.2 & 12.6

Calculate the line voltages and the line currents. Also calculate the complex power at the source and at the load. Assume the source voltage is given as RMS.

> A Y-connected balanced three-phase generator with an impedance of $0.4 + j0.3\ \Omega$ per phase is connected to a Y-connected balanced load with an impedance of $24 + j19\ \Omega$ per phase. The line joining the generator and the load has an impedance of $0.6 + j0.7\ \Omega$ per phase. Assuming a positive sequence for the source voltages and that $\mathbf{V}_{an} = 120\underline{/30^\circ}$ V, find: (a) the line voltages, (b) the line currents.

AS7's Practice Problem 12.2 and 12.6

SOLUTION

The textbook does not say so, but you only get its answers if you take the values as RMS.

There are **three** impedances in each line here: one inside the generator, one for the transmission line, and one for the load. The node names below say which is which — **ag** in the generator, **as** on the supply side, **ad** on the demand side.

```
type
true→userms
"ea0,ag,0,(120.∠30°):eb0,bg,0,(120.∠-90°):ec0,cg,0,(120.∠150°):rag,ag,as,
.4+i.3:rbg,bg,bs,.4+i.3:rcg,cg,cs,.4+i.3:rat,as,ad,.6+i.7:rbt,bs,bd,
.6+i.7:rct,cs,cd,.6+i.7:ra0,ad,0,24.+19.i:rb0,bd,0,24.+19.i:rc0,cd,0,
24.+19.i"→cir
s\ac(cir,ω)
```

The question asks for the line voltages — the differences between the lines. For the answers the textbook wants, ask for these:

```
type
{s\aa(vag-vbg),s\aa(vbg-vcg),s\aa(vcg-vag)}
```

```
returns
{"207.8e0∠60.°","207.8e0∠-60.°","207.8e0∠180.°"}
```

> TIP
> **Which line voltage did you want?**
> If you ask me, that is not really the voltage between the lines, because the transmission line starts *after* the generator. At the start of the transmission line the answer is **204.6∠60.03°** V, and at the end of it **198.7∠59.71°** V — ask for `s\aa(vas-vbs)` and `s\aa(vad-vbd)` respectively. All three are different, and which one you want depends on where you are standing.

The line currents are the currents through the transmission-line impedances:

type
```
{s\aa(irat),s\aa(irbt),s\aa(irct)}
```

returns
```
{"3.748e0∠-8.66°","3.748e0∠-128.66°","3.748e0∠111.34°"}
```

Finally the complex power. At the source:

type
```
sea0+seb0+sec0
```

returns
```
-1053.7-842.9i
```

Note this does *not* include the power lost in the source's own internal impedances, which you could add if you wanted them. At the load:

type
```
sra0+srb0+src0
```

returns
```
1011.5+800.8i
```

All correct.

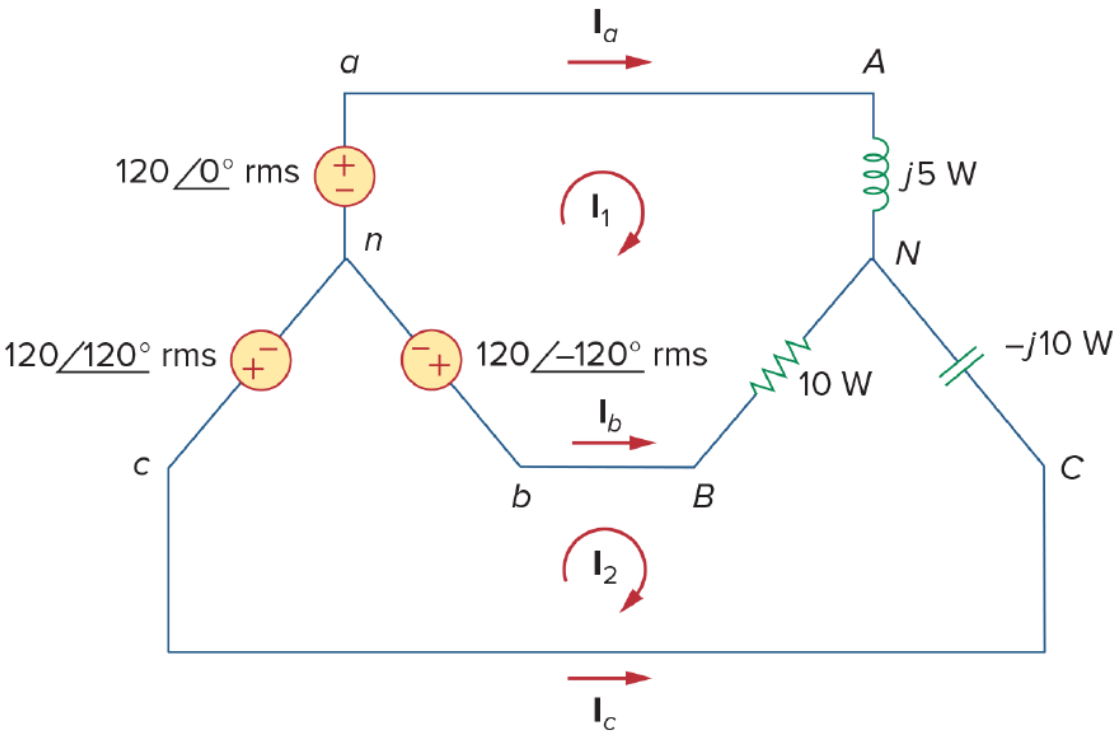
Unbalanced wye-wye system

The balanced case let you use node **0** for the centre of *both* wyes, because the voltage at the centre of the load is zero. Unbalanced, it is not — so the load needs a centre node of its own.

AS7's Example 12.10

For the unbalanced circuit, find (a) the line currents, (b) the total complex power absorbed by the load, and (c) the total complex power absorbed by the source.

SOLUTION



AS7's Example 12.10

The centre of the generator's wye is ground, and the centre of the load's wye is a node of its own, called `n`.

type
```
true→userms
"ea,a,0,(120.∠0°):eb,b,0,(120.∠-120°):ecc,c,0,(120.∠120°):ra,a,n,5.i:rb,b,
n,10.:rcc,c,n,-10.i"→cir
s\ac(cir,ω)
```

type
```
{s\aa(ira),s\aa(irb),s\aa(ircc),sra,srb,srcc,sea,seb,secc}
```

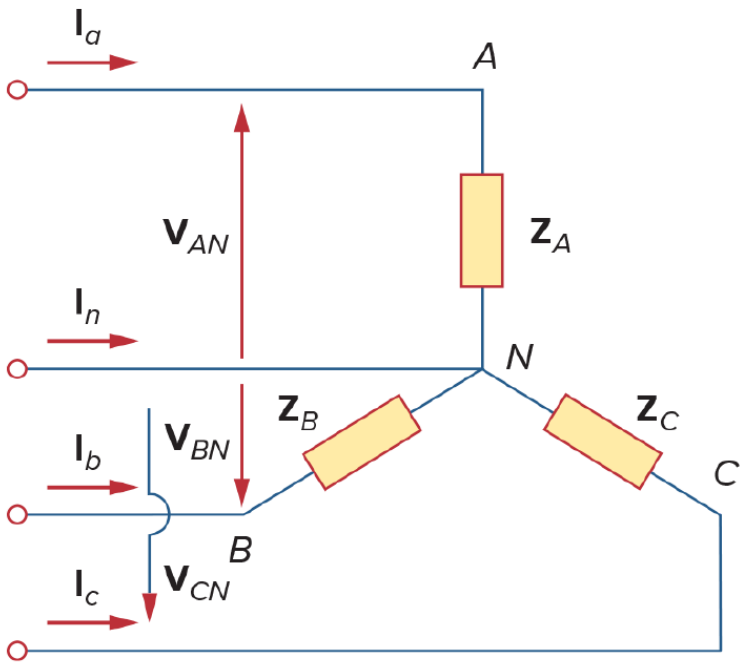
returns
```
{"56.78e0∠0.°","25.46e0∠135.°","42.76e0∠-155.1°",16122.i,6480.,-18282.i,
-6814.,790.6-2951.i,-456.5+5111.i}
```

All correct. Two things are worth checking for yourself. Complex power is conserved, so `sea+seb+secc+sra+srb+srcc` evaluates to zero. And the voltage at the centre of the load is *not* zero, as it was in the balanced case — `vn` is **120.0 – 283.9i** V.

> TIP
>
> **Why `ecc` and `rcc` rather than `ec` and `rc`**
>
> The third phase is named with a doubled letter because on the calculator the short names collide with reserved ones. Version 9 has no such restriction and names the third phase like the other two.

AS7's Example 12.9

The unbalanced Y-load has balanced voltages of 100 V in the *acb* sequence. Calculate the line currents and the neutral current. Take $\boldsymbol{Z_A}$ = 15 Ω, $\boldsymbol{Z_B}$ = 10 + j5 Ω and $\boldsymbol{Z_C}$ = 6 − j8 Ω.



AS7's Example 12.9

SOLUTION

Two things differ from the last one. The source values are *not* RMS this time. And there is a neutral line joining the centre of the load to the centre of the source, which you describe as a short circuit — an **s** element, which takes just a name and two nodes.

Note also the angles: this is the *acb* sequence, so they run the other way round.

```
type
false→userms
"ea,a,0,(100.∠0°):eb,b,0,(100.∠120°):ecc,c,0,(100.∠-120°):sn,0,n:ra,a,n,
15:rb,b,n,10.+i5.:rcc,c,n,6.-i8."→cir
s\ac(cir,ω)
```

```
type
{s\aa(ira),s\aa(irb),s\aa(ircc),s\aa(isn)}
```

```
returns
{"6.667e0∠0.°","8.944e0∠93.43°","10.000e0∠-66.87°","10.06e0∠178.47°"}
```

All four are correct. The neutral carries a current precisely because the load is unbalanced; in the balanced case it would be zero, which is why a balanced three-wire system needs no neutral at all.

9.3 Wye-Delta

A wye source feeding a delta load. The source still has a centre to use as ground; the load does not, which is the only new thing here.

Balanced

AS7's Example 12.3

For the balanced Y-Δ circuit, find the phase currents and the line currents.

> A balanced *abc*-sequence Y-connected source with $\mathbf{V}_{an} = 100\angle 10°$ V is connected to a Δ-connected balanced load $(8 + j4)\Omega$ per phase. Calculate the phase and line currents.

AS7's Example 12.3

SOLUTION

Nothing says these values are RMS, so leave the convention alone — the question asks for no powers, so it makes no difference either way. The three sources get the angles of the abc sequence.

```
type
"ea0,a,0,(100.∠10°):eb0,b,0,(100.∠-110°):ec0,c,0,(100.∠130°):rab,a,b,
8.+4.i:rca,c,a,8.+4.i:rbc,b,c,8.+4.i"→cir:s\ac(cir,ω)
```

The phase currents are the currents in the three load impedances:

```
type
{s\aa(irab),s\aa(irbc),s\aa(irca)}
```

```
returns
{"19.36e0∠13.43°","19.36e0∠-106.57°","19.36e0∠133.43°"}
```

The line currents are the currents the sources deliver, which is the opposite of the current through each source element:

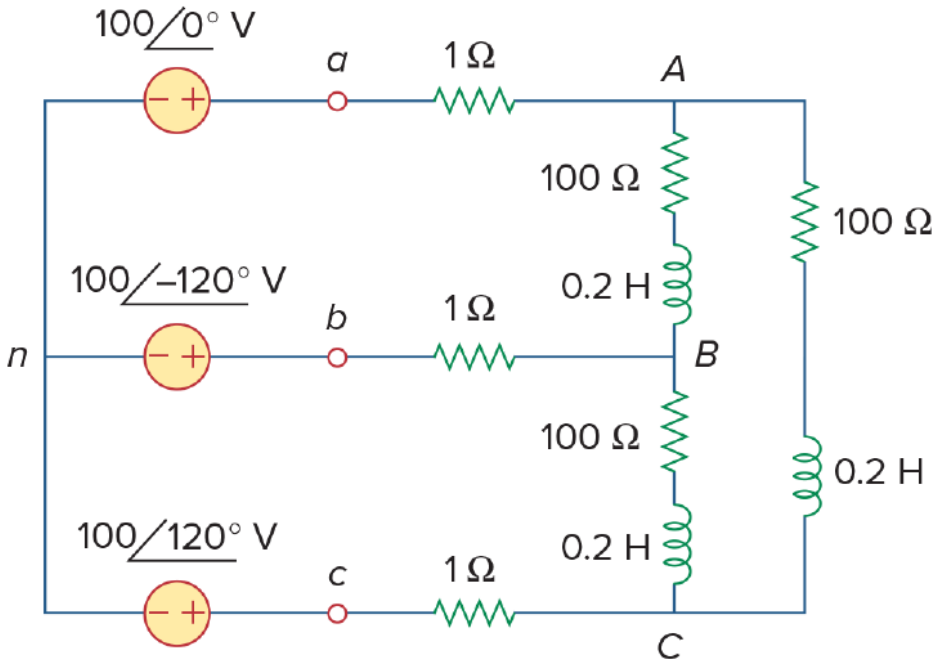
```
type
{s\aa(-iea0),s\aa(-ieb0),s\aa(-iec0)}
```

```
returns
{"33.54e0∠-16.57°","33.54e0∠-136.57°","33.54e0∠103.43°"}
```

Both sets are correct, and they show the relationship you would expect of a delta load: the line current is √3 times the phase current, and lags it by 30°.

AS7's Example 12.11

For the balanced Y-Δ circuit, find the line current I_{aA}, the phase voltage V_{AB}, and the phase current I_{AC}. The source frequency is 60 Hz.



AS7's Example 12.11

SOLUTION

This one has line impedances, so each source reaches the load through a resistor, and the load's three nodes are separate from the source's three.

type
```
"ea1,na1,0,(100.∠0°):eb1,nb1,0,(100.∠–120°):ec1,nc1,0,(100.∠120°):raa,na1,
na2,1:rbb,nb1,nb2,1:rcc,nc1,nc2,1:rac,na2,nc2,100.+24.*π*i:rcb,nc2,nb2,
100.+24.*π*i:rba,nb2,na2,100.+24.*π*i"→cir:s\ac(cir,ω)
```

The line current is the current in one of the line resistors, and the phase voltage is the difference between two load nodes:

type
```
{s\aa(iraa),s\aa(vna2-vnb2),s\aa(irac)}
```

returns
```
{"2.35∠-36.2°","169.94∠30.8°","1.36∠-66.2°"}
```

Unbalanced

If you find a simple unbalanced wye-delta problem, let me know.

9.4 Delta-Delta

Until now, choosing a ground node was easy: the centre of the wye of sources. A delta has no centre, which is the first problem. It is solved by picking one of the delta's own nodes on the generator side and calling it **0**.

The second problem is subtler, and it is not Symbulator's alone — SPICE-like simulators dislike it too. **A triangle of three voltage sources cannot be solved.** The third source adds no information, because the first two already fix the voltages at all three nodes, but it does add an unknown: the current through it. Drop the redundant equation and the system has one unknown too many.

> ⚠ WARNING
>
> **Describe a delta source with two sources, not three**
>
> The trick is to leave one source out — the one opposite the node you chose as ground. Two sources fix all three node voltages, so everything outside the delta behaves exactly as it should.
>
> The cost is that you learn nothing about the currents *inside* the sources: the currents you get for the two you kept are not the currents they would carry in a real three-source arrangement. Everything beyond the delta is right; the inside of the generator is not.

Balanced

AS7's Example 12.4

A balanced Δ-connected load of 20 − j15 Ω is fed by a Δ-connected, positive-sequence generator with $\boldsymbol{V_{ab}}$ = 330∠0° V. Find the phase currents of the load and the line currents.

> A balanced Δ-connected load having an impedance $20 - j15\ \Omega$ is connected to a Δ-connected, positive-sequence generator having $\mathbf{V}_{ab} = 330\underline{/0^\circ}$ V. Calculate the phase currents of the load and the line currents.

AS7's Example 12.4

SOLUTION

Node **c** becomes ground, so the source left out is the one opposite it. To read the line currents we add three shorts to act as the lines.

type

```
"e0a,0,ag,(330.∠120°):eb0,bg,0,(330.∠-120°):sat,ag,ad:sbt,bg,bd:sct,0,cd:
rab,ad,bd,20.-15.i:rbc,bd,cd,20.-15.i:rca,cd,ad,20.-15.i"→cir:s\ac(cir,
ω)
```

The line currents are the currents through the shorts:

type
```
{s\aa(isat),s\aa(isbt),s\aa(isct)}
```

returns
```
{"22.86e0∠6.87°","22.86e0∠-113.13°","22.86e0∠126.87°"}
```

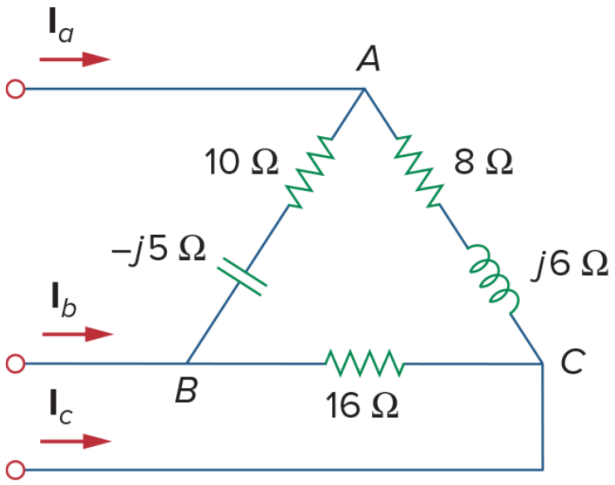
Correct.

Unbalanced

Nothing changes in the method. The load impedances simply differ, and the answers stop being three copies of one another.

AS7's Practice Problem 12.9

The unbalanced Δ-load is supplied by balanced line-to-line voltages of 440 V in positive sequence. Find the line currents, taking $\boldsymbol{V_{ab}}$ as the reference.



AS7's Practice Problem 12.9

SOLUTION

type
```
false→userms
"e0a,0,ag,(440.∠120°):eb0,bg,0,(440.∠-120°):sla,ag,ad:slb,bg,bd:slc,0,cd:
rab,ad,bd,10.-i5.:rbc,bd,cd,16.:rca,cd,ad,8.+i6."→cir:s\ac(cir,ω)
```

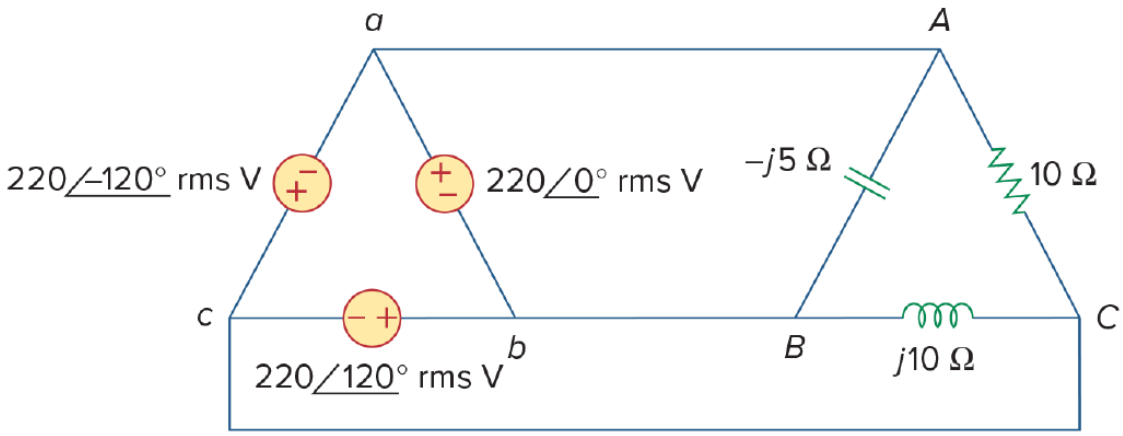
type
```
{s\aa(isla),s\aa(islb),s\aa(islc)}
```

returns
```
{"39.71e0∠-41.07°","64.12e0∠-139.77°","70.13e0∠74.27°"}
```

Correct — and unlike the balanced case, all three differ in magnitude as well as angle.

AS7's Practice Problem 12.10

Find the line currents in the unbalanced three-phase circuit, and the real power absorbed by the load.



AS7's Practice Problem 12.10

SOLUTION

These values are RMS, so set the flag — this one does ask for power.

type
```
true→userms
"e0a,0,ag,(220.∠−120°):eb0,bg,0,(220.∠120°):sla,ag,ad:slb,bg,bd:slc,0,cd:
rab,ad,bd,−i5.:rbc,bd,cd,i10.:rca,cd,ad,10."→cir:s\ac(cir,ω)
```

type
```
{s\aa(isla),s\aa(islb),s\aa(islc),prca+prab+prbc}
```

returns
```
{"64.00e0∠80.1°","38.11e0∠-60.°","42.50e0∠-135.°",4840.0}
```

Correct.

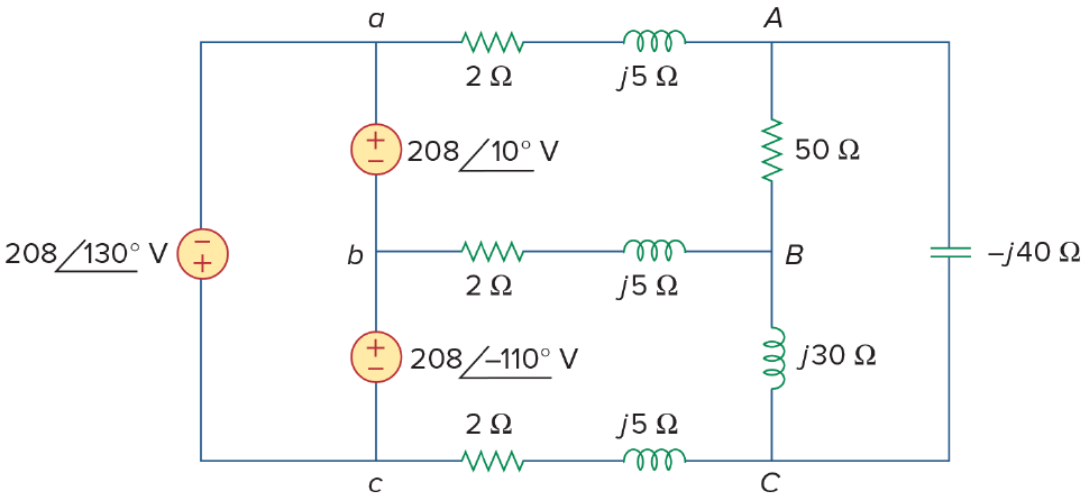
AS7's Example 12.12

For the unbalanced Δ-Δ circuit, find the generator current I_{ab}, the line current I_{bB} and the phase current I_{BC}.

SOLUTION

Two of the three are ordinary element currents. The first — the current *inside* the generator — is the awkward one, and it is dealt with after them.

type
```
false→userms
"e0a,0,ag,(208.∠130°):eb0,bg,0,(208.∠−110°):rla,ag,ad,2.+i5.:rlb,bg,bd,
```



AS7's Example 12.12

```
2.+i5.::rlc,0,cd,2.+i5.::rab,ad,bd,50.::rbc,bd,cd,i30.::rca,cd,ad,-i40."→cir:
s\ac(cir,ω)
```

type

```
{s\aa(irlb),s\aa(irbc)}
```

returns

```
{"9.106e0∠168.48°","5.500e0∠172.47°"}
```

The remaining answer, the generator current, is trickier. They ask for the current in the source we did not simulate — and even for one of the two we did have, we could not trust it, since the real circuit uses three.

Now, I tried something, and I think I got lucky, because I got the answer the book gives. This is what I tried:

type

```
s\aa(-(ie0a+ieb0)/3)
```

What I thought was: the current coming out of the two sources in my simulation would, in reality, come out of three. So adding those two currents and dividing by three may approximate the current out of one source. I am confident that holds in a balanced circuit. This one is not balanced, but it was the best I had. So I tried it.

returns

```
"5.959e0∠-177.18°"
```

And it worked. That is the answer in the book.

> ⚠ **WARNING**
>
> **Treat that as a guess that was checked, not as a method**

> The argument holds in a balanced circuit. This one is not balanced; it was used because there was nothing better to hand, and it happened to land on the book's figure. Check it against something before trusting it elsewhere.

9.5 Delta-Wye

The Δ-Y is the platypus of three-phase systems. The sensible route is usually to convert the Δ source into an equivalent Y and solve it as a Y-Y — and if you are doing this by hand, do that.

Inside Symbulator there is another way: make the **centre of the wye load** the ground node, and describe the delta source with two sources as before.

Balanced

AS7's Example 12.5

For the balanced Δ-Y circuit, find the line currents.

> A balanced Y-connected load with a phase impedance of $40 + j25\ \Omega$ is supplied by a balanced, positive sequence Δ-connected source with a line voltage of 210 V. Calculate the phase currents. Use V_{ab} as a reference.

AS7's Example 12.5

SOLUTION

```
type
false→userms
"eca,c,a,(210.∠120°):ebc,b,c,(210.∠-120°):ra,a,0,40.+i25.:rb,b,0,40.+i25.:
rcc,c,0,40.+i25."→cir:s\ac(cir,ω)
```

Node **0** here is the centre of the load's wye, not a node of the source at all.

```
type
{s\aa(ira),s\aa(irb),s\aa(ircc)}
```

```
returns
{"2.570e0∠-62.01°","2.570e0∠177.99°","2.570e0∠57.99°"}
```

Correct — balanced, so three equal magnitudes 120° apart.

Unbalanced

If you find a simple unbalanced delta-wye problem, let me know.

LESSON 10

Mutual inductance and ideal transformers

Learn how to describe a *mutual inductance* between two inductors or two impedances using the **m** element. Learn how to describe an ideal *transformer* using the **t** element.

Two coils that share a magnetic field are not two separate elements, and a circuit description has to say so. This lesson covers the two ways of saying it: **m** for a mutual inductance between a pair of elements, and **t** for an ideal transformer.

10.1 How to describe mutual inductance

Textbooks give a coupled pair one of two ways, and Symbulator takes both.

- Two inductors in **henries**, coupled by a mutual inductance also in henries.
- Two impedances in **j**Ω, coupled by a mutual impedance also in jΩ. Written this way the coils are **r** elements with imaginary values.

Either way the coupling itself is an **m** element with four fields: a name starting with **m**, the names of the two coupled elements, and the value of the coupling. For example: `m1,l1,l2,1.5`

That is a mutual inductance of 1.5 H between inductors **l1** and **l2**. In jΩ it looks the same, naming two **r** elements instead: `m1,r1,r2,2j`

> ⚠ WARNING
>
> **Don't mix the two**
>
> `m1,l1,r1,2` couples an inductor given in henries to one given in jΩ. It will not be refused and it will not be right. Keep all three values in the same units: all henries, or all jΩ.

The dots are the node order

Every coupled-inductor problem in a textbook has a dot on each coil, marking the ends that share polarity. Symbulator has no way to draw a dot, so **the first node you name in each coupled element is its dotted end**.

That is the whole convention, and it is the thing most likely to give you a correct-looking wrong answer. If you list the nodes of one coil the other way round, the sign of the cou-

pling flips and every current comes back wrong.

> ⚠ WARNING
>
> **Keep the dots consistent**
>
> Read the schematic, decide which end of each coil is dotted, and declare that node first — in both coupled elements. Look at the solved examples below and notice that the dotted ends are always the first node given.

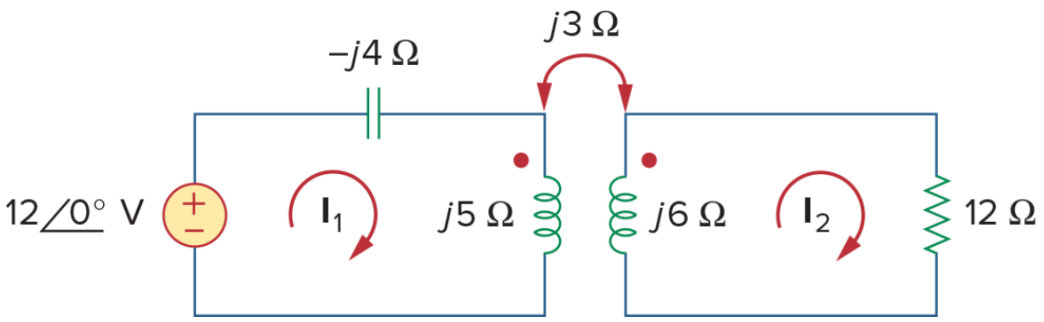
The value of a mutual inductance can use SI prefixes.

What answers do you get? None. An **m** element is a statement about two other elements, not a component with its own voltage and current — its effect shows up in the currents of the elements it couples.

10.2 Instructive m problems

AS7's Example 13.1

Calculate the phasor currents I_1 and I_2 in the circuit.



AS7's Example 13.1

SOLUTION

Three things to notice in the description. The bottom node of both halves is ground. The coils are given in jΩ, so they are **r** elements with imaginary values, and the mutual is imaginary too, to match. And the dotted node of each coupled element is named first.

type

```
false→userms:"e,1,0,12:r1,1,2,-i4:r2,2,0,i5:m,r2,r3,i3:r3,3,0,i6:r4,3,0,
12"→cir:s\ac(cir,ω)
```

type

```
{s\aa(ir2),s\aa(ir4)}
```

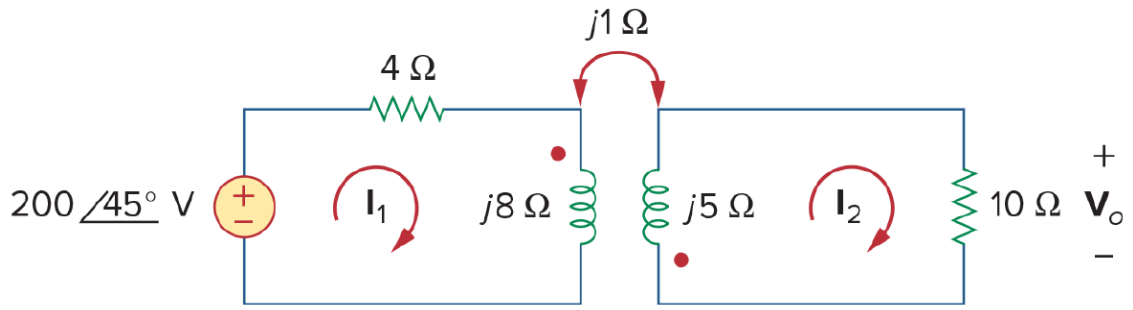
returns

```
{"13.02e0∠-49.4°","2.910e0∠14.04°"}
```

Both are correct.

AS7's Practice Problem 13.1

Determine the voltage V_o in the circuit.



AS7's Practice Problem 13.1

SOLUTION

Again the dotted node of the coupled inductor comes first.

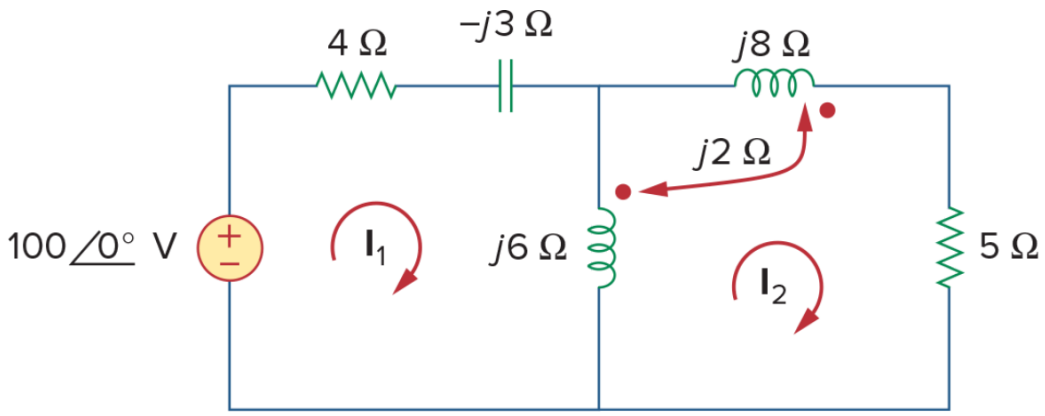
```
type
false→userms:"e,1,0,(200.∠45°):r1,1,2,4.:r2,2,0,i8.:m,r2,r3,i1.:r3,0,o,
i5.:r4,o,0,10"→cir:s\ac(cir,ω)
```

```
returns
"20.00e0∠-134.43°"
```

Correct.

AS7's Example 13.2

Calculate the mesh currents in the circuit.



AS7's Example 13.2

SOLUTION

Notice the node order again, matching the dots.

type

```
false→userms:"e,1,0,100.:r1,1,2,4.-i3.:r2,2,0,i6.:m,r2,r3,i2.:r3,3,2,i8.:
r4,3,0,5"→cir:s\ac(cir,ω)
```

returns

```
{"20.30e0∠3.5°","8.693e0∠19.03°"}
```

Correct.

10.3 How to describe ideal transformers

An ideal transformer has four terminals — two a side. Symbulator simulates the case where **one terminal on each side is grounded**, which covers the textbook problems; you name only the two live nodes.

A **t** element takes five fields: a name starting with `t`, the live node on the first side, the live node on the second side, and the turns on each side. For example: `t1,1,2,80,800`

That is 80 turns on the side connected to node **1** and 800 on the side connected to node **2**, wound the same way. Only the ratio matters, so `1,10` would describe the same transformer.

Polarity is a minus sign. If the dots are on opposite ends, make one of the two turn counts negative: `t,a,b,1,-5`

It does not matter which of the two you make negative, as long as one of them is.

What answers do you get? The voltages at the live nodes, as for any node in the circuit, and **the current entering the transformer at each of its live nodes**, named with the transformer and the node — for a transformer `t` on nodes **1** and **2**, that is `it1` and `it2`.

10.4 Instructive t problems

AS7's Figure 13.33

Obtain V_{Th} and Z_{Eq} for the part of the circuit to the right of nodes **a** and **b**.

SOLUTION

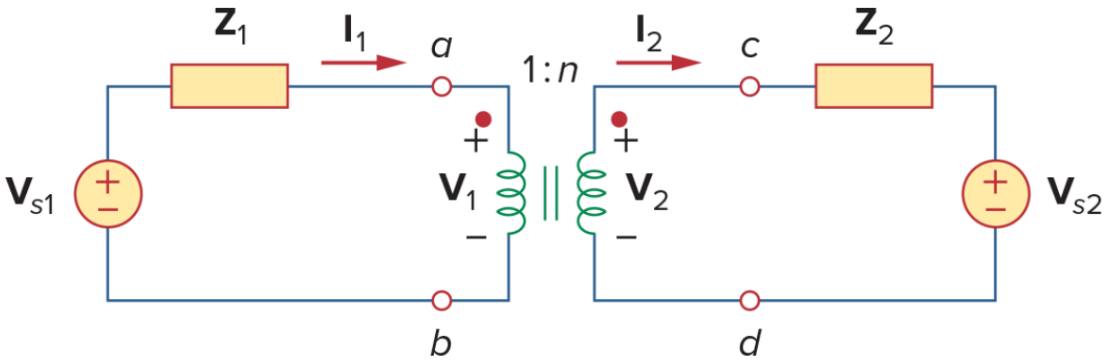
Everything left of a and b is ignored. Note that this one is entirely symbolic — the turns ratio, the impedance and the source are all names.

type

```
"t,2,3,1,n:r2,3,4,z2:e2,4,0,vs2"→cir:s\th(cir,2,0)
```

returns

```
{vs2/n,z2/n^2}
```

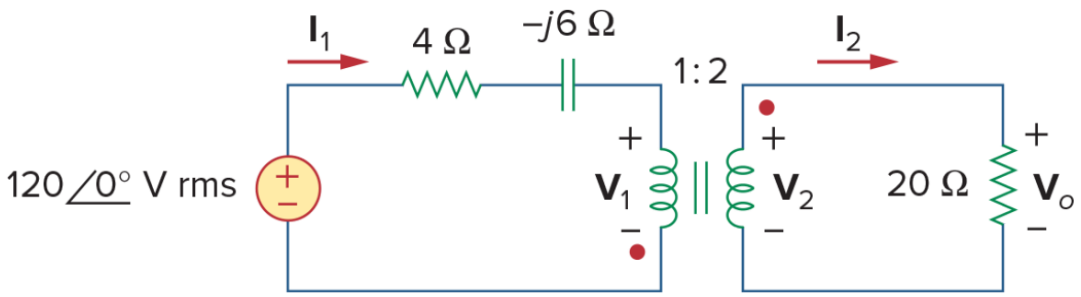


AS7's Figure 13.33

Correct — the transformer refers the source by the turns ratio and the impedance by its square, which is exactly what an ideal transformer does.

AS7's Example 13.8

For the ideal transformer circuit, find the source current I_1, the output voltage V_o, and the complex power supplied by the source.



AS7's Example 13.8

SOLUTION

One of the turn counts is negative here, because the dots are of opposite polarity.

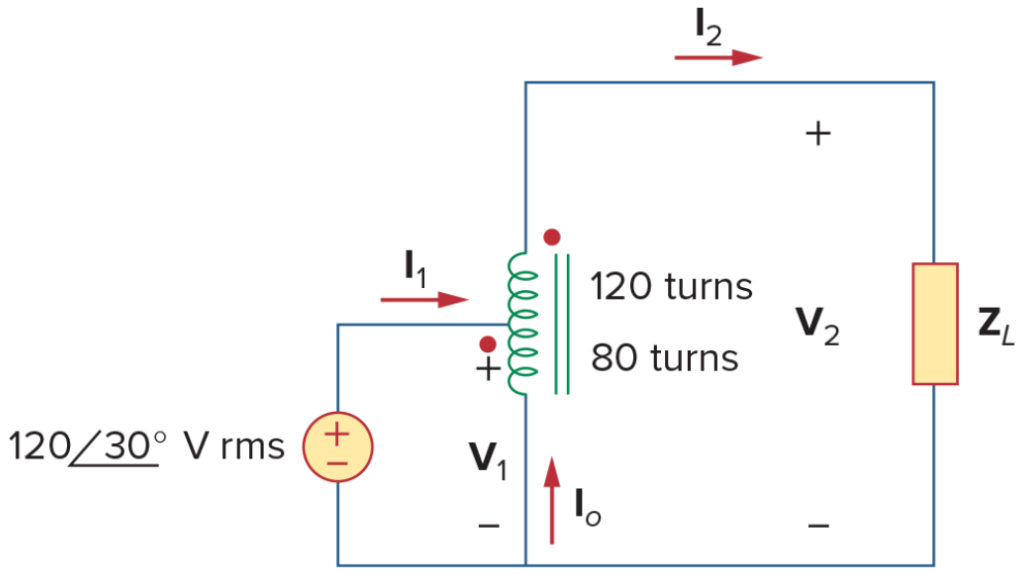
type

```
true→userms:"e,1,0,120.:r1,1,2,4.-i6.:t,2,3,1,-2:ro,3,0,20."→cir:
s\ac(cir,ω)
```

returns

```
{"11.09e0∠33.69°","110.9e0∠-146.31°","1.331e3∠-33.69°"}
```

All three are correct.



AS7's Example 13.11

AS7's Example 13.11

Refer to the autotransformer circuit. Calculate I_1, I_2 and I_o if Z_L = 8 + j6 Ω, and the complex power supplied to the load.

SOLUTION

Symbulator has no dedicated element for an autotransformer. The same **t** element does it: the input is node **1**, the output is node **2**, and the turns are the 80 of the lower winding and the 80 + 120 of the whole.

```
type
true→userms:"e,1,0,(120.∠30°):t,1,2,80,80+120:rl,2,0,8.+i6."→cir:
s\ac(cir,ω)
```

This is how the three currents come out of the two the transformer reports:

- I_1 is the current entering the transformer at node **1**, `it1`.
- I_2 is the opposite of the current entering it at node **2**, `it2`.
- I_o is the opposite of the sum of the two, since what enters the transformer at its two live nodes leaves it through the common one.

```
type
{s\aa(it1),s\aa(-it2),s\aa(-it1-it2)}
```

```
returns
{"75.00e0∠-6.87°","30.00e0∠-6.87°","45.00e0∠173.13°"}
```

All three are correct.

LESSON 11

Bode plots and resonance

Learn to do gain and phase *Bode plots* for transfer functions of ω and s with the **bode** tool. Find the *resonant frequency* of an answer by solving for the frequency at which the reactance vanishes.

> ⓘ NOTE
>
> **Not in this version**
>
> Symbulator 8 has no Bode plotter. The Nspire's own graphing tools will plot a transfer function once you have it, and finding the transfer function itself is covered in the lesson on the complex frequency domain.

LESSON 12

The complex frequency domain

Learn to solve *complex-frequency domain* problems using **fd**. Move between the time domain and the s-domain with the **t2s** and **s2t** shortcuts. Learn to solve transfer function problems.

Every analysis so far has answered in the same domain you asked in. This one does not, and that is the point of it: the complex frequency domain, or s-domain, is where a differential equation becomes an algebraic one. Symbulator will do the algebra; your job is to be clear about which domain each number is in.

12.1 s-domain analysis

Symbulator has a tool called **fd** that solves a circuit in the domain of complex frequency, where $s = j\omega$.

The rule that matters is this one, and it catches everybody once:

> ⚠ WARNING
>
> **FD is in s. TR is in time.**
>
> The value of a source given to **FD** is read as an s-domain expression. The same value given to **TR** is read as a function of time.
> A source value of 1 in FD is not the same thing as a source value of 1 in TR. In FD it is an impulse; in TR it is a constant. Neither is wrong — they are answers to different questions — but they are not interchangeable, and nothing will warn you.

The same applies to what comes back: **FD answers in s**. Voltages and currents are functions of s, and if you want them in time you have to transform them back.

Resistances are read in the s-domain too, which is more useful than it sounds: it means you can describe a capacitor or an inductor *as an impedance*, writing its value in terms of s, as long as its initial condition is zero.

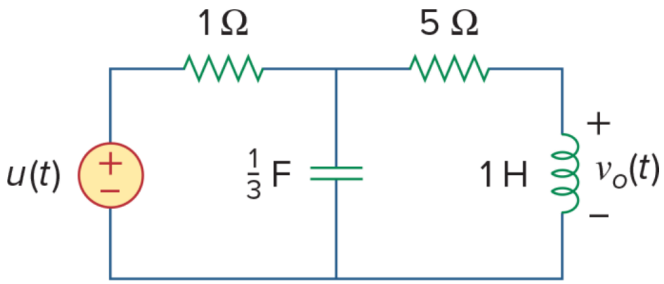
Other than the domain, FD and TR are twins. Capacitors and inductors given in farads and henries are described identically in both, and so is everything else — shorts, op-amps, transformers, all of it.

AS7's Example 16.1

Find $v_o(t)$ in the circuit, assuming zero initial conditions.

SOLUTION

Before we start: this problem does not need the s-domain at all. The source is given as



AS7's Example 16.1

a function of time and the answer is wanted as a function of time, so TR will do it in one step. Notice that the source value is written in the time domain, because that is what TR reads:

```
type
"e,1,0,u(t):r1,1,2,1:r2,2,o,5:c,2,0,1/3,0:l,o,0,1,0"→cir:s\tr(cir):vo
```

Now let us do it properly, in the s-domain, which is what you would want if any of the intermediate answers interested you.

The extreme version describes everything as an impedance in s — the source as `1/s`, the capacitor as `3/s`, the inductor as `s`:

```
type
"e,1,0,1/s:r1,1,2,1:r2,2,o,5:r3,2,0,3/s:r4,o,0,s"→cir:s\fd(cir):vo
```

That was to prove it can be done. In practice the schematic gives you farads and henries, so give Symbulator farads and henries and let it do the converting:

```
type
"e,1,0,1/s:r1,1,2,1:r2,2,o,5:c,2,0,1/3,0:l,o,0,1,0"→cir:s\fd(cir):vo
```

Both give the same answer, in the s-domain:

$$v_o(s) = \frac{3}{s^2 + 8s + 18}$$

Note the source: `1/s`, not `1`. The unit step is `1/s` in the s-domain, and writing `1` there would have described an impulse instead.

12.2 The t2s and s2t shortcuts

Having to transform every source by hand before typing it, and every answer by hand afterwards, would make the s-domain more trouble than it saves. Symbulator gives you both directions.

s\t2s(expression) converts an expression that is a function of time into its s-domain equivalent, and **s\s2t(expression)** converts back.

These are wrappers around Laplace Functions's Laplace and inverse Laplace transforms. Symbulator uses DiffEq, by Lars Frederiksen, as a dependency for them.

So the whole problem can be done in the s-domain while you write the source in the domain the schematic gave it to you in:

```
type
"e,1,0,s\t2s(u(t)):r1,1,2,1:r2,2,o,5:c,2,0,1/3,0:l,o,0,1,0"→cir:
s\fd(cir):s\s2t(vo)
```

The curly-bracket shorthand

There is a shorter way to say "this source value is in the time domain": wrap it in curly brackets, and Symbulator converts it for you.

```
type
"e,1,0,{u(t)}:r1,1,2,1:r2,2,o,5:c,2,0,1/3,0:l,o,0,1,0"→cir:s\fd(cir)
```

The shorthand works only inside a circuit description, so the answer still needs `s\s2t` when you ask for it:

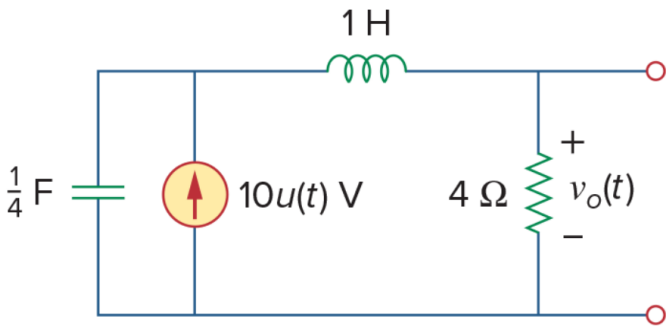
```
type
s\s2t(vo)
```

Either way, the same answer.

12.3 Instructive FD problems

AS7's Practice Problem 16.1

Determine $v_o(t)$ in the circuit.



AS7's Practice Problem 16.1

SOLUTION

The same shape as the example above, so the same approach: describe it in FD with the

source converted, then transform the answer back.

```
type
"e,1,0,s\t2s(u(t)):r1,1,2,6:c,1,0,1/4,0:l,1,o,2,0:r2,o,0,3"→cir:
s\fd(cir):s\s2t(vo)
```

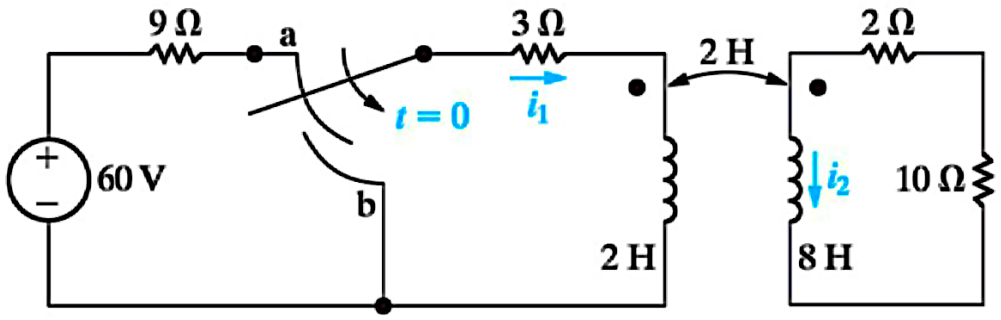
12.4 Transfer function problems

A transfer function is a ratio of two answers, so there is nothing new to learn: solve the circuit with a symbolic source, then divide.

Because both answers are functions of s, so is their ratio — which is exactly what a transfer function is meant to be.

NR11's Example 13.7

Find the transfer function H(s) = V_o/V_s.



NR11's Example 13.7

SOLUTION

Give the source a symbolic value and solve in FD:

```
type
"e,1,0,vs:r1,1,2,1'k:c,2,0,1'μ:r2,2,o,1'k"→cir:s\fd(cir):vo/vs
```

The answer is a function of s, as a transfer function should be.

LESSON 13

Two-ports

Find the *two-port equivalent* of a network using the **port** script. Learn how to include *two-ports* in your circuit using the **z**, **y**, **h**, **g**, **a** and **b** elements. Calculate gains with the **gain** tool.

A two-port is a network you have stopped caring about the inside of. What is left is four numbers relating the voltage and current at one pair of terminals to the voltage and current at the other. Symbulator will find those four numbers for you, and will also take them as an element in a larger circuit.

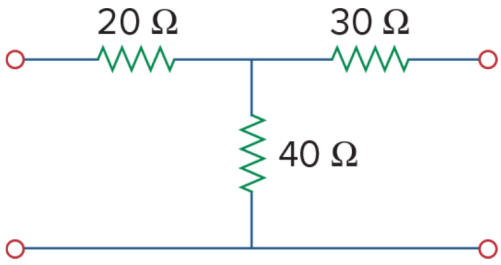
13.1 Find two-port equivalents

The tool is called **port**. It takes the network and the two **top** nodes of the pair of terminals you want to reduce. The two bottom nodes are always assumed to be ground.

There are six kinds of parameters — z, y, h, g, a and b — and you choose which you want.

AS7's Example 19.1

Determine the z parameters for the circuit.



AS7's Example 19.1

SOLUTION

First describe the network, exactly as you would describe any circuit:

```
type
"r1,1,2,20:r2,2,0,40:r3,2,3,30"→cir:s\port(cir,1,3)
```

When asked, specify DC as the analysis and `z` as the parameter type.

Then it wants a name. You may add between one and six more characters, or leave it empty — the Nspire does not reserve `z11`, `z12`, `z21` and `z22`, so the two-port can simply

be called `z`.

returns
```
z11 is 60, z12 is 40, z21 is 40, z22 is 70
```

Correct.

13.2 Use two-ports as elements

A two-port can be an element in a bigger circuit. Its description is three fields: a name whose first letter says which kind it is, then the **top left** node and the **top right** node. Both bottom nodes are ground, always — which is why neither top node may be `0`, and the two may not be the same node; Symbulator stops with a message if they are.

That is a z-type two-port between nodes **1** and **2**.

Giving it its four parameters

Three possibilities.

Store them first, in variables named after the two-port plus 11, 12, 21 and 22:

type
```
40→z11:i*20→z12:i*30→z21:50→z22
```

Enter them when prompted. If you run a simulation without storing them first, Symbulator asks whether you want to define them, and you can use SI shorthand in the values you type.

Leave them undefined, in which case they stay symbolic and appear as variables in the answers.

What answers do you get? The current entering each port, named with the two-port and the **node** it sits on — for a two-port **z** on nodes **1** and **2**, that is `iz1` and `iz2`.

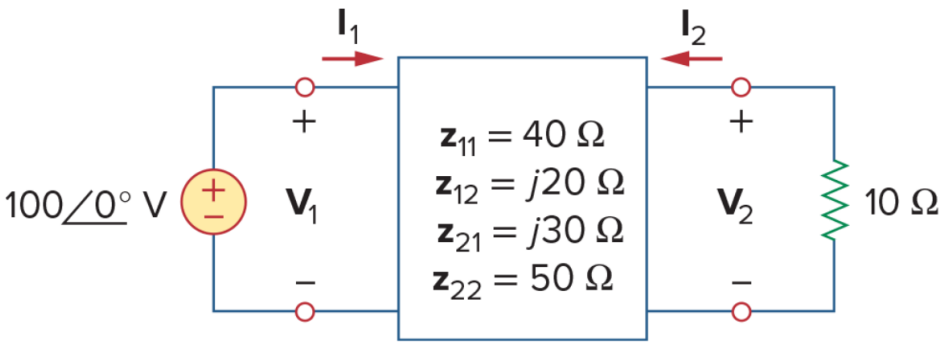
AS7's Example 19.2

Find I_1 and I_2 in the circuit.

SOLUTION

type
```
"e,1,0,100:z,1,2:r1,2,0,10"→cir:s\ac(cir,ω)
```

type
```
s\aa(iz1)
s\aa(iz2)
```

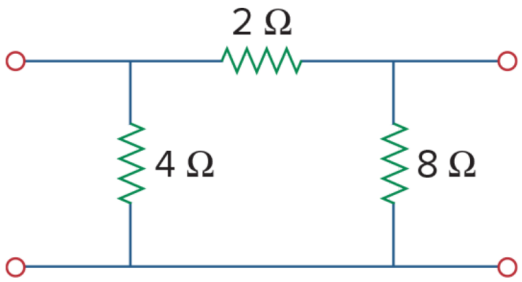


AS7's Example 19.2

13.3 Instructive two-port problems

AS7's Example 19.3

Obtain the y parameters for the network.



AS7's Example 19.3

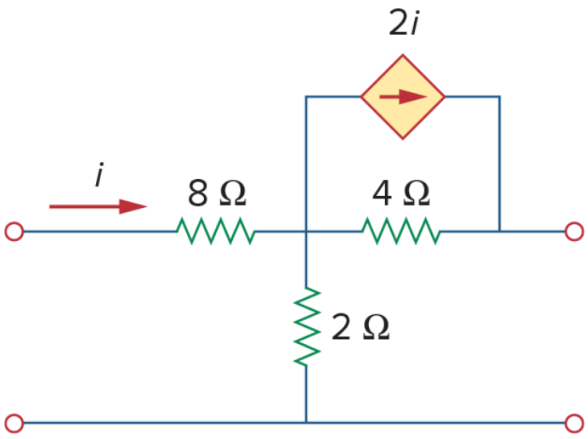
SOLUTION

type
```
"r1,1,0,4:r2,1,2,2:r3,2,0,8"→cir:s\port(cir,1,2)
```

type
```
{y11,y12,y21,y22}
```

returns
```
{3/4,-1/2,-1/2,5/8}
```

Correct.



AS7's Example 19.4

AS7's Example 19.4

Determine the y parameters for the two-port shown. Note the dependent source.

SOLUTION

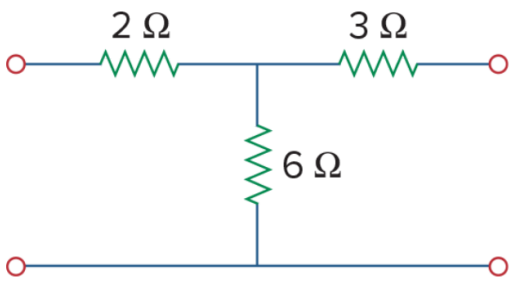
type
```
"r1,1,2,8:r2,2,0,2:r3,2,3,4:j,2,3,2*ir1"→cir:s\port(cir,1,3):{y11,y12,
y21,y22}
```

returns
```
{3/20,-1/20,-1/4,1/4}
```

Correct.

AS7's Example 19.5

Find the hybrid parameters for the two-port network.



AS7's Example 19.5

SOLUTION

type

```
"r1,1,2,2:r2,2,0,6:r3,2,3,3"→cir:s\port(cir,1,3):{h11,h12,h21,h22}
```

For h, g, a and b there is no name clash, so the single letter is available. Specify DC and `h`, and leave the extra character empty.

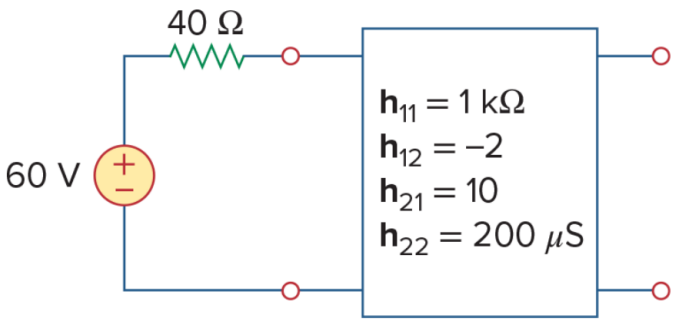
returns

```
{4,2/3,-2/3,1/9}
```

Correct.

AS7's Example 19.6

Determine the Thévenin equivalent at the output port of the circuit, whose h parameters are 1000 Ω, −2, 10 and 2×10^{-4} S.



AS7's Example 19.6

SOLUTION

type

```
"e,1,0,60:r1,1,2,40:h,2,3"→cir:1000.→h11:-2→h12:10→h21:2.e-4→h22:
s\th(cir,3,0):{vth,req}
```

returns

```
{-29.69,51.46}
```

Correct.

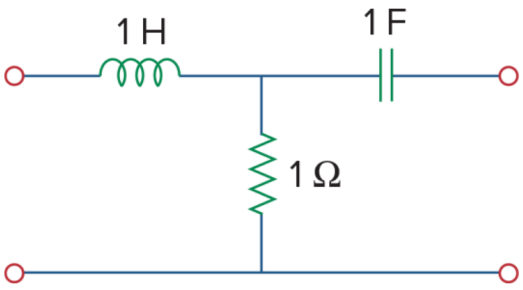
AS7's Example 19.7

Find the g parameters as functions of s for the circuit.

SOLUTION

type

```
"l,1,2,1:r1,2,0,1:c,2,3,1"→cir:s\port(cir,1,3):{{g11,g12},{g21,g22}}
```

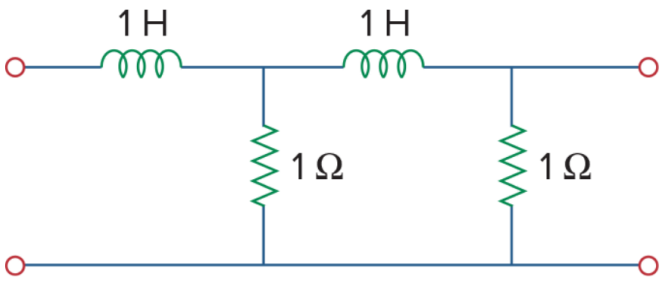


AS7's Example 19.7

returns
```
{1/(s+1),-1/(s+1),1/(s+1),(s^2+s+1)/(s*(s+1))}
```

AS7's Practice Problem 19.7

For the ladder network, determine the g parameters in the s domain.



AS7's Practice Problem 19.7

SOLUTION

type
```
"l1,1,2,1:r1,2,0,1:l2,2,3,1:r2,3,0,1"→cir:s\port(cir,1,3):{{g11,g12},
{g21,g22}}
```

returns
```
[[(s+2)/(s^2+3*s+1),-1/(s^2+3*s+1)][1/(s^2+3*s+1),s*(s+2)/(s^2+3*s+1)]]
```

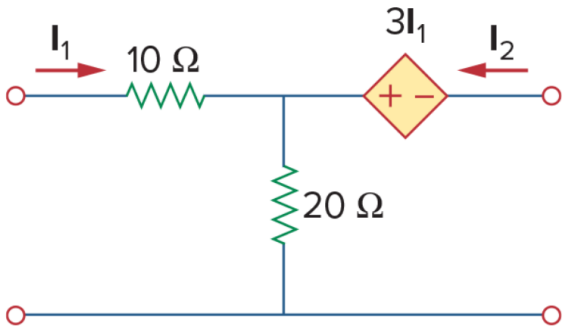
Correct.

AS7's Example 19.8

Find the transmission parameters for the two-port network.

SOLUTION

type
```
"r1,1,2,10:r2,2,0,20:e,2,3,3ir1"→cir:s\port(cir,1,3):[[a11,a12][a21,a22]]
```



AS7's Example 19.8

returns
```
[[1.765,15.29][.0588,1.176]]
```

Correct.

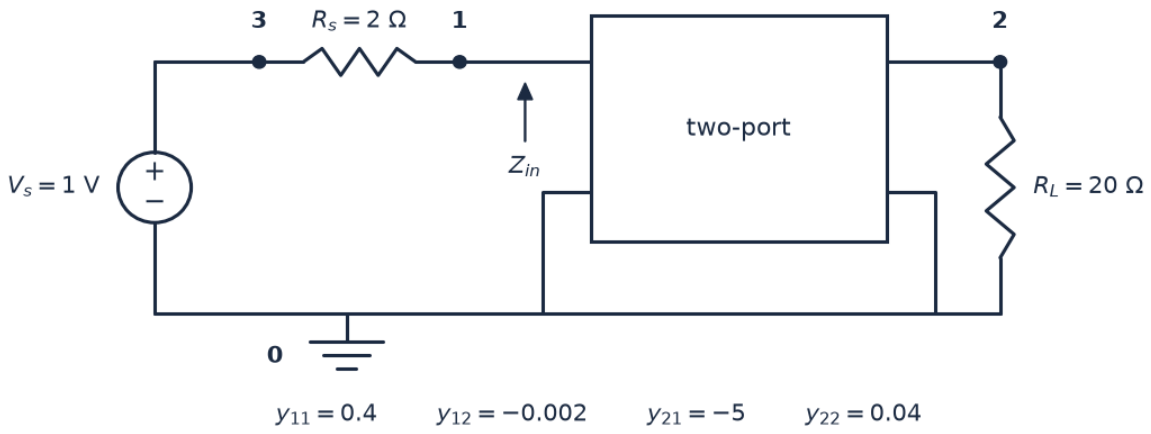
13.4 The gain tool

Network problems ask for gains: voltage gain, current gain, power gain, and the impedance seen at the input. Symbulator has a tool for that, called **gain**.

It wants four values — the voltage and current at the input, then the voltage and current at the output — and answers with all four figures at once.

Gain Example 1

For the circuit, find G_v, G_i, G_p and Z_{in}. The two-port has y parameters y_{11} = 0.4 S, y_{12} = −0.002 S, y_{21} = −5 S and y_{22} = 0.04 S.



Gain Example 1

SOLUTION

The source is 1 V because its value does not matter to a gain — it cancels. A symbolic `vs` would do just as well.

```
type
"es,3,0,1:rs,3,1,2:rl,2,0,20:y,1,2"→cir:s\dc(cir)
```

```
type
s\gain()
```

```
returns
Av 55.6, Ai -9.62, Ap 534, Zin 3.46 Ω
```

These are correct.

Gain Example 2

Find G_v, G_i, G_p and Z_{in} for a two-port with z parameters z_{11} = 4 Ω, z_{12} = 1.5 Ω, z_{21} = 10 Ω and z_{22} = 3 Ω, driven by a source V_s with 5 Ω in series and loaded with 2 Ω.

There is no picture of this one — the description below is the whole circuit.

SOLUTION

The same shape as before, with z parameters instead of y.

```
type
"es,3,0,1:rs,3,1,5:rl,2,0,2:z,1,2"→cir:s\dc(cir)
```

When prompted, specify DC as the analysis and **z** as the parameter type. You can store the parameters in variables beforehand or type them when asked.

```
type
s\gain()
```

It asks for the in voltage, in current, out voltage and out current. Type `v1`, `izp1`, `v2` and `izp2` respectively.

```
returns
Gv 4, Gi -2, Gp 8, Zin 1 Ω
```

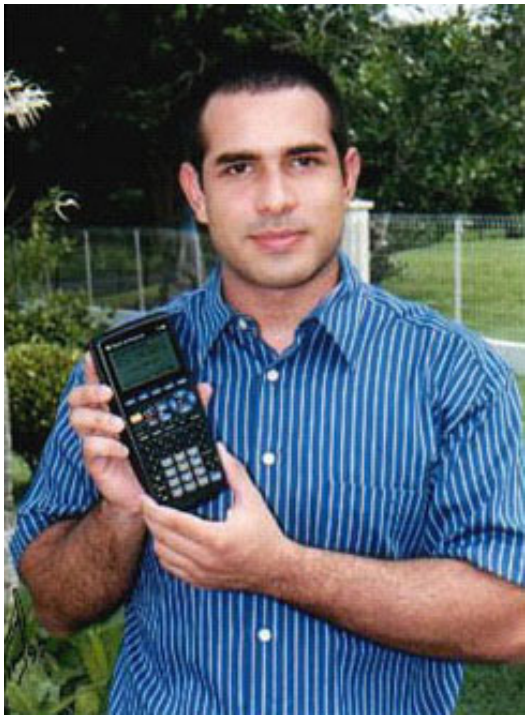
These are correct.

Roll the credits

If you are interested in the people behind Symbulator, here's who made it and who helped, plus some acknowledgements. And some glowing reviews received over the decades.

Who made Symbulator

Symbulator and its documentation, in all its versions up to this date, have been made by me, **Roberto Perez-Franco**. I started coding it as an engineering student at the Technological University of Panama (Universidad Tecnológica de Panamá) in Azuero, and the timeline below has the dates.



Roberto in 2000, with a TI-89

A timeline

Date	Event
30 March 1999	The first line of code, at the Technological University of Panama in Azuero.
2 April 1999	Version 0.92, the first public release, goes onto the Internet.
29 May 1999	Version 0.994 — the equivalent of version 1.
19 August 1999	Version 2, presented in person for the first time, at UTP Azuero.
October 1999	The Elect SCS plug-in. The program is still called SCS.
November 1999	The paper is written.
Late 1999 or early 2000	Version 3.
14 May 2000	The Power Tools plug-in. The program is now called Symbulator.
October 2000	First place, IEEE Region 9 Student Paper Competition.
December 2000	The paper is published in Panama, in the bulletin of the IEEE section.
9 January 2001	Version 4.
30 March 2001	Version 5, called Q.
9 July 2001	The thesis is defended, and graded 100 out of 100.
September 2001	The paper is published in Barcelona, in the magazine of the IEEE student branch.
June 2013	Version 6, written at MIT.
July 2023	Versions 7 and 8, written in Melbourne.
13 August 2026	The symbulator package goes onto PyPI.
28 August 2026	Version 9, a public beta.
1 October 2026	Versions 9 and 8 leave beta.

What Symbulator did for me

I've never made any money from Symbulator: it has always been and will remain free of charge. But I have benefited from Symbulator in other ways. Thanks in large part to Symbulator, I landed my first job, received a Distinguished Alum award from UTP in 2008 and an Outstanding Young Person (TYOP) award from the Panamanian chapter of Junior Chamber International (JCI), and won a Fulbright Scholarship that took me to a Master of Engineering and later a PhD in Engineering Systems at the Massachusetts Institute of Technology. MIT then hired me, and I lived in Boston for almost twelve years. I now live in Melbourne, Australia. You can say that writing Symbulator changed my life. Beyond my lifetime, Symbulator may preserve a part of my mind as it solves circuits for new generations, the same way a composer lives in the tunes or a writer in the words they conceived.

Dedication

Two people made Symbulator possible, with their love and support.

The first is my wife **Monica**, who had the patience to live with a cyborg for all the years my mind was focused on coding and documenting. *Mi reina*, my attention was elsewhere, but my heart was always with you.

The second is my late father **Tito**, who invested time and money in my education and bought me every gadget and every book I needed or wanted for my studies, with a blind faith that one day something good would come of it. I'm happy to know that I made you proud, Dad, and I miss you every day.

How to reach me

If you liked Symbulator, let me know. My email is my given name, then my last name with a hyphen, as a dot-com address, like so: rxxxxxx@pxxxx-fxxxxx.cxx

Acknowledgements

Human collaborators

Over the past decades, many people have made Symbulator better through their suggestions and corrections, on both the software and the documentation. Many were complete strangers at first and became dear friends through our exchanges. Others were dear friends that became collaborators.

Versions 1 through 6 received feedback and suggestions from José Vega (Panama), Tim Hutcheson (USA), Lars Frederiksen (Denmark), Joe Riel (USA), Arne Harstad (Norway), Erwin Baert (Belgium), Charles 'Chuck' Ware (USA), Doug Burkett (USA), Reinhard Willinski (Germany), Kamil Malinski (Austria), Jake Adams (USA), Daniele Martini (Italy), Rozgonyi Szabolcs (Hungary), Michael Rans (UK), Alex Astashyn (Russia), Al Charpentier (USA), Nevin McChesney (USA), Ivan Oro Yu (Panama), Pepe Iborra (Spain), and Dave Conklin (USA). **Versions 7 and 8** received feedback and suggestions from Carlos Perez Ortega (Chile) and Qifan Wang (China).

Version 9 benefited from feedback and excellent suggestions from my dear old friend Antony García (Panama), who is not only one hell of an engineer and a consummate circuit theorist, but who also taught the same Circuits class at UTP as Prof. Eliane Boulet for many years. I met Antony in May 2013, while he was a student at UTP Azuero in the same major that I had completed more than a dozen years earlier. I was immediately struck by his brilliance, and we became friends. In August 2026, I shared an early version of Symbulator 9 with Antony while he was a PhD candidate in the ECE Department at WPI (MA, USA). Antony generously provided me with a series of fantastic recommendations — from his vantage point as both a student and a teacher of circuit theory — regarding features he would like to see in Symbulator. I worked with Claude over a few weeks in September to turn them into reality.

My thanks to all of them. I truly am in their debt.

Software I relied on

Symbulator 8 relies on Lars Frederiksen's Laplace functions, adapted for the TI-Nspire by Philippe Fortin. I have renamed that file **LF** here, in homage to Lars.

Problem credits

To make this book useful to actual students, the problems were chosen from widely used textbooks on electric circuit analysis, and their circuit diagrams are reproduced here as they appear in the source.

Code	Textbook
AS2	*Fundamentals of Electric Circuits* (2nd ed.), Charles K. Alexander and Matthew N. O. Sadiku, McGraw-Hill, 2004
AS5	*Fundamentals of Electric Circuits* (5th ed.), Alexander and Sadiku, McGraw-Hill, 2013
AS7	*Fundamentals of Electric Circuits* (7th international ed.), Alexander and Sadiku, McGraw-Hill, 2020
B11	*Introductory Circuit Analysis* (11th ed.), Robert L. Boylestad, Pearson – Prentice Hall, 2007
Bo2	*Elementary Linear Circuit Analysis* (2nd ed.), Leonard S. Bobrow, Oxford, 1987
HK5	*Engineering Circuit Analysis* (5th ed.), William H. Hayt Jr. and Jack E. Kemmerly, McGraw-Hill, 1993
NR9	*Electric Circuits* (9th ed.), James W. Nilsson and Susan A. Riedel, Prentice Hall, 2011
NR11	*Electric Circuits* (11th ed.), Nilsson and Riedel, Pearson, 2020
NR12	*Electric Circuits* (12th ed.), Nilsson and Riedel, Pearson, 2023
RM3	*Circuit Analysis: Theory and Practice* (3rd ed.), Allan H. Robbins and Wilhelm C. Miller, Thomson – Delmar Learning, 2004
TR5	*The Analysis and Design of Linear Circuits* (5th ed.), Roland E. Thomas and Albert J. Rosa, Wiley, 2006

For the purpose of teaching students how to use Symbulator, these diagrams are reproduced under the principle of fair use. No copyright infringement is intended.

Praise received

Over the past quarter of a century, the words of encouragement I received from users of Symbulator were my incentive to keep working on it. Here is a little of that precious collection.

> *The masterpiece of TI-89 programming.*
>
> — **Alex Astashyn**, EE+CoE+CoS major, Polytechnic University, Brooklyn, New York, USA

Symbulator is and always will be the best program for TI-89/92/V200.

— **Charles Ware**, EE major, Columbus, Ohio, USA

Roberto, your program Symbulator is by itself a reason to buy the calc. It's a masterpiece of programming.

— **Pepe Iborra**, Telecommunications, Spain

The program is really a masterpiece. It gives you everything you need, very accurate, very fast. It's really what every Electrical Engineer dreams of. It's so easy to use and it does everything! DC, AC, Transient analysis, Thévenin, Bode plots... everything! I'm really amazed.

— **Nikolaos Trichakis**, EE major, Aristotle University of Thessaloniki, Greece

I wish I had it while going through school.

— **Joe Riel**, BSEE, creator of Syrup (a symbolic circuit simulator for Maple V), University of California, Irvine, USA

Congratulations for your program: it's simply fantastic. Symbulator is worth the cost of a TI-89!

— **Pier Giorgio Raponi**, engineering student, Pisa, Italy

You've made an excellent program. Symbulator is really the best circuit simulator ever made for a calculator.

— **Rui Sebastião**, EE major, University of Coimbra, Portugal

*Symbulator is probably the most useful program ever made for a TI calculator. I am an electronics student and even paid for the EE*Pro software, yet I find myself using Symbulator more because it is much better at what it does.*

— **Josh Cunningham**, electronics major

Symbulator, by far, has been the best tool for EE I have come across yet. [...] It is truly an impressive piece of work. You should be proud. It certainly is helping me to focus on my studying of EE instead of trying to remember all the math.

— **Jim Wachtel**, EE, Process Control & Electrical Co., St. Louis, Missouri, USA

Licence

From June 2013, Symbulator was offered under a Creative Commons BY-NC-SA licence. Since 2026 it is open source under the MIT licence, which supersedes it. If you want to port it to another platform, you are free to do so, provided the copyright notice and the attribution of authorship travel with it.

Index

aa tool, 167

capacitor, 123
circuit description, 8
conductance, 37
current source, 36

dependent source, 41, 172

er script, 67
expert mode, 31

frequency domain, 206

gain tool, 216
ground node, 10

impedance, 165
inductor, 124

Laplace transform, 207

mutual inductance, 199

only tool, 126
operational amplifier, 97

phasor, 165
polar phasors, 166
power factor, 181
pr tool, 38

resistor, 9
RMS, 180

short circuit, 65
SI prefixes, 10
symbolic circuit, 28

th script, 68
three-phase circuits, 186
transformer, 202
two-port, 210

unbalanced system, 189

voltage source, 11