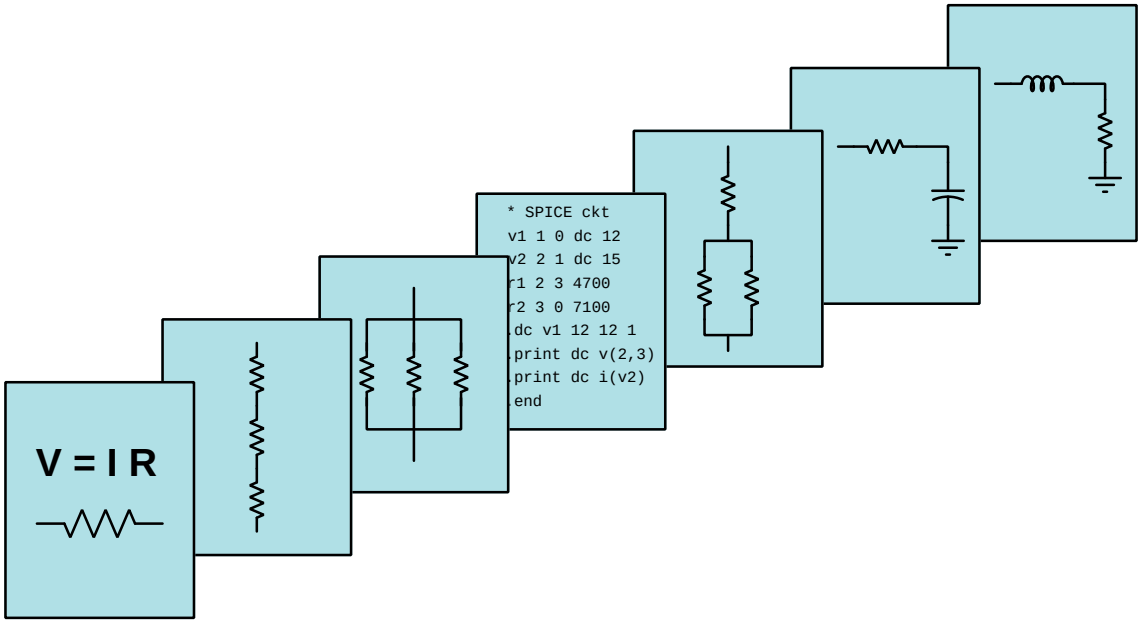
Modular Electronics Learning (ModEL) project



AC Series Circuits

Last update = 9 September 2024

Contents

Chapter 1

Introduction

1.1 Recommendations for students

The fundamental distinction between an *AC* circuit and a *DC* circuit is that voltages and currents in an AC circuit *alternate* polarity and direction over time rather than remain steady as is the case in a DC circuit. The rate at which an AC quantity alternates is called *frequency*. While DC is conceptually simpler, AC is useful for many more applications such as electrically representing vibrations (e.g. sound waves represented as AC voltages or currents in an audio recording or playback system) as well as electric power transmission over long distances.

Due to the fact that AC quantities persistently change and are not steady like DC, analysis of AC circuits is necessarily more complicated than DC circuits. However, it is possible to represent AC voltages, currents, and impedances in the form of complex numbers, and doing so permits all the fundamental properties of DC circuits to be applied to these AC quantities as well. With regard to series AC networks, voltages still add in series, currents are still equal in series, and impedances still add in series. Ohm's Law and Kirchhoff's Laws equations look very much the same, the only difference with AC being that each variable represents a complex number rather than a simple number.

Important concepts related to AC series networks include **DC** versus **AC** electricity, **phasors**, **frequency**, **phase shift**, **apparent power**, **true power**, **reactive power**, **Kirchhoff's Voltage Law**, **Kirchhoff's Current Law**, properties of **series** networks, **sources** versus **loads**, complex numbers in **polar** and **rectangular** forms, **Conservation of Energy**, **Conservation of Charge**, diagram **annotation**, and **polarity**.

A very important problem-solving strategy applied throughout the Tutorials is that of *annotating diagrams* with labels showing voltages, polarities, and currents (with direction). Mapping calculated values onto these diagrams is a helpful way to avoid confusion and to maintain proper context for all the quantities. Another important point about problem-solving is the need for *patience*, and the willingness to proceed with calculations even if the final strategy for solving the problem eludes one's immediate grasp.

When reading any mathematically-based presentation, a useful habit for effective learning is to actually perform the mathematics being shown in the text. Don't just passively read what the

text tells you and trust that the math works – try the math for yourself. Not only will this serve to confirm what you are reading, but it is also an excellent way to practice those mathematical techniques.

Here are some good questions to ask of yourself while studying this subject:

- How might an experiment be designed and conducted to demonstrate the properties of series electrical networks? What hypotheses (i.e. predictions) might you pose for that experiment, and what result(s) would either support or disprove those hypotheses?
- What does it mean to say that an AC quantity has a *frequency*?
- How does true power differ from apparent or reactive power in an AC circuit?
- Which principles from DC circuit analysis still apply to AC circuits?
- Which principles from DC circuit analysis do not apply to AC circuits?
- Why do we say that the phase angle of an inductor's impedance is positive 90 degrees?
- Why do we say that the phase angle of a capacitor's impedance is negative 90 degrees?
- What defines a *series* connection between two or more components?
- How may we explain all the properties of series networks in terms of more fundamental principles?
- How do we analyze a series network step-by-step?
- What is a "phasor diagram" and what does it show us?
- Why is it a good practice to store all calculated values in memory, rather than re-type those values into your calculator when needed for other calculations?
- What are some good ways to check our mathematical work when finishing a complex circuit-analysis problem?

1.2 Challenging concepts related to series AC circuits

The following list cites concepts related to this module's topic that are easily misunderstood, along with suggestions for properly understanding them:

- **Defining a series network** – series networks are defined by having a single path for current between the components. From this definition, and from the Conservation of Electric Charge, follows the conclusion that series-connected components must experience the same amount of continuous current. One challenge is that sometimes series-connected components do not physically or pictorially align to suggest a single path, even though they have but one *electrical* path for current. Another challenge is reasoning logically from this definition of "one path" to the necessary conclusion of "same current", as many students are so accustomed to memorizing facts without seeing the logical connections between those facts.

- **Phasors representing AC amplitudes and phase shifts** – a powerful tool used for understanding the operation of AC circuits is the *phasor diagram*, consisting of arrows pointing in different directions: the length of each arrow representing the amplitude of some AC quantity (voltage, current, or impedance), and the angle of each arrow representing the shift in phase relative to the other arrows. By representing each AC quantity thusly, we may more easily calculate their relationships to one another, with the phasors showing us how to apply trigonometry (Pythagorean Theorem, sine, cosine, and tangent functions) to the various calculations. An analytical parallel to the graphic tool of phasor diagrams is *complex numbers*, where we represent each phasor (arrow) by a pair of numbers: either a magnitude and angle (polar notation), or by "real" and "imaginary" magnitudes (rectangular notation). Where phasor diagrams are helpful is in applications where their respective AC quantities *add*: the resultant of two or more phasors stacked tip-to-tail being the mathematical sum of the phasors. Complex numbers, on the other hand, may be added, subtracted, multiplied, and divided; the last two operations being difficult to graphically represent with arrows.

- **Complex numbers in calculators** – while the ability of certain scientific calculators to perform complex-number arithmetic is an enormously helpful tool for students first learning to analyze AC circuits, some of these calculators prove to be finicky in their handling and entry of these quantities. Advice proven to be sound for all complex-number calculators is to save each and every complex-valued quantity into a memory location and then perform arithmetic operations on those stored variables rather than enter the complex numbers directly into the computation. For example, storing `3 - j4` into memory location `A` and `25` $\angle$ `30`^{o} into memory location `B`, then multiplying `A` $\times$ `B` rather than entering `3 - j4` $\times$ `25` $\angle$ `30`^{o}. Storing values into calculator memory and then retrieving them as needed for calculations is actually sound advice for many reasons, but many students resist taking these "extra" steps and as a result incur all the risks of hand-entering values (e.g. rounding errors due to truncating, keystroke errors when the same value must be used more than once, crowded displays where you cannot see the whole calculation, order-of-operations errors when complex numbers aren't enclosed in parentheses, etc.). Complex-number calculations reward good practices through consistently good results!

- **Complex numbers in measurement** – complex numbers may be expressed in either *rectangular* or *polar* form, either one of these being perfectly valid. However, measurement

instruments such as multimeters only provide the magnitude of the polar form of the voltage or current in question. For example, if a component's voltage is 4.8 Volts RMS $\angle$ 35^o with the circuit's source voltage being the phase reference (0^o), an voltmeter reading that voltage will simply register 4.8 Volts RMS. An oscilloscope simultaneously measuring that component voltage on one channel and the source voltage on another will show the 35^o shift on the horizontal axis between the two waveforms.

- **Resistance versus Reactance versus Impedance** – these three terms represent different forms of opposition to electric current. Despite the fact that they are measured in the same unit (ohms: Ω), they are not the same concept. Resistance is best thought of as electrical *friction*, whereas reactance is best thought of as electrical *inertia*. Whereas resistance creates a voltage drop by dissipating energy, reactance creates a voltage drop by *storing* and *releasing* energy. Impedance is a term encompassing both resistance and reactance, usually a combination of both.

The *Case Tutorial* chapter contains several worked problems for series AC circuits, showing solutions using complex numbers as well as solutions using scalar calculations.

1.3 Recommendations for instructors

This section lists realistic student learning outcomes supported by the content of the module as well as suggested means of assessing (measuring) student learning. The outcomes state what learners should be able to do, and the assessments are specific challenges to prove students have learned.

- **Outcome** – Demonstrate effective technical reading and writing

  Assessment – Students present their outlines of this module's instructional chapters (e.g. Case Tutorial, Tutorial, Historical References, etc.) ideally as an entry to a larger Journal document chronicling their learning. These outlines should exhibit good-faith effort at summarizing major concepts explained in the text.

  Assessment – Students show how quantitative results were obtained by the author in the Tutorial chapter's examples.

- **Outcome** – Apply the concepts of reactance and impedance to series AC circuits

  Assessment – Calculate all component voltages and currents in a series AC circuit given component values.

  Assessment – Sketch phasor diagrams of all component voltages and currents in a series AC circuit given component values.

- **Outcome** – Apply the concept of power factor to realistic circuits

  Assessment – Calculate the various powers (P, Q, S) and/or power factor for a series AC circuit given component values.

- **Outcome** – Independent research

  Assessment – Read and summarize in your own words reliable historical documents on the subject of applying complex numbers to AC circuit calculations. Recommended readings include books written by Charles Proteus Steinmetz.

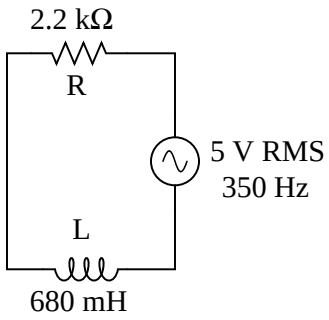
Chapter 2

Case Tutorial

The idea behind a *Case Tutorial* is to explore new concepts by way of example. In this chapter you will read less presentation of theory compared to other Tutorial chapters, but by close observation and comparison of the given examples be able to discern patterns and principles much the same way as a scientific experimenter. Hopefully you will find these cases illuminating, and a good supplement to text-based tutorials.

These examples also serve well as challenges following your reading of the other Tutorial(s) in this module – can you explain *why* the circuits behave as they do?

2.1 Example: series RL circuit



Scalar calculations

$R = 2.2\ \text{k}\Omega \qquad X_L = 2\pi fL = 1.495\ \text{k}\Omega$

$Z_{series} = \sqrt{R^2 + {X_L}^2}$
$Z_{series} = \sqrt{2200^2 + 1495^2} = 2660\,\Omega$

$I = \frac{V}{Z_{series}} = \frac{5}{2660} = 1.8796\ \text{mA}$

$V_R = IR = (1.8796 \times 10^{-3})(2200) = 4.1352\ \text{V}$
$V_L = IX_L = (1.8796 \times 10^{-3})(1495) = 2.8108\ \text{V}$

Phasor calculations

$R = 2.2\ \text{k}\Omega \qquad X_L = 2\pi fL = 1.495\ \text{k}\Omega$

$Z_R = 2.2\ \text{k}\Omega \angle\, 0^o \qquad Z_L = 1.495\ \text{k}\Omega \angle\, 90^o$ (Polar form)
$Z_R = 2.2\ \text{k}\Omega + j0\,\Omega \qquad Z_L = 0\,\Omega + j1.495\ \text{k}\Omega$ (Rectangular form)

$Z_{series} = Z_1 + Z_2 + \cdots Z_n$ (General rule of series impedances)
$Z_{series} = Z_R + Z_L$ (Specific application to this circuit)

$Z_{series} = 2.2\ \text{k}\Omega \angle\, 0^o + 1.495\ \text{k}\Omega \angle\, 90^o = 2.66\ \text{k}\Omega \angle\, 34.2^o$
$Z_{series} = (2.2\ \text{k}\Omega + j0\,\Omega) + (0\,\Omega + j1.495\ \text{k}\Omega) = 2.2\ \text{k}\Omega + j1.495\ \text{k}\Omega$

$I = \frac{V}{Z_{series}} = \frac{5 \angle 0^o}{2660 \angle 34.21^o} = 1.8796\ \text{mA} \angle -34.21^o$

$V_R = IZ_R = (1.8796 \times 10^{-3} \angle -34.21^o)(2200 \angle 0^o) = 4.1352\ \text{V} \angle -34.21^o = 3.420 - j2.325\ \text{V}$
$V_L = IZ_L = (1.8796 \times 10^{-3} \angle -34.21^o)(1495 \angle 90^o) = 2.8108\ \text{V} \angle 55.80^o = 1.580 + j2.325\ \text{V}$

The Python programming language is *interpreted* in nature, which means any command entered into a Python interpreter application is immediately executed by the interpreter. Like most modern programming languages, Python offers extensive libraries of mathematical functions, making it useful as a powerful scientific calculator. Here we see Python being used to perform all the previously-shown scalar calculations.

Every line of text beginning with the prompt (>>>) is entered by the user. Every line of text missing this prompt is displaying a result computed by the Python interpreter.

Using Python to perform scalar calculations

```
Python 3.7.2 (default, Feb 19 2019, 18:15:18)
[GCC 4.1.2] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from math import *
>>> r = 2200
>>> xl = 2*pi*350*680e-3
>>> xl
1495.3981031087415
>>> zseries = sqrt(r**2 + xl**2)
>>> zseries
2660.1156904881454
>>> i = 5/zseries
>>> i
0.0018796174985466416
>>> i * r
4.135158496802611
>>> i * xl
2.810776441896645
```

Python's mathematics capabilities include complex numbers, and the "complex math" (`cmath`) library provides certain functions useful for manipulating complex numbers (such as the "phase" function). Here we see Python being used as a scientific calculator to perform all the previously-shown phasor calculations.

Note the use of the semicolon character (`;`) to instruct Python to execute multiple commands in one entry, used here to show polar magnitude and polar phase angles for certain variables.

Using Python to perform phasor calculations

```
Python 3.7.2 (default, Feb 19 2019, 18:15:18)
[GCC 4.1.2] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from math import *
>>> from cmath import *
>>> r = 2200
>>> xl = 2*pi*350*680e-3
>>> xl
1495.3981031087415
>>> zl = complex(0,xl)
>>> zl
1495.3981031087415j
>>> zseries = r + zl
>>> zseries
(2200+1495.3981031087415j)
>>> i = 5 / zseries
>>> i
(0.001554503253970803-0.0010566369169383205j)
>>> abs(i) ; degrees(phase(i))
0.0018796174985466416
-34.20498172920196
>>> vr = i * r
>>> vl = i * zl
>>> vr
(3.4199071587357666-2.324601217264305j)
>>> vl
(1.5800928412642332+2.324601217264305j)
>>> abs(vr) ; degrees(phase(vr))
4.135158496802611
-34.204981729201954
>>> abs(vl) ; degrees(phase(vl))
2.8107764418966457
55.79501827079805
```

Here we see the SPICE circuit simulation program being used to simulate this simple RL circuit.

SPICE netlist

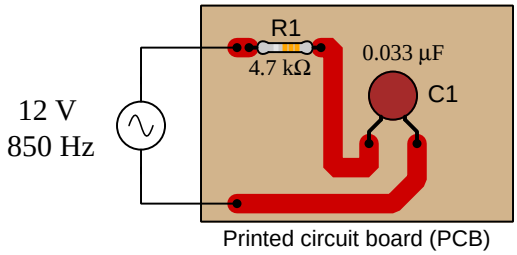
```
* Series RL circuit
v1 1 0 ac 5
r1 1 2 2200
l1 2 0 680e-3
.control
set units=degrees
.endc
.ac lin 1 350 350
.print ac vm(1,2) vp(1,2)
.print ac vm(2) vp(2)
.end
```

NGSPICE simulation results

```
--------------------------------------------------------------------------------
Index   frequency       mag(v(1)-v(2))  ph(v(1)-v(2))
--------------------------------------------------------------------------------
0       3.500000e+02    4.135158e+00   -3.42050e+01

--------------------------------------------------------------------------------
Index   frequency       vm(2)           vp(2)
--------------------------------------------------------------------------------
0       3.500000e+02    2.810776e+00    5.579502e+01
```

2.2 Example: series RC circuit



Scalar calculations

$R = 4.7\ \text{k}\Omega \qquad X_C = \frac{1}{2\pi f C} = 5.674\ \text{k}\Omega$

$Z_{series} = \sqrt{R^2 + {X_C}^2}$
$Z_{series} = \sqrt{4700^2 + 5674^2} = 7368\ \Omega$

$I = \frac{V}{Z_{series}} = \frac{12}{7368} = 1.6287\ \text{mA}$

$V_R = IR = (1.6287 \times 10^{-3})(4700) = 7.6550\ \text{V}$
$V_L = IX_L = (1.6287 \times 10^{-3})(5674) = 9.2413\ \text{V}$

Phasor calculations

$R = 4.7\ \text{k}\Omega \qquad X_C = \frac{1}{2\pi f C} = 5.674\ \text{k}\Omega$

$Z_R = 4.7\ \text{k}\Omega \angle\ 0^o \qquad Z_C = 5.674\ \text{k}\Omega \angle\ -90^o \quad \text{(Polar form)}$
$Z_R = 4.7\ \text{k}\Omega + j0\ \Omega \qquad Z_C = 0\ \Omega - j5.674\ \text{k}\Omega \quad \text{(Rectangular form)}$

$Z_{series} = Z_1 + Z_2 + \cdots Z_n \quad \text{(General rule of series impedances)}$
$Z_{series} = Z_R + Z_C \quad \text{(Specific application to this circuit)}$

$Z_{series} = 4.7\ \text{k}\Omega \angle\ 0^o + 5.674\ \text{k}\Omega \angle\ -90^o = 7.368\ \text{k}\Omega \angle\ -50.36^o$
$Z_{series} = (4.7\ \text{k}\Omega + j0\ \Omega) + (0\ \Omega - j5.674\ \text{k}\Omega) = 4.7\ \text{k}\Omega - j5.674\ \text{k}\Omega$

$I = \frac{V}{Z_{series}} = \frac{12 \angle 0^o}{7368 \angle -50.36^o} = 1.6287\ \text{mA}\ \angle 50.36^o$

$V_R = IZ_R = (1.6287 \times 10^{-3} \angle 50.36^o)(4700 \angle 0^o) = 7.6550\ \text{V}\ \angle 50.36^o$
$V_C = IZ_C = (1.6287 \times 10^{-3} \angle 50.36^o)(5673 \angle -90^o) = 9.2413\ \text{V}\ \angle -39.64^o$

SPICE netlist

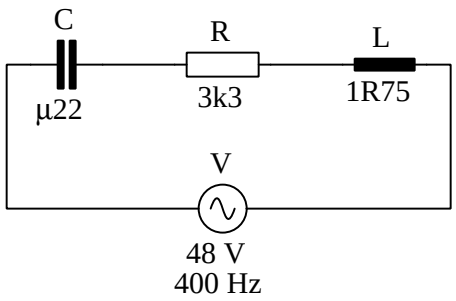
```
* Series RC circuit
v1 1 0 ac 12
r1 1 2 4700
c1 2 0 0.033e-6
.control
set units=degrees
.endc
.ac lin 1 850 850
.print ac vm(1,2) vp(1,2)
.print ac vm(2) vp(2)
.end
```

NGSPICE simulation results

```
--------------------------------------------------------------------------------
Index   frequency       mag(v(1)-v(2))  ph(v(1)-v(2))
--------------------------------------------------------------------------------
0       8.500000e+02    7.654967e+00    5.036356e+01

--------------------------------------------------------------------------------
Index   frequency       vm(2)           vp(2)
--------------------------------------------------------------------------------
0       8.500000e+02    9.241292e+00   -3.96364e+01
```

2.3 Example: series RLC circuit



Scalar calculations

$R = 3.3\ \text{k}\Omega \qquad X_L = 2\pi fL = 4.3982\ \text{k}\Omega \qquad X_C = \frac{1}{2\pi fC} = 1.8086\ \text{k}\Omega$

$Z_{series} = \sqrt{R^2 + (X_L - X_C)^2}$
$Z_{series} = \sqrt{3300^2 + (4398.2 - 1808.6)^2} = 4194.8\ \Omega$

$I = \frac{V}{Z_{series}} = \frac{48}{4194.8} = 11.443\ \text{mA}$

$V_R = IR = (11.443 \times 10^{-3})(3300) = 37.761\ \text{V}$
$V_L = IX_L = (11.443 \times 10^{-3})(4398.2) = 50.328\ \text{V}$
$V_C = IX_C = (11.443 \times 10^{-3})(1808.6) = 20.695\ \text{V}$

Phasor calculations

$R = 3.3\ \text{k}\Omega \qquad X_L = 2\pi fL = 4.3982\ \text{k}\Omega \qquad X_C = \frac{1}{2\pi fC} = 1.8086\ \text{k}\Omega$

$Z_R = 3.3\ \text{k}\Omega \angle\ 0^o \qquad Z_L = 4.3982\ \text{k}\Omega \angle\ 90^o \qquad Z_C = 1.8086\ \text{k}\Omega \angle\ -90^o$ (Polar form)
$Z_R = 3.3\ \text{k}\Omega + j0\ \Omega \qquad Z_L = 0\ \Omega + j4.3982\ \text{k}\Omega \qquad Z_C = 0\ \Omega - j1.8086\ \text{k}\Omega$ (Rectangular form)

$Z_{series} = Z_1 + Z_2 + \cdots Z_n$ (General rule of series impedances)
$Z_{series} = Z_R + Z_L + Z_C$ (Specific application to this circuit)

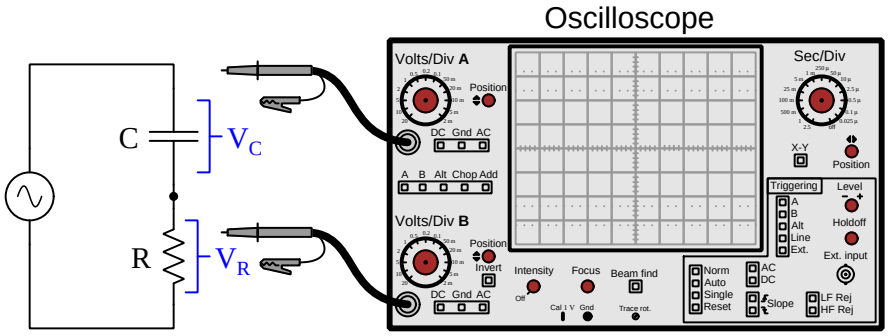
$Z_{series} = (3.3\ \text{k}\Omega \angle\ 0^o) + (4.3982\ \text{k}\Omega \angle 90^o) + (1.8086\ \text{k}\Omega \angle\ -90^o) = 4.1948\ \text{k}\Omega \angle\ 38.123^o$
$Z_{series} = (3.3\ \text{k}\Omega + j0\ \Omega) + (0\ \Omega + j4.3982\ \text{k}\Omega) + (0\ \Omega - j1.8086\ \text{k}\Omega) = 3.3\ \text{k}\Omega + j2.5897\ \text{k}\Omega$

$I = \frac{V}{Z_{series}} = \frac{48 \angle 0^o}{4194.8 \angle 33.123^o} = 11.443\ \text{mA}\ \angle -38.123^o$

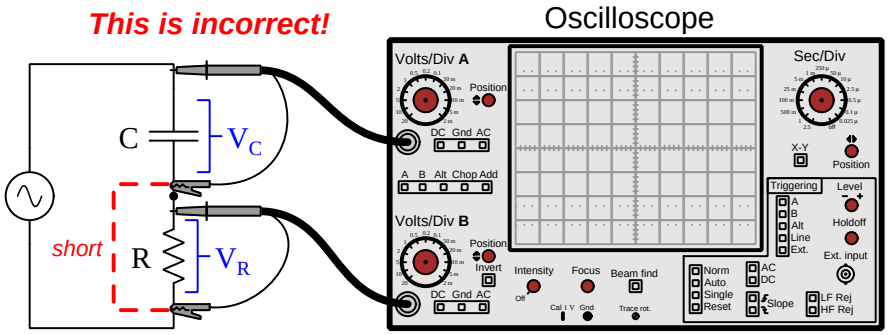
$V_R = IZ_R = (11.443 \times 10^{-3} \angle -38.123^o)(3300 \angle 0^o) = 37.761\ \text{V}\ \angle -38.123^o$
$V_L = IZ_L = (11.443 \times 10^{-3} \angle -38.123^o)(4398.2 \angle 90^o) = 50.328\ \text{V}\ \angle 51.877^o$
$V_C = IZ_C = (11.443 \times 10^{-3} \angle -38.123^o)(1808.6 \angle -90^o) = 20.695\ \text{V}\ \angle -128.12^o$

2.4 Example: measuring series voltage phase shift

Suppose we wished to measure the resistor and capacitor voltages in this AC circuit using an oscilloscope, comparing those two AC waveforms on the oscilloscope's screen to measure the phase shift between them:

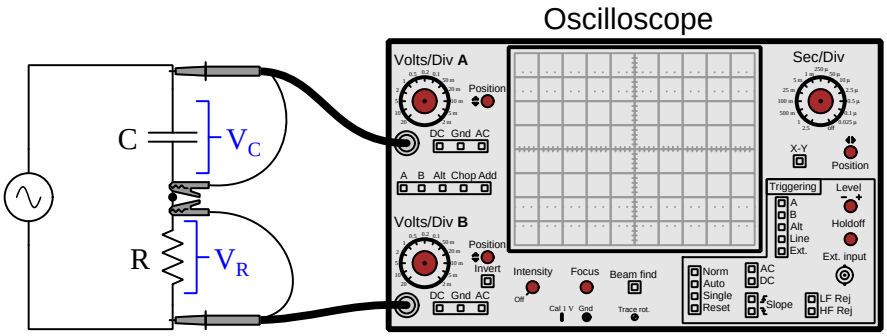


How should we connect the oscilloscope's probes across each of the components to perform this test? At first it may seem like this would be the best way, connecting each probe similarly across the two components:

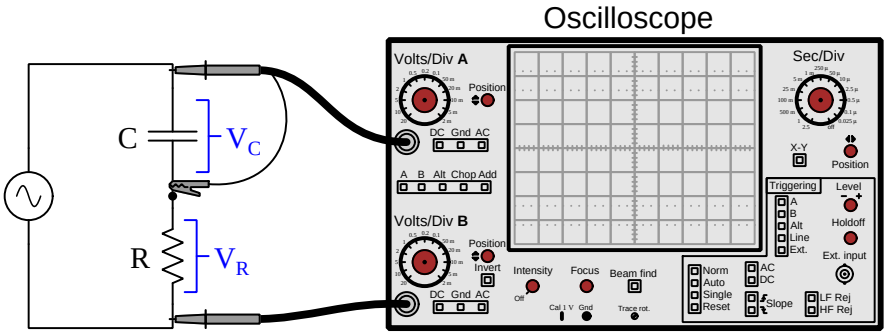


However, the fact that both of the oscilloscope's input channels share a common ground (chassis) connection means that there will be a short-circuit imposed between the two ground clips in this circuit, effectively bypassing the resistor!

When using ordinary oscilloscope probes sharing a common ground connection, we must ensure both channels reference the same point in the circuit under test. This means there is only one electrically valid way to connect these two channels to measure both components' voltage drop:

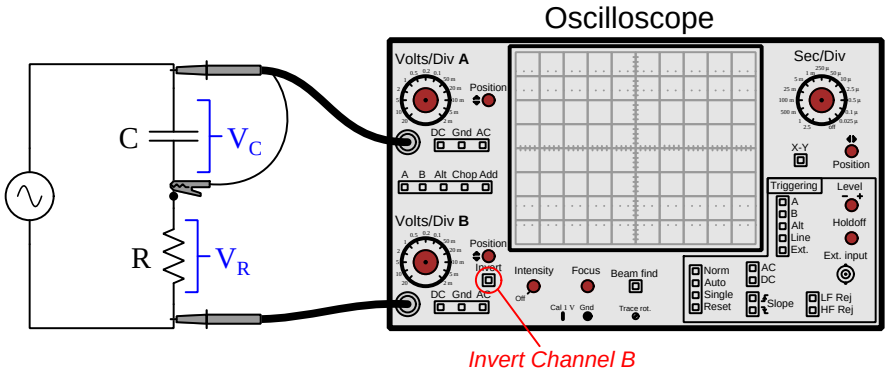


Given that the two channels' ground clips are electrically common to each other, we can actually eliminate either one of them and still have the same result:



The only problem with this scheme is that we have inadvertently *inverted* these two channels with respect to one another. This is easy to see if we imagine this circuit being a simple voltage divider with two resistors (no capacitor) at an instant in time where the AC source is positive on top and negative on bottom: with reference to ground (the mid-point between the two components) channel A's probe tip will sense a positive potential while channel B's probe tip will sense a negative, even though we the voltages dropped by a pair of series-connected resistors should be exactly in-phase with one another. What the oscilloscope would show on its display for a two-resistor AC circuit is *two waveforms 180 degrees out of phase with each other!* For the resistor-capacitor circuit in our example, the resistor's voltage waveform would have an extra 180^o of phase shift, confusing our attempt to measure the true phase shift.

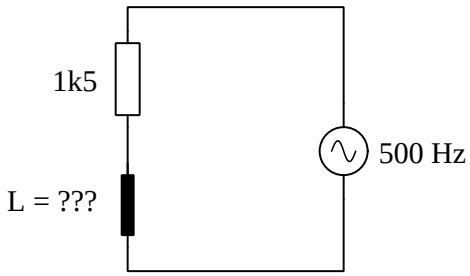
Oscilloscopes typically provide a feature to compensate for this accidental inversion, appropriately named *invert*. On the oscilloscope shown in this illustration we see Channel B's "invert" feature activated by a button:



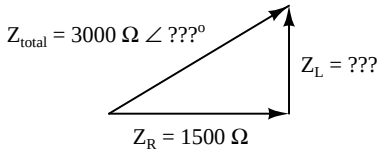
By intentionally inverting Channel B's displayed waveform, we "un-do" the 180^o inversion imposed by the necessary probe connection with respect to the oscilloscope's common ground. With this feature engaged, we may correctly interpret the oscilloscope's display to discern the true phase shift between these two components' voltages.

2.5 Example: solving for an unknown inductance

Suppose we wish this series RL circuit to have a total impedance of 3 kΩ at a power supply frequency of 500 Hz:



This is an example of a problem that cannot be solved simply by using a calculator with complex-number capability and applying the standard series network principle that total impedance is equal to the sum of the individual component impedances, because we do not know what the phase angle of the total impedance will be when its magnitude is 3000 Ohms. In other words, there is information missing that would be necessary for a complex-arithmetic solution. Therefore, we must resort to phasor diagrams and trigonometry, treating the phasor magnitudes and angles separately:



$$Z_{total} = \sqrt{Z_R^2 + Z_L^2}$$

$$Z_{total}^2 = Z_R^2 + Z_L^2$$

$$Z_L^2 = Z_{total}^2 - Z_R^2$$

$$Z_L = \sqrt{Z_{total}^2 - Z_R^2}$$

$$Z_L = \sqrt{3000^2 - 1500^2}$$

$$Z_L = 2598.08\,\Omega$$

Assuming a pure inductance (i.e. no parasitic resistance or capacitance within its windings), the magnitude of its impedance (Z_L) will be equal to its reactance (X_L). Thus,

$$Z_L = X_L = 2\pi f L$$

$$L = \frac{X_L}{2\pi f}$$

$$L = \frac{2598.08}{2\pi 500}$$

$$L = 0.82699 \text{ H}$$

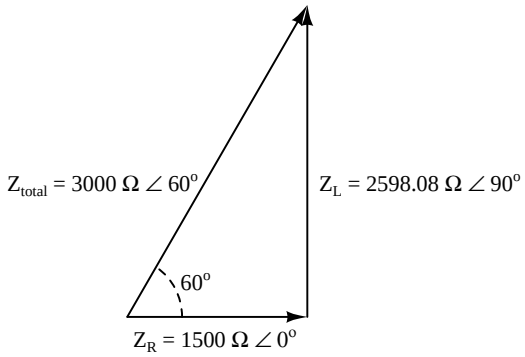
Incidentally, the total impedance phase angle may be easily calculated at this point using the arc-cosine function, relating phase angle to the adjacent (Z_R) and hypotenuse (Z_{total}) side-lengths of this right triangle:

$$\theta = \arccos \frac{Z_R}{Z_{total}}$$

$$\theta = \arccos \frac{1500}{2598.08}$$

$$\theta = 60^o$$

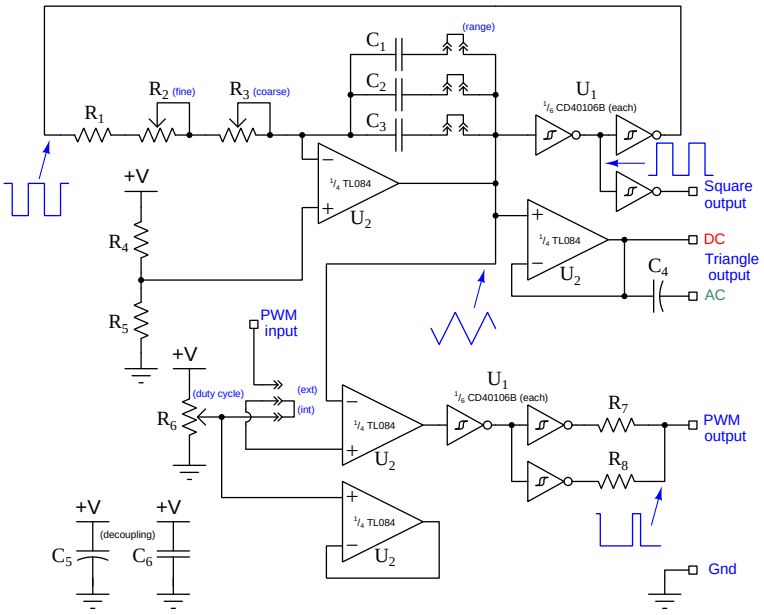
Therefore, Z_{total} has a complex-number value of 3000 Ohms at 60 degrees, which means our original phasor diagram was not drawn to scale, and will actually look like this instead:

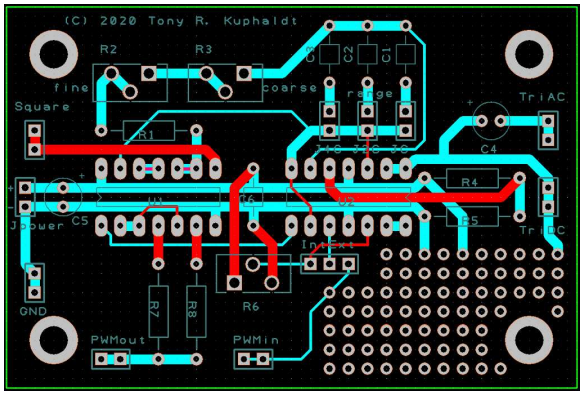


2.6 Example: sine versus non-sine AC sources

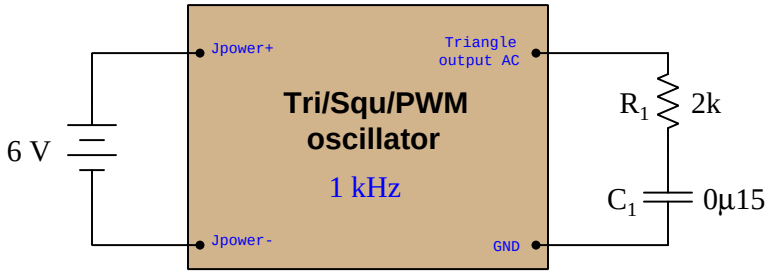
Students learning to analyze RLC networks powered by AC voltage sources typically rely on expensive signal generators to produce the pure sine-wave AC excitation voltage necessary for voltmeter measurements to closely match predictions. However, robust triangle-wave oscillator circuits are much less complicated to design and build than sine-wave oscillator circuits, so if students wish to build their own signal generators for these introductory AC experiments it is good to know that triangle-wave excitation yields results very close to sine-wave excitation.

A simple and versatile signal generator circuit appears below, outputting triangle, square, and PWM (pulse-width-modulated) signals. The first image is the schematic diagram, followed by a PCB layout:





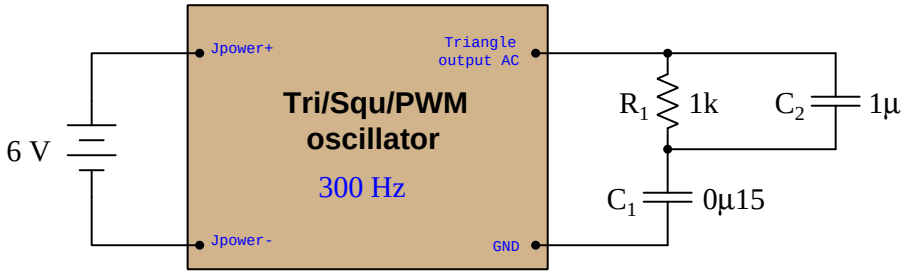
Here are some test results on simple RC networks:



Connected to the RC network, frequency was measured using a Fluke model 87-III multimeter and adjusted to 1 kHz, and then total voltage measured across the series $R_1 \leftrightarrow C_1$ combination as 233.0 mVAC.

Parameter	**Measured** (triangle-wave)	**Predicted** (sine-wave)
V_{R1}	205.7 mVAC	205.8 mVAC
V_{C1}	109.3 mVAC	109.2 mVAC

Testing a slightly more complex circuit at a frequency of 300 Hz, the loaded voltage output of the oscillator being 231.6 mVAC this time:



Parameter	**Measured** (triangle-wave)	**Predicted** (sine-wave)
V_{R1}	23.7 mVAC	27.43 mVAC
V_{C1}	207.5 mVAC	207.0 mVAC
V_{C2}	23.7 mVAC	27.43 mVAC

In both applications, the greatest error between measured voltage and predicted voltage as a percentage of total voltage was in the second circuit across $R_1||C_1$ (23.7 milliVolts rather than 27.43 milliVolts), and this is only -1.61% of the source voltage which is considerably less than the $\pm$ 5% tolerance of the resistor and capacitors!

If we compare the Fourier series for a sine wave and a triangle wave (both having unity peak values and a frequency of ω) we see that the first harmonic of the triangle wave function is identical to the sine wave, and that all the other harmonics in the triangle wave are significantly smaller-amplitude than the fundamental:

Sine wave

$$\cos \omega t$$

Triangle wave

$$\cos \omega t + \frac{1}{9} \cos 3\omega t + \frac{1}{25} \cos 5\omega t + \frac{1}{49} \cos 7\omega t + \cdots + \frac{1}{n^2} \cos n\omega t$$

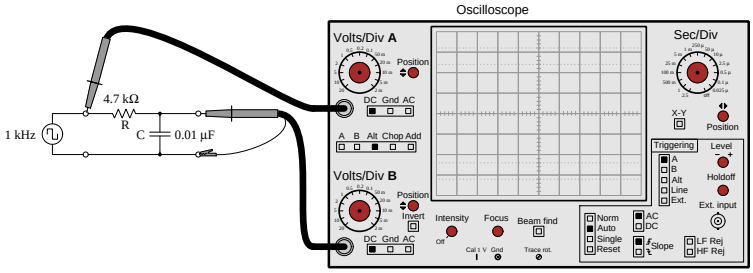
This tells us any deviations between the measured (triangle-wave) and predicted (sine-wave) voltage values are likely to be minimal, the third harmonic being only 11.1% of the fundamental's amplitude, the fifth harmonic being only 4% of the fundamental's amplitude, etc. The effects of higher-order harmonics are truly negligible due to their vastly smaller amplitudes as well as due to the fact that most digital multimeters suffer "cut off" in the audio-frequency range and therefore cannot measure signal components in the tens of thousands of Hertz.

With access to a digital oscilloscope having FFT capability[1] to show precise voltage values for each harmonic of a measured waveform, we have an even better solution for obtaining voltage measurements in agreement with predicted values when not using perfectly sinusoidal signal generators. Since the oscilloscope's FFT algorithm separates and displays each of the sinusoidal harmonics apart from one another in any non-sinusoidal waveform, if we simply pay attention to the magnitudes of a common harmonic frequency within each voltage measurement we will essentially take circuit measurements on purely sinusoidal voltages of the same frequency. For example, we could measure the fundamental (i.e. the first harmonic)[2] amplitude of source voltage, then the fundamental amplitudes of each of the other components' voltages, and check to see that these measured voltage values match well with our predictions at that frequency. This technique, in effect, lets us measure the effects of a purely sinusoidal signal even when the real signal is not sinusoidal at all, by taking measurements only on a common harmonic of the measured voltages!

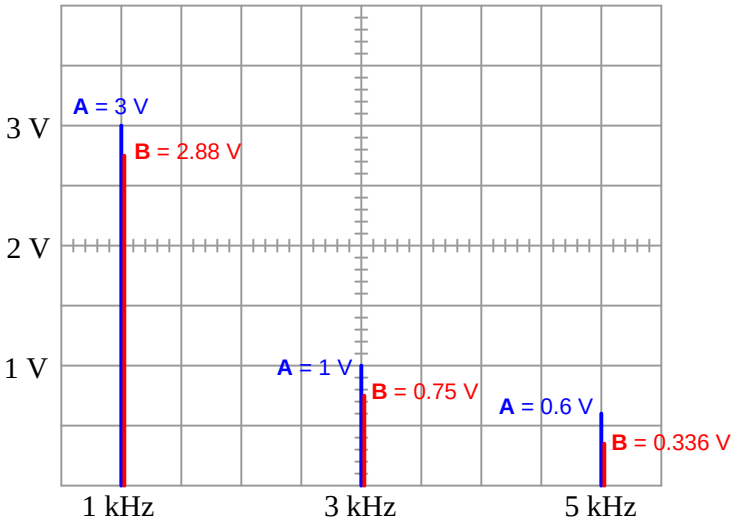
[1] At the time of this writing (2022) some inexpensive oscilloscopes may be found with rather poor FFT resolution, resulting in wide spectral peaks with uncertain height (voltage) values. You know you are working with a sufficiently precise instrument when the harmonic peaks show as thin lines rather than exaggerated bell-curves.

[2] There is no particular reason why we might choose the first harmonic over any of the others, other than the fact that with triangle and square waves this fundamental will be vastly stronger than any of the other harmonics.

This testing technique deserves some elaboration, and so we shall explore it by example. Consider the following test circuit where a signal generator configured to output a square-wave AC signal at 1000 Hz energizes a simple RC network consisting of a 4.7 kΩ resistor and a 0.01 μF capacitor:

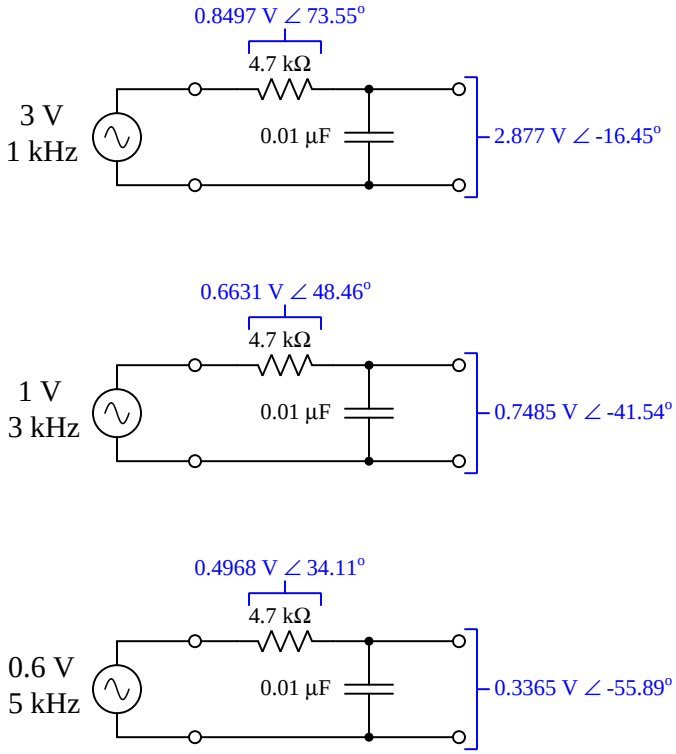


We know from Fourier analysis that a square wave is actually equivalent to a sine wave at the same fundamental frequency added to another sine wave one-third the amplitude at three times that frequency (3rd harmonic) added to another sine wave one-fifth the amplitude of the fundamental at five times that frequency (5th harmonic), and so on. If we examine the frequency-domain plots of the signal generator's output (channel A) versus the capacitor's voltage drop (channel B), we see the circuit's response to pure sine waves at each of those frequencies:



The relative peak heights of the channel A signal (3 Volts, 1 Volt, 0.6 Volts) are simply the result of the Fourier series for a square wave and has nothing to do with the RC network. The ratios between peak heights of channel A and channel B at each harmonic frequency, however, are unique to the 4.7 kΩ and 0.01 μF RC network because those voltage pairs represent the attenuation of this particular network at each of those sinusoidal frequencies.

If we mathematically analyze this same RC network for each of the square wave's harmonic amplitudes and frequencies used in the test circuit, we should obtain results verifiable by using the oscilloscope in FFT mode:



For students with access to oscilloscopes having fine-resolution FFT capability, this not only means it is unnecessary to secure a signal generator with pure sinusoidal output, but it also means the ability to energize any AC circuit with *any* waveshape and test its response at multiple sinusoidal frequencies *simultaneously*!

Chapter 3

Tutorial

3.1 AC versus DC

The study of electricity always begins with *Direct Current* (DC) because circuits where current direction and voltage polarity remain constant are conceptually easier to grasp than circuits where current direction and voltage polarity switch over time (i.e. *Alternating Current* or AC circuits). However, AC circuits are every bit as prevalent as DC, and so no study of electricity or electronics would be complete without a thorough exploration of alternating current.

Just a few principal differences exist between DC and AC circuits from the perspective of analysis:

- The magnitudes of all AC voltage and currents must be carefully quantified, because unlike the steady voltages and currents of DC these quantities continually change over time. One way to do this is to express an AC quantity by its *peak* value over time: its greatest magnitude over all the points in its cycle. Another way is to regard its *peak-to-peak* value. Perhaps the most common expression is *RMS* (Root-Mean-Square), which refers to the equivalent DC magnitude that would deliver the same amount of power to a load.
- With AC we have a new quantity called *frequency* (f): the rate at which voltages and currents complete their alternating cycles. Frequency is typically expressed in the unit of *Hertz* (Hz), equivalent to cycles per second. Alternatively, frequency may be expressed as an angular velocity in the unit of circular radians per second, symbolized as ω.
- With AC we have another new quantity called *phase shift*: the degree to which two alternating voltages' or currents' cycles are out of step with each other in time. Phase shift is typically expressed in the unit of *degrees*, there being 360 degrees in one complete cycle.
- In order to mathematically account for phase shift in an AC circuit, every quantity needs to be quantified as a complex number called a *phasor*.
- Power takes three different forms in an AC circuit: apparent power (S), true power (P), and reactive power (Q). Apparent power is the product of voltage and current magnitudes, as given by a voltmeter and ammeter, measured in Volt-Amperes. True power is the actual rate of energy leaving the circuit, measured in Watts. Reactive power is the rate of energy exchanged losslessly by inductance and/or capacitance, measured in Volt-Amperes Reactive. Power factor is the unitless ratio of true power to apparent power, also given by the cosine of the circuit impedance phase angle.
- Certain components exist for AC circuits that have no equivalent in DC. *Transformers* are a good example of this, with their ability to "step up" and "step down" AC voltages and currents.

The list of similarities between AC and DC is thankfully longer:

- *Energy* is still that which is able to set matter into motion, and it is *always* conserved. This should come as no surprise, as energy and its conservation are fundamental features of our universe. Electric charges moving though electric fields, like masses moving through gravitational fields, are able to gain or lose potential energy in AC and DC circuits alike.
- *Voltage* (symbolized as V, measured in Volts) is still defined as the amount of potential energy gained or lost by electric charges moving from one location to another, and is always a relative quantity between two points. *Current* (symbolized as I, measured in Amperes) is still defined as the drift rate of electric charges past a given point.
- *Resistance* (symbolized as R, measured in Ohms) is still defined as that which causes energy to be dissipated from moving electric charges to leave the circuit entirely. A new quantity called *Reactance* (symbolized as X, also measured in Ohms) is defined as that which causes energy to be alternately extracted from and then returned to moving electric charges. *Impedance* (symbolized as Z, also measured in Ohms) is a general term encompassing resistance, reactance, or any combination of the two.
- *Sources* still infuse energy into passing electric charges, and *Loads* still extract energy from passing charges. Resistance always acts as a load, while reactance alternately acts as a source and a load within each AC cycle.
- *Ohm's Law* in DC circuits defines resistance as the ratio of voltage to current ($R = \frac{V}{I}$). The only difference with AC is that we substitute the more general concept of impedance for resistance, and so Ohm's Law becomes $Z = \frac{V}{I}$, and we represent all variables as complex numbers (phasors).
- *Kirchhoff's Voltage Law* still holds true for modest frequencies[1], in that an electric charge moved from one location to another until it arrives at its starting location will experience a net voltage (energy gain/loss) of zero. This means the phasor sum of all AC voltages in any loop must be zero.
- *Kirchhoff's Current Law* still holds true for modest frequencies[2], in that the flow of electric charges into any location must be balanced by an equal flow of charges out of that location. This means the phasor sum of all AC currents entering and exiting any point must be zero.
- *Series* networks still experience the same current through all components, a total voltage equal to the phasor sum of all component voltages, and a total impedance equal to the phasor sum of all component impedances.
- *Parallel* networks still experience the same voltage across all components, a total current equal to the phasor sum of all component currents, and a total impedance equal to the reciprocal of the phasor sum of all components' reciprocated impedances.

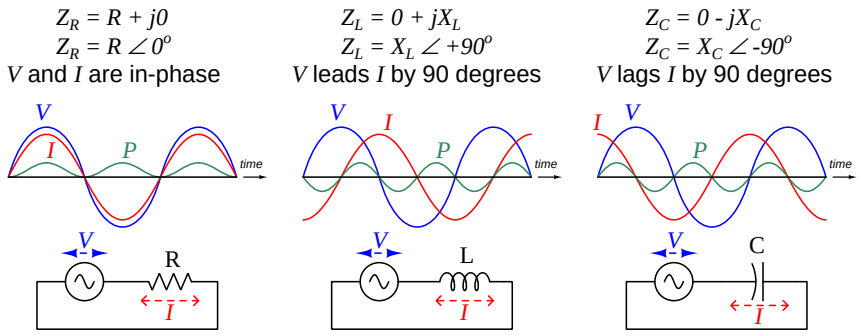
[1] At very high frequencies – high enough that the rate of alternation is faster than the electric field is able to propagate along the entire length of the circuit – electric charges cannot be conceived as existing within a steady electric field, and therefore one of the fundamental assumptions of Kirchhoff's Voltage Law is no longer true. In most applications the frequency must be in the megaHertz range before such effects become noticeable.

[2] At very high frequencies, typically megaHertz and beyond, the pulsations of electric charge flow become so rapid that it is possible for the direction of current to switch before the propagation reaches all portions of the circuit.

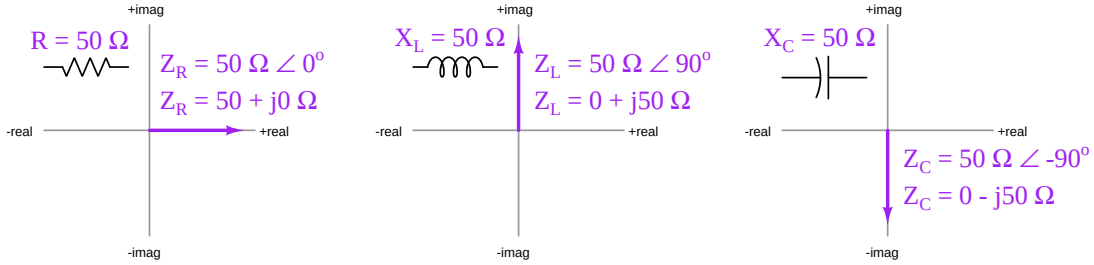
The major challenge facing students learning to analyze AC circuits is how to represent all numerical quantities as phasors, in order to make AC circuit analysis as similar to DC circuits as possible. For any voltage or current value in an AC circuit, the polar representation of the phasor will be the magnitude (usually RMS unless otherwise specified) and a phase angle expressing how far shifted that quantity's waveform is from some reference voltage or current at the same frequency. If an AC circuit is powered by a single voltage or current source, that source will typically serve as the phase reference (i.e. the source's phase angle will be zero degrees by default).

Phasor angles for impedances, by contrast, require no such reference[3]. The phase angle for an impedance represents the phase shift between that component's *voltage* versus that component's *current* and therefore its value intrinsic to that component.

For example, a pure resistance has an impedance with a phase angle of 0^o because zero phase shift exists between a resistance's voltage and current waveforms. A pure inductor's impedance is simply its reactance with a phase angle of $+90^o$ because an inductor's voltage leads its current by $+90^o$. Capacitive impedance is reactance with a phase angle of -90^o because a capacitor's voltage lags its current by -90^o:



Graphical expressions of complex-number resistance and reactance values are called *phasor diagrams*. Three such diagrams illustrate the difference between a 50 Ω resistor versus an inductor having 50 Ω of reactance and a capacitor also having 50 Ω of reactance:



[3]Technically, impedance is not a true *phasor* even though it is a complex-number quantity, because a phasor fundamentally represents a waveform. Impedance is really the ratio between two different waveforms and not a waveform unto itself. However, you will sometimes find the term *phasor* in reference to any AC circuit quantity in complex-number form.

Some examples[4] of component impedances are shown here:

- A 570 Ω resistor at any frequency will have an impedance of:

  570 Ω $\angle$ 0^o *(polar form)*

  570 + j0 Ω *(rectangular form)*

- A 3.5 H inductor at a frequency of 120 Hz will have an impedance of:

  2.639 kΩ $\angle$ $+90^o$ *(polar form)*

  0 + j2.639 kΩ *(rectangular form)*

- A 0.01 μF capacitor at a frequency of 3 kHz will have an impedance of:

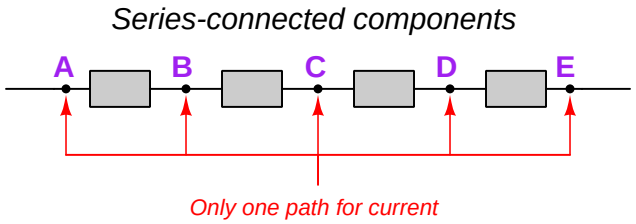
  5.305 kΩ $\angle$ -90^o *(polar form)*

  0 − j5.305 kΩ *(rectangular form)*

[4] Try calculating these impedance values from the given component values, to check your understanding of how reactances related to impedances. This is a good learning strategy to apply when reading any mathematical text: work through the presented examples on your own to see if you achieve the same results! Please note that when you apply either the $X_L = 2\pi fL$ formula or the $X_C = \frac{1}{2\pi fC}$ formula using your calculator to compute reactance, the result will *only* be a reactance value and not a (complex) impedance value. In order to attach the desired phase angle to your computed reactance value, you will have to perform the additional step of multiplying that reactance by a *unit phasor* which is nothing more than the quantity of 1 with the correct phase angle. For example, a capacitive reactance of 5.305 kΩ would be multiplied by 1 $\angle$ -90^o to yield a capacitive *impedance* of 5.305 kΩ $\angle$ -90^o.

3.2 Series network properties

Now that we have reviewed some fundamentals of AC electric circuits and drawn comparisons to DC circuits, let us move on to the topic at hand: *series AC circuits*. A "series" network, by definition, is one where all components are connected in-line with each other, such that they all share a single path for electric charge carriers to flow. The following illustration shows four components (each one represented as a nondescript rectangle) connected in series with each other, with lettered points for reference:



This definition of a series network naturally leads to a set of properties unique to this network type:

Definition: *Series*-connected electrical components provide exactly one path for current.

Property #1 Series-connected components experience the *same current* at any given time, due to the Conservation of Electric Charge (i.e. electric charges cannot disappear nor come into being, and so every charge entering one portion of a series network must eventually exit that portion). $I_1 = I_2 = I_3 \cdots = I_n$

Property #2 Voltages add in series: total voltage across a string of series-connected components is equal to the *phasor sum* of the components' voltages, due to the Conservation of Energy (i.e. the sum of all energy gains and losses must equal the total gain/loss). $V_{total} = V_1 + V_2 + V_3 \cdots + V_n$, which is graphically equivalent to stacking voltage phasors tip-to-tail in a phasor diagram.

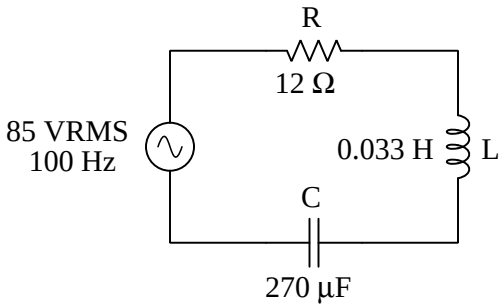
Property #3 Impedances add in series: total impedance for a string of series-connected impedances is equal to the *phasor sum* of those impedance values. $Z_{total} = Z_1 + Z_2 + Z_3 \cdots + Z_n$, which is graphically equivalent to stacking impedance phasors tip-to-tail in a phasor diagram.

These properties hold true for any series network because they are rooted in fundamental conservation laws. They apply for AC as well as DC circuits, and they apply for any types of components thusly connected. The major difference between these properties as applied to AC versus DC is that all calculations must be done using phasors in AC rather than simple numbers as in DC. If we use phasor diagrams to represent additive quantities in a series network (e.g. voltages,

impedances), we may stack those phasors tip-to-tail to graphically determine resultant phasors representing those sums.

3.3 Example circuit analysis using complex calculator

Let's analyze the following series circuit containing a resistor, inductor, and a capacitor all powered by an AC voltage source:



A good first step is to express each of the passive components' values as an *impedance* in complex-number form, recalling that resistance has an impedance phase angle[5] of zero degrees, inductance an angle of $+90^o$, and capacitance an angle of -90^o. The resistor's impedance will simply be its resistance value with a phase angle of zero degrees. For each of the reactive components, though, we must first calculate the amount of reactance (in Ohms) and then apply the proper phase angles:

$$Z_R = 12\,\Omega\angle 0^o$$

$$X_L = 2\pi fL = (2\pi)(100\text{ Hz})(0.033\text{ H}) = 20.735\,\Omega \qquad Z_L = 20.735\,\Omega\angle 90^o$$

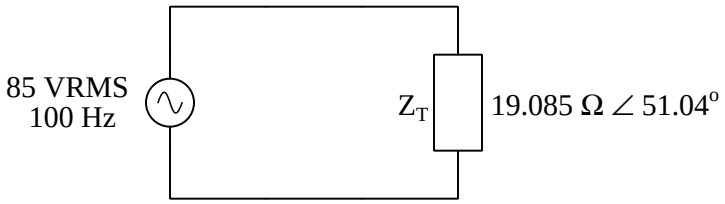
$$X_C = \frac{1}{2\pi fC} = \frac{1}{(2\pi)(100\text{ Hz})(270 \times 10^{-6}\text{ F})} = 5.895\,\Omega \qquad Z_C = 5.895\,\Omega\angle -90^o$$

The source in this circuit outputs 85 Volts (RMS), but being a series circuit we know this voltage will be divided somehow between the three passive components. Therefore, the next logical step is to combine the three passive components' impedances together to form a single equivalent impedance for the whole circuit, just as we would do in a three-resistor series DC circuit.

[5] The phase angle value of any impedance represents the phase shift between voltage and current sinusoidal waveforms for that component.

Since we know impedances, like resistances, add in series, we may use our calculator's complex addition capability to compute this total circuit impedance:

$$Z_T = Z_R + Z_L + Z_C = (12\ \Omega\angle 0^o) + (20.735\ \Omega\angle 90^o) + (5.895\ \Omega\angle -90^o) = 19.085\ \Omega\angle 51.04^o$$



Now we are properly staged to apply Ohm's Law to the calculation of circuit current. Taking source voltage and dividing by circuit impedance will yield source current, and since the source voltage was specified in Volts *RMS* this means the calculated current result will also be expressed in RMS units. Since no phase reference was specified in the circuit, we will arbitrarily set the source voltage at 0 degrees:

$$I = \frac{V}{Z} = \frac{85\ \text{V}\angle 0^o}{19.085\ \Omega\angle 51.04^o} = 4.454\ \text{A (RMS)}\angle -51.04^o$$

At this point we have all the information we need to compute circuit powers and power factor. Apparent power (S) is just the product of V and I RMS magnitudes[6] with no phase angles considered:

$$S = (85\ \text{V})(4.454\ \text{A}) = 378.58\ \text{VA}$$

True power (P) and reactive power (Q) are equal to apparent power (S) multiplied by the cosine and sine of the impedance phase angle[7], respectively:

$$P = IV\cos\theta = S\cos\theta = (378.58\ \text{VA})(\cos 51.04^o) = 238.04\ \text{W}$$

$$Q = IV\sin\theta = S\sin\theta = (378.58\ \text{VA})(\sin 51.04^o) = 294.38\ \text{VAR}$$

Only the *true* power value of 238.04 Watts represents the rate of energy leaving this circuit, and only the resistor dissipates this power. The reactive power value represents the rate of energy alternately absorbed and released by the reactive components (i.e. capacitor and inductor). Power factor is simply the cosine of the circuit impedance phase angle, equivalent to the ratio between true power and apparent power:

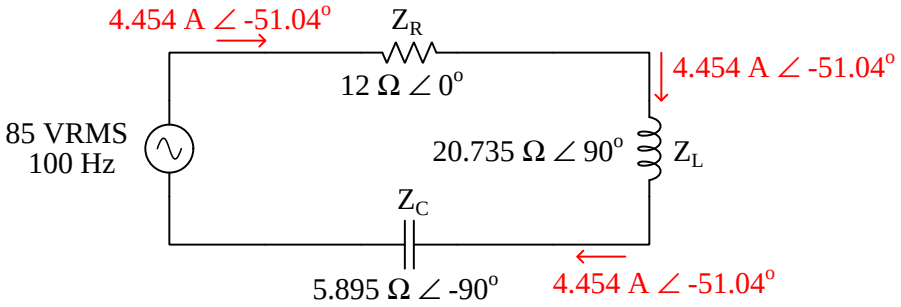
$$\text{P.F.} = \cos\theta = \cos 51.04^o = 0.6288$$

[6] Any electronic calculator capable of performing complex-number arithmetic should provide a function to deliver just the polar-form magnitude (and not the angle) of any complex quantity. Applying this function to the complex current value of $4.454\ \angle\ -51.04^o$ yields the simple value of 4.454 with no rounding.

[7] Similarly, any electronic calculator with complex-number functions will provide a means to extract the polar-form angle from any complex quantity so that the angle by itself may be used with no rounding resulting from re-typing. Applying this function to the total impedance value of $19.085\ \Omega\ \angle\ 51.04^o$ yields 51.04^o precisely.

Power factor is a unitless quantity, but we may further qualify this value by saying it is equal to *0.6288 lagging*, because the negative phase angle of current (-51.04^o) means current *lags* behind voltage in this circuit.

Returning to our analysis of voltages and currents, we are ready to "expand" this single-impedance equivalent circuit to its original form. Since we are expanding Z_T into three series-connected impedances, and we know it is *current* that is common throughout any series network, we may transfer the calculated value for circuit current to each of the three impedances:



Now we have all the information needed to calculate voltage drop across each of the passive components, using Ohm's Law:

$$V_R = IZ_R = (4.454\ \text{A}\angle -51.04^o)(12\ \Omega\angle 0^o) = 53.45\ \text{V}\angle -51.04^o$$

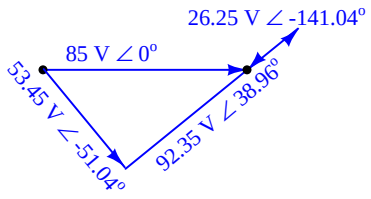
$$V_L = IZ_L = (4.454\ \text{A}\angle -51.04^o)(20.735\ \Omega\angle 90^o) = 92.35\ \text{V}\angle 38.96^o$$

$$V_C = IZ_C = (4.454\ \text{A}\angle -51.04^o)(5.895\ \Omega\angle -90^o) = 26.25\ \text{V}\angle -141.04^o$$

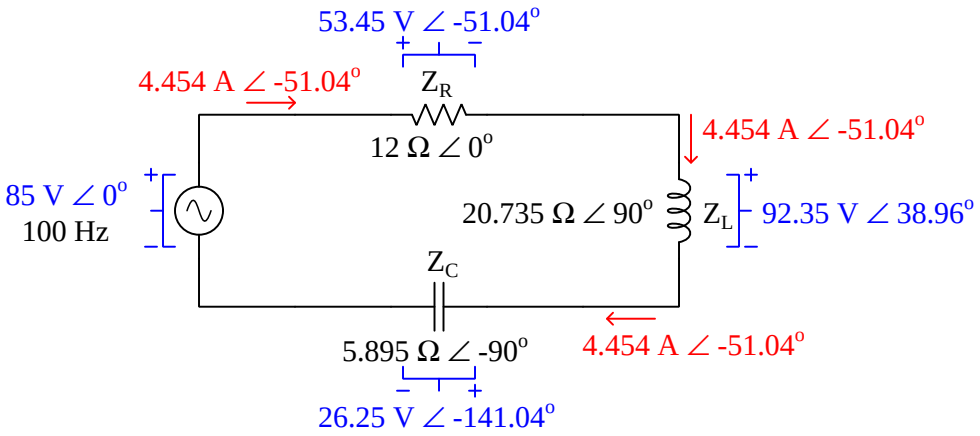
As a final check on our work, we can apply Kirchhoff's Voltage Law by adding these three component voltages together to verify that they do indeed add up to the source voltage. Recalling all values from our calculator's memory to avoid rounding errors, the result is *exactly* 85 Volts at 0 degrees:

$$(53.45\ \text{V}\angle -51.04^o) + (92.35\ \text{V}\angle 38.96^o) + (26.25\ \text{V}\angle -141.04^o) = 85.00\ \text{V}\angle 0^o$$

Note just how *non-intuitive* this sum is. If we focus only on the voltage magnitudes, it seems implausible that 53.45 Volts plus 92.35 Volts plus 26.25 Volts could somehow equal 85 Volts. However, we must remember these are all *phasor* quantities, possessing *angles* as well as magnitudes. A *phasor diagram* makes the result more comprehensible:



A final step we should take is the assignment of *polarity symbols* on the circuit diagram. Even though this is AC and not DC, those + and − symbols are still useful in order to give meaning[8] to the calculated voltage angles, just as arrows showing the direction of (conventional) current give context to the phase angle of any AC current. You will notice the clockwise direction of current in this circuit was arbitrarily chosen, since the original circuit gave no hint as to either current direction or voltage polarity. The general rule in AC circuit analysis is this: *Assign voltage polarities and current arrows to the AC energy source as though it was a DC source, and to all reactive and resistive components as though they were DC loads.* We could have done this at the very beginning of our analysis, but in this rather uncomplicated circuit it works just as well to do it now:

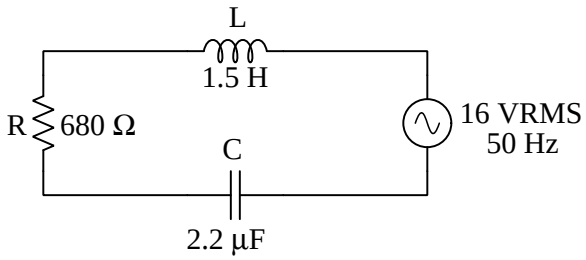


Voltage polarity and current arrow annotations become essential when applying Kirchhoff's Laws, because they indicate to us whether the quantities in question must be *added to* each other or *subtracted from* each other. Our last analytical step of adding the three passive component voltages together to check that the sum equals the source voltage really didn't *require* us to have polarity symbols written on the diagram because the concept was familiar enough to anyone well-practiced in DC circuit analysis. However, now that the polarity symbols have been included in the diagram, it is clear to see how the resistor, inductor, and capacitor voltages *must* add up to equal the source voltage: the passive component voltages are all aiding each other, but are all opposed to the source voltage.

[8] + and − voltage polarity symbols represent the actual polarity of the voltage at the instant its waveform reaches the 0^o point in time, just as arrow symbols show the actual direction of current at the instant in time its waveform reaches 0^o. The inductor and capacitor voltages, for example, appear in the schematic diagram as though they are aiding each other, with the capacitor's + connected to the inductors −. However, their phase angles are entirely opposed to each other, with -141.04^o and 38.96^o being a full *180 degrees apart.* The phasor diagram makes this clear, with the capacitor's voltage phasor pointed opposite to the inductor's voltage phasor. Thus, the + and − annotations must be considered in tandem with the phase angle of each voltage, and not as simultaneous indicators.

3.4 Example circuit analysis using ordinary calculator

Calculators capable of complex-number arithmetic ease the burden of AC circuit analysis because they make that analysis almost as simple as for DC circuits: simply represent each and every quantity as a complex number rather than a scalar number, and nearly all the same principles (e.g. Ohm's Law, Kirchhoff's Laws, properties of series and parallel networks) still apply. However, it is good to know how to perform these same analyses without the benefit of an advanced calculator not just for occasions where such a calculator is unavailable, but also for scenarios with missing information that might make it impossible to represent all the given information in full complex form. In this section we will analyze a series RLC circuit using arithmetic functions provided by any scientific calculator:



As usual, we will begin by translating each of these passive component values into an *impedance*:

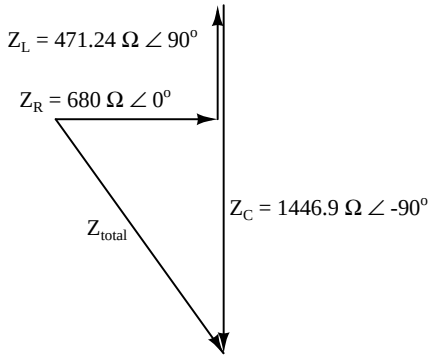
$$Z_R = 680\,\Omega\angle 0^o$$

$$X_L = 2\pi f L = (2\pi)(50\text{ Hz})(1.5\text{ H}) = 471.24\,\Omega \qquad Z_L = 471.24\,\Omega\angle 90^o$$

$$X_C = \frac{1}{2\pi f C} = \frac{1}{(2\pi)(50\text{ Hz})(2.2 \times 10^{-6}\text{ F})} = 1446.9\,\Omega \qquad Z_C = 1446.9\,\Omega\angle -90^o$$

We know that impedances, like resistances, always add in a series network. However, we must respect the phase angles of each impedance when we add them, which means we cannot simply compute $680\,\Omega + 471.24\,\Omega + 1446.9\,\Omega$ as though these were three series-connected resistors.

A helpful aid in complex-number addition is the *phasor diagram* where every quantity is shown as a vector with both a magnitude (length) and direction (angle). Here we see these three impedances represented graphically, the sum represented by the resultant vector from the tail of the first to the tip of the last[9]:



Since these stacked vectors comprise a right-triangle, we may calculate total impedance (Z_{total}) magnitude using the Pythagorean Theorem with just the magnitudes[10] of Z_R, Z_L, and Z_C:

$$Z_{total} = \sqrt{Z_R^2 + (Z_L - Z_C)^2}$$

$$Z_{total} = \sqrt{(680\ \Omega)^2 + ((471.24\ \Omega) - (1446.9\ \Omega))^2}$$

$$Z_{total} = 1189.2\ \Omega$$

We still need to find the angle of this total impedance in order to fully quantify it, though. The Pythagorean Theorem only yields the *length* of the hypotenuse vector, not its angle. To find the angle relative to the horizontal, we must apply a trigonometric arc-function. In this case we will use the arc-tangent, with Z_R's magnitude as the adjacent side-length and the magnitude of $Z_L - Z_C$ as the opposite side-length:

$$\theta = \arctan\left(\frac{Z_L - Z_C}{Z_R}\right)$$

$$\theta = \arctan\left(\frac{471.24\ \Omega - 1446.9\ \Omega}{680\ \Omega}\right)$$

$$\theta = -55.124^o$$

[9] Alternatively, we may represent each of these phasors in *rectangular* form, add up their real and imaginary portions respectively, and then convert back to polar form. In this case, $(680+j0)+(0+j471.24)+(0-j1446.9) = 680-j975.62$. This rectangular-form sum may be converted to polar form using the Pythagorean Theorem to calculate magnitude and the arctangent function to calculate angle, yielding $1189.2\angle -55.124^o$.

[10] Since we are doing this calculation using only scalar quantities, we could have just as accurately written the Pythagorean Theorem formula as $Z_{total} = \sqrt{G^2 + (X_L - X_C)^2}$, being the square-root of the sum of the square of conductance and the square of the difference in reactances.

Therefore, the total impedance for this circuit is:

$$Z_{total} = 1189.2\,\Omega\angle -55.124^o$$

The total impedance's phase angle may be used to calculate the circuit's power factor:

$$\text{P.F.} = \cos\theta = \cos -55.124^o = 0.57180$$

We can tell that this power factor is *leading* because the total impedance phase angle is negative rather than positive. In other words, the capacitive influence is greater than the inductive influence in this circuit.

Apparent power may be calculated from source current and total impedance magnitude, and the other powers from that:

$$S = \frac{V^2}{Z_{total}} = \frac{16\text{ V}}{1189.2\,\Omega} = 215.27\text{ mVA}$$

$$P = S\cos\theta = \cos(-55.124^o) = 123.09\text{ mW}$$

$$Q = S\sin\theta = \sin(-55.124^o) = 176.60\text{ mVAR leading}$$

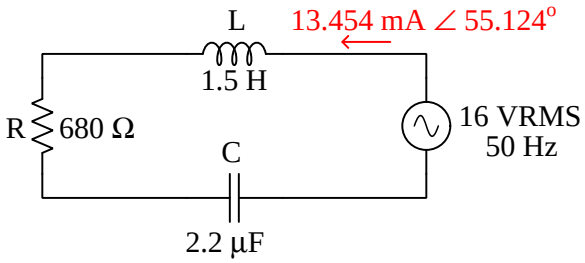
Now that we know the circuit's total impedance, we may calculate the current through all four series components by applying Ohm's Law. Division of polar complex numbers is simple, consisting of the quotient of the magnitudes and the difference of the angles:

$$I_{total} = \frac{V_{total}}{Z_{total}}$$

$$I_{total} = \frac{16\text{ V}\angle 0^o}{1189.2\,\Omega\angle -55.124^o}$$

$$I_{total} = \frac{16\text{ V}}{1189.2\,\Omega}\angle\,(0 - (-55.124^o))$$

$$I_{total} = 13.454\text{ mA}\angle 55.124^o$$



Now we are prepared to calculate voltages across each of the passive components using Ohm's Law, multiplying these polar complex quantities by finding the products of the magnitudes and sums of the angles:

$$V_R = I_{total} Z_R = (13.454 \text{ mA} \angle 55.124^o)(680\,\Omega \angle 0^o) = 9.1489 \text{ V} \angle 55.124^o$$

$$V_L = I_{total} Z_L = (13.454 \text{ mA} \angle 55.124^o)(471.24\,\Omega \angle 90^o) = 6.3401 \text{ V} \angle 145.12^o$$

$$V_C = I_{total} Z_C = (13.454 \text{ mA} \angle 55.124^o)(1446.9\,\Omega \angle -90^o) = 19.466 \text{ V} \angle -34.876^o$$

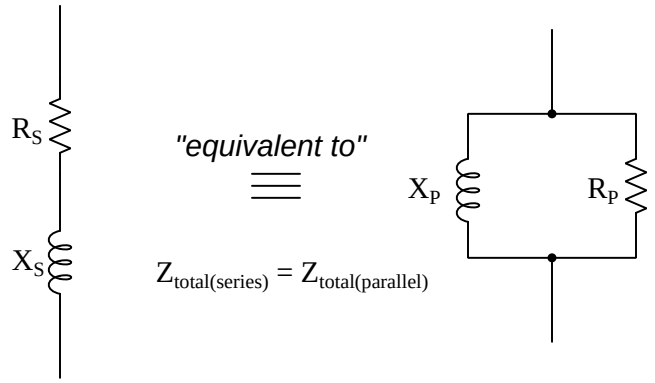
Chapter 4

Derivations and Technical References

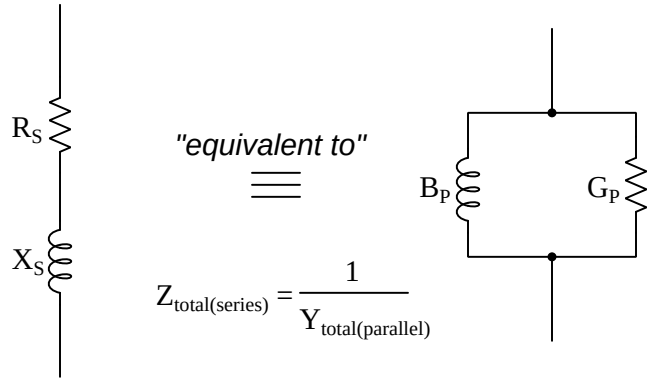
This chapter is where you will find mathematical derivations too detailed to include in the tutorial, and/or tables and other technical reference material.

4.1 Equivalent series and parallel XR networks

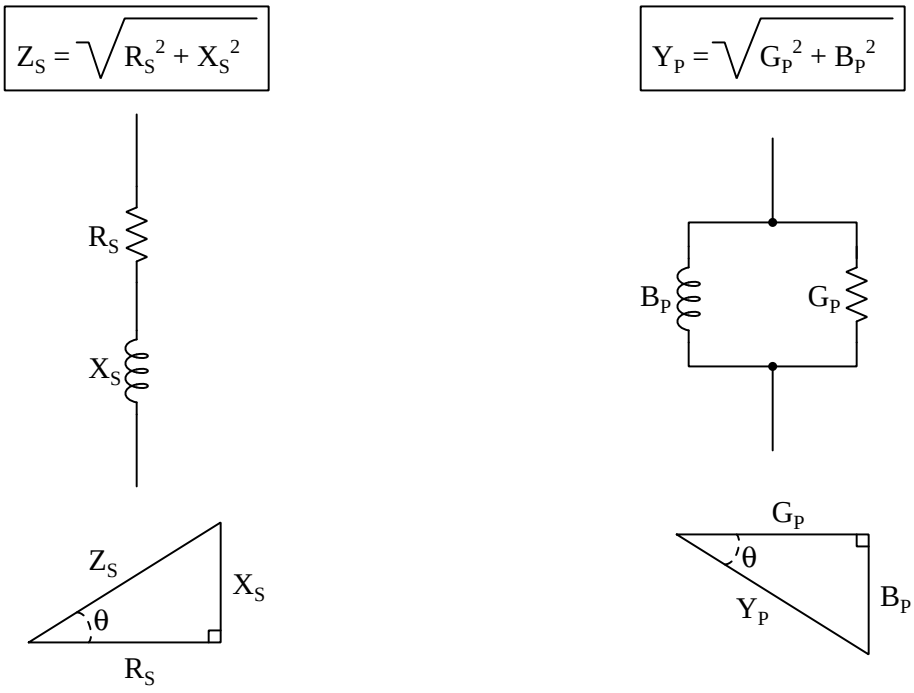
It is often useful in AC circuit analysis to be able to convert a series combination of resistance and reactance into an equivalent parallel combination of resistance and reactance, or vise-versa:



We know that while resistances and reactances add (as phasor quantities) in series, the same cannot be said in parallel. However, the reciprocal of resistance (called *conductance*, G) and the reciprocal of reactance (called *susceptance*, B) do add as phasor quantities in parallel to yield a sum known as *admittance* (Y) which is the reciprocal of impedance (Z):



These sums may be computed using scalar numbers, with series impedance (Z) being the Pythagorean sum of R and X being Z, and parallel admittance (Y) being the Pythagorean sum of G and B:



If these two circuits are truly equivalent to one another, having the same total impedance, then their representative triangles should be geometrically similar (identical angles, same proportions of side lengths). With equal proportions, $\frac{R_S}{Z_S}$ in the series circuit triangle should be the same ratio as $\frac{G_P}{Y_P}$ in the parallel circuit triangle, that is $\frac{R_S}{Z_S} = \frac{G_P}{Y_P}$.

We may derive an equation relating resistance in the series network to resistance in the parallel network (and to the total resistance of each network) by taking this proportionality and substituting $\frac{1}{R_P}$ for G_P and $\frac{1}{Z_P}$ for Y_P:

$$\frac{R_S}{Z_S} = \frac{G_P}{Y_P}$$

$$\frac{R_S}{Z_S} = \frac{\frac{1}{R_P}}{\frac{1}{Z_P}}$$

$$\frac{R_S}{Z_S} = \frac{Z_P}{R_P}$$

$$R_S R_P = Z_S Z_P$$

If these are truly equivalent circuits, then they must have the same total impedance. In other words, $Z_S = Z_P$. Simply calling this equal impedance Z and substituting for both Z_S and Z_P:

$$R_S R_P = ZZ$$

$$R_S R_P = Z^2$$

The same proportionality applies just as well to reactance in the two equivalent networks, with similar results:

$$\frac{X_S}{Z_S} = \frac{B_P}{Y_P}$$

$$\frac{X_S}{Z_S} = \frac{\frac{1}{X_P}}{\frac{1}{Z_P}}$$

$$\frac{X_S}{Z_S} = \frac{Z_P}{X_P}$$

$$X_S X_P = Z_S Z_P$$

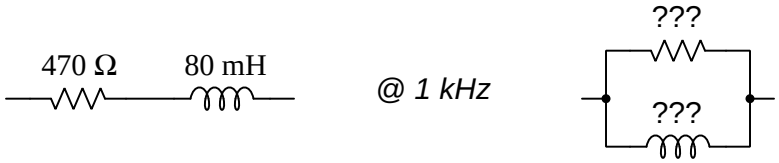
$$X_S X_P = ZZ$$

$$X_S X_P = Z^2$$

With that, we have two equations useful for finding the resistance and reactance values necessary to create an equivalent parallel network for a given series network, or vice-versa:

$$R_S R_P = Z^2 \qquad X_S X_P = Z^2$$

Let's apply this to a practical example to see how it works. Take for instance a series network comprised of a 470 Ω resistor and an 80 mH inductor operating at a frequency of 1 kHz, with the goal of determining component values necessary to build an equivalent parallel LR network.



The 80 mH inductor will have a reactance of 502.65 Ω at 1 kHz. Together, the 470 Ω resistor and the 80 mH inductor make a series impedance value of:

$$Z = \sqrt{R^2 + X^2}$$

$$Z = \sqrt{(470\,\Omega)^2 + (502.65\,\Omega)^2} = 688.16\,\Omega$$

Now that we know the total impedance of the series network, we may calculate both the parallel resistance and parallel reactance values for the equivalent parallel network using the two equations previously derived:

$$R_S R_P = Z^2$$

$$R_P = \frac{Z^2}{R_S}$$

$$R_P = \frac{(688.16\,\Omega)^2}{470\,\Omega} = 1007.58\,\Omega$$

$$X_S X_P = Z^2$$

$$X_P = \frac{Z^2}{X_S}$$

$$X_P = \frac{(688.16\,\Omega)^2}{502.65\,\Omega} = 942.12\,\Omega$$

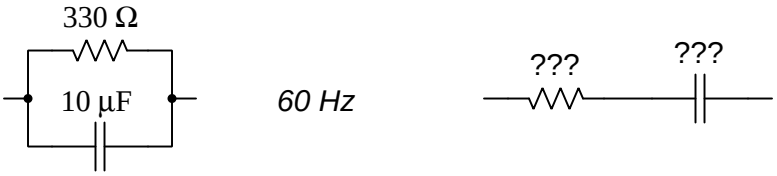
Solving for inductance at 1 kHz:

$$X_L = 2\pi f L$$

$$L = \frac{X_L}{2\pi f} = \frac{942.12\,\Omega}{(2\pi)(1000\text{ Hz})} = 149.94\text{ mH}$$

Therefore the equivalent parallel network consists of a 1007.58 Ω resistor in parallel with a 149.94 mH inductor (assuming a frequency of 1 kHz).

Let's apply this to another practical example. Take for instance a parallel network comprised of a 330 Ω resistor and a 10 μF capacitor operating at a frequency of 60 Hz, with the goal of determining component values necessary to build an equivalent series RC network.



The 10 μF capacitor will have a reactance of 265.26 Ω at 60 Hz. Avoiding the use of complex numbers, we may convert 330 Ω into a conductance (G) and the 10 μF capacitor's reactance into a susceptance (B) to compute the parallel network's total impedance:

$$X_C = \frac{1}{2\pi f C} = \frac{1}{(2\pi)(60)(10\,\mu\text{F})} = 265.26\,\Omega$$

$$B = \frac{1}{X_C} = \frac{1}{265.26\,\Omega} = 3.770\text{ mS}$$

$$G = \frac{1}{R} = \frac{1}{330\,\Omega} = 3.0303\text{ mS}$$

$$Y = \sqrt{G^2 + B^2} = \sqrt{(3.0303\text{ mS})^2 + (3.770\text{ mS})^2} = 4.837\text{ mS}$$

$$Z = \frac{1}{Y} = 206.75\,\Omega$$

Now that we know the parallel network's impedance magnitude, we may calculate resistance and reactance for the equivalent series network using our two equations:

$$R_S R_P = Z^2$$

$$R_P = \frac{Z^2}{R_S}$$

$$R_P = \frac{(206.75\,\Omega)^2}{330\,\Omega} = 129.53\,\Omega$$

$$X_S X_P = Z^2$$

$$X_P = \frac{Z^2}{X_S}$$

$$X_P = \frac{(206.75\,\Omega)^2}{265.26\,\Omega} = 161.14\,\Omega$$

Calculating the necessary capacitance to yield 161.14 Ω of reactance at 60 Hz:

$$X_C = \frac{1}{2\pi f C}$$

$$C = \frac{1}{2\pi f X_C}$$

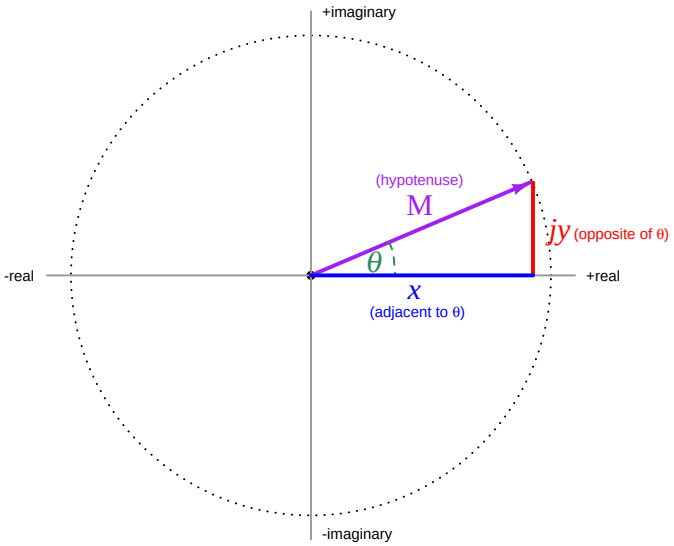
$$C = \frac{1}{(2\pi)(60\text{ Hz})(161.14\,\Omega)} = 16.46\,\mu\text{F}$$

Therefore the equivalent series network consists of a 129.53 Ω resistor in series with a 16.46 μF capacitor (assuming a frequency of 60 Hz).

4.2 Complex-number arithmetic

Complex numbers are very useful in AC circuit analysis because each one has the ability to represent both a magnitude and a phase shift between that quantity and some other reference quantity. Despite the existence of electronic calculators and computer software capable of performing arithmetic on complex-number quantities, there are still times when we must perform some calculation on these quantities "by hand". This technical reference reviews the basic arithmetic operations on complex numbers, complete with examples.

Recall that complex numbers may be represented in either *rectangular* or *polar* form, rectangular being a quantity with both a "real" and an "imaginary" component, and polar being a quantity with a magnitude and an angle. Graphically, these two forms relate to the sides of a right triangle:



Rectangular form: $x + jy$ (where $j = \sqrt{-1}$)

Polar form: $M\angle\theta$

To convert from rectangular form to polar form, $M = \sqrt{x^2 + y^2}$ and $\theta = \arctan \frac{y}{x}$

To convert from polar form to rectangular form, $x = M \cos\theta$ and $y = M \sin\theta$

As we will see, addition and subtraction is easiest to do with rectangular-form notation while multiplication and division is easiest to do with polar-form notation. Thus, circuit analysis doing "long-hand" complex-number arithmetic often involves conversions back and forth between rectangular and polar forms in order to set up the quantities before applying Ohm's Law, Kirchhoff's Laws, etc. This can be tedious, and it is also prone to rounding errors. The reader is advised to store all intermediate results in their calculator's memory and recall when needed, rather than re-type quantities and thereby incur rounding errors due to truncation.

4.2.1 Negating complex numbers

The sign of a complex number may be reversed just as easily in rectangular form as in polar form. Rectangular-form negation consists of multiplying -1 through to both the real and imaginary terms. Polar-form negation consists solely of adding 180 degrees to the angle, or alternatively, by reversing the sign of the magnitude and leaving the angle alone.

Example: reverse the sign of $5 - j4$

$$-(5 - j4)$$

$$-5 + j4$$

Example: reverse the sign of $6\angle 30^o$

$$-(6\angle 30^o)$$

$$6\angle 210^o = 6\angle -150^o = -6\angle 30^o$$

4.2.2 Adding complex numbers

Complex numbers are most easily added in *rectangular form*: simply add the real portions and then add the imaginary portions.

Example: add $5 - j4$ to $-1 - j3$

$$(5 - j4) + (-1 - j3)$$

$$(5 + (-1)) + (-j4 + (-j3))$$

$$4 - j7$$

4.2.3 Subtracting complex numbers

Complex numbers are most easily subtracted in *rectangular form*: simply subtract the real portions and then subtract the imaginary portions.

Example: subtract $5 - j4$ from $-1 - j3$

$$(-1 - j3) - (5 - j4)$$

$$(-1 - (5)) + (-j3 - (-j4))$$

$$-6 + j1$$

4.2.4 Multiplying complex numbers

Complex numbers are most easily multiplied in *polar form*: simply multiply the magnitudes and add the angles.

Example: multiply $6\angle 30^o$ by $2\angle -10^o$

$$(6\angle 30^o) \times (2\angle -10^o)$$

$$(6 \times 2)\angle(30^o + (-10^o))$$

$$12\angle 20^o$$

Multiplication of rectangular-form complex numbers less straight-forward then with polar-form numbers, and resembles multiplication of algebraic polynomials:

Example: multiply $5 - j4$ by $-1 - j3$

$$(5 - j4) \times (-1 - j3)$$

$$(5 \times (-1)) + (5 \times (-j3)) + (-j4 \times (-1)) + (-j4 \times (-j3))$$

$$(-5) + (-j15) + (j4) + (j^2 12)$$

$$(-5) + (-j15) + (j4) + ((-1)12)$$

$$(-5) + (-j15) + (j4) + (-12)$$

$$-17 - j11$$

4.2.5 Dividing complex numbers

Complex numbers are most easily divided in *polar form*: simply divide the magnitudes and subtract the angles.

Example: divide $6\angle 30^o$ by $2\angle -10^o$

$$\frac{6\angle 30^o}{2\angle -10^o}$$

$$\frac{6}{2}\angle(30^o - (-10^o))$$

$$3\angle 40^o$$

4.2.6 Reciprocating complex numbers

Reciprocation is division into one, and so complex numbers are reciprocated most easily in *polar form* just as division is best performed in polar form: simply reciprocate the magnitude and negate the angle.

Example: reciprocate $2\angle -10^o$

$$\frac{1}{2\angle -10^o}$$

$$\frac{1}{2}\angle -(-10^o)$$

$$0.5\angle 10^o$$

4.2.7 Calculator tips

Here is some advice when using calculators to do complex-number arithmetic:

- When manually entering a complex-number value, enclose that value in parentheses. Some calculators struggle to properly perform order-of-operations with complex numbers. For example, some calculators will interpret $45\angle 30^o \times 5$ as $45\angle(30^o \times 5)$ to give $45\angle 150^o$ when what was really intended was $(45\angle 30^o) \times 5 = 225\angle 30^o$. Also, note that the practice of highlighting previous results in a multi-line display and then "pasting" those results into a new calculation may suffer similar problems.
- *Never* re-enter a non-round computed result, but instead save that to a memory location and then recall from memory when needed for further calculations. You will find that rounding errors compound *aggressively* in complex-number arithmetic, and so the general good habit of using memory locations becomes a near-necessity with these calculations. Another important benefit to using memory locations is the avoidance of the order-of-operations problem mentioned previously: when recalling a complex-number value from memory and then placing that variable name (e.g. x) into subsequent calculations, the calculator treats the memory variable as a complete number rather than incorrectly operating on only one of its parts.

Chapter 5

Programming References

A powerful tool for mathematical modeling is text-based *computer programming*. This is where you type coded commands in text form which the computer is able to interpret. Many different text-based languages exist for this purpose, but we will focus here on just two of them, *C++* and *Python*.

5.1 Programming in C++

One of the more popular text-based computer programming languages is called *C++*. This is a *compiled* language, which means you must create a plain-text file containing C++ code using a program called a *text editor*, then execute a software application called a *compiler* to translate your "source code" into instructions directly understandable to the computer. Here is an example of "source code" for a very simple C++ program intended to perform some basic arithmetic operations and print the results to the computer's console:

```
#include <iostream>
using namespace std;

int main (void)
{
  float x, y;

  x = 200;
  y = -560.5;

  cout << "This simple program performs basic arithmetic on" << endl;
  cout << "the two numbers " << x << " and " << y << " and then" << endl;
  cout << "displays the results on the computer's console." << endl;

  cout << endl;

  cout << "Sum = " << x + y << endl;
  cout << "Difference = " << x - y << endl;
  cout << "Product = " << x * y << endl;
  cout << "Quotient of " << x / y << endl;

  return 0;
}
```

Computer languages such as C++ are designed to make sense when read by human programmers. The general order of execution is left-to-right, top-to-bottom just the same as reading any text document written in English. Blank lines, indentation, and other "whitespace" is largely irrelevant in C++ code, and is included only to make the code more pleasing[1] to view.

[1] Although not included in this example, *comments* preceded by double-forward slash characters (`//`) may be added to source code as well to provide explanations of what the code is supposed to do, for the benefit of anyone reading it. The compiler application will ignore all comments.

Let's examine the C++ source code to explain what it means:

- `#include <iostream>` and `using namespace std;` are set-up instructions to the compiler giving it some context in which to interpret your code. The code specific to your task is located between the brace symbols ({ and }, often referred to as "curly-braces").

- `int main (void)` labels the "Main" function for the computer: the instructions within this function (lying between the { and } symbols) it will be commanded to execute. Every complete C++ program contains a `main` function at minimum, and often additional functions as well, but the `main` function is where execution always begins. The `int` declares this function will return an *integer* number value when complete, which helps to explain the purpose of the `return 0;` statement at the end of the `main` function: providing a numerical value of zero at the program's completion as promised by `int`. This returned value is rather incidental to our purpose here, but it is fairly standard practice in C++ programming.

- Grouping symbols such as (parentheses) and {braces} abound in C, C++, and other languages (e.g. Java). Parentheses typically group data to be processed by a function, called *arguments* to that function. Braces surround lines of executable code belonging to a particular function.

- The `float` declaration reserves places in the computer's memory for two *floating-point* variables, in this case the variables' names being `x` and `y`. In most text-based programming languages, variables may be named by single letters or by combinations of letters (e.g. `xyz` would be a single variable).

- The next two lines assign numerical values to the two variables. Note how each line terminates with a semicolon character (`;`) and how this pattern holds true for most of the lines in this program. In C++ semicolons are analogous to periods at the ends of English sentences. This demarcation of each line's end is necessary because C++ ignores whitespace on the page and doesn't "know" otherwise where one line ends and another begins.

- All the other instructions take the form of a `cout` command which prints characters to the "standard output" stream of the computer, which in this case will be text displayed on the console. The double-less-than symbols (`<<`) show data being sent *toward* the `cout` command. Note how verbatim text is enclosed in quotation marks, while variables such as `x` or mathematical expressions such as `x - y` are not enclosed in quotations because we want the computer to display the numerical values represented, not the literal text.

- Standard arithmetic operations (add, subtract, multiply, divide) are represented as `+`, `-`, `*`, and `/`, respectively.

- The `endl` found at the end of every `cout` statement marks the end of a line of text printed to the computer's console display. If not for these `endl` inclusions, the displayed text would resemble a run-on sentence rather than a paragraph. Note the `cout << endl;` line, which does nothing but create a blank line on the screen, for no reason other than esthetics.

After saving this *source code* text to a file with its own name (e.g. `myprogram.cpp`), you would then *compile* the source code into an *executable* file which the computer may then run. If you are using a console-based compiler such as *GCC* (very popular within variants of the Unix operating system[2], such as Linux and Apple's OS X), you would type the following command and press the Enter key:

```
g++ -o myprogram.exe myprogram.cpp
```

This command instructs the `GCC` compiler to take your source code (`myprogram.cpp`) and create with it an executable file named `myprogram.exe`. Simply typing `./myprogram.exe` at the command-line will then execute your program:

```
./myprogram.exe
```

If you are using a graphic-based C++ development system such as Microsoft Visual Studio[3], you may simply create a new console application "project" using this software, then paste or type your code into the example template appearing in the editor window, and finally run your application to test its output.

As this program runs, it displays the following text to the console:

```
This simple program performs basic arithmetic on
the two numbers 200 and -560.5 and then
displays the results on the computer's console.

Sum = -360.5
Difference = 760.5
Product = -112100
Quotient of -0.356824
```

As crude as this example program is, it serves the purpose of showing how easy it is to write and execute simple programs in a computer using the C++ language. As you encounter C++ example programs (shown as source code) in any of these modules, feel free to directly copy-and-paste the source code text into a text editor's screen, then follow the rest of the instructions given here (i.e. save to a file, compile, and finally run your program). You will find that it is generally easier to

[2] A very functional option for users of Microsoft Windows is called *Cygwin*, which provides a Unix-like console environment complete with all the customary utility applications such as GCC!

[3] Using Microsoft Visual Studio community version 2017 at the time of this writing to test this example, here are the steps I needed to follow in order to successfully compile and run a simple program such as this: (1) Start up Visual Studio and select the option to create a New Project; (2) Select the Windows Console Application template, as this will perform necessary set-up steps to generate a console-based program which will save you time and effort as well as avoid simple errors of omission; (3) When the editing screen appears, type or paste the C++ code within the `main()` function provided in the template, deleting the "Hello World" `cout` line that came with the template; (4) Type or paste any preprocessor directives (e.g. `#include` statements, `namespace` statements) necessary for your code that did not come with the template; (5) Lastly, under the Debug drop-down menu choose either Start Debugging (F5 hot-key) or Start Without Debugging (Ctrl-F5 hotkeys) to compile ("Build") and run your new program. Upon execution a console window will appear showing the output of your program.

learn computer programming by closely examining others' example programs and modifying them than it is to write your own programs starting from a blank screen.

5.2 Programming in Python

Another text-based computer programming language called *Python* allows you to type instructions at a terminal prompt and receive immediate results without having to compile that code. This is because Python is an *interpreted* language: a software application called an *interpreter* reads your source code, translates it into computer-understandable instructions, and then executes those instructions in one step.

The following shows what happens on my personal computer when I start up the Python interpreter on my personal computer, by typing `python3`[4] and pressing the Enter key:

```
Python 3.7.2 (default, Feb 19 2019, 18:15:18)
[GCC 4.1.2] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

The `>>>` symbols represent the prompt within the Python interpreter "shell", signifying readiness to accept Python commands entered by the user.

Shown here is an example of the same arithmetic operations performed on the same quantities, using a Python interpreter. All lines shown preceded by the `>>>` prompt are entries typed by the human programmer, and all lines shown without the `>>>` prompt are responses from the Python interpreter software:

```
>>> x = 200
>>> y = -560.5
>>> x + y
-360.5
>>> x - y
760.5
>>> x * y
-112100.0
>>> x / y
-0.35682426404995538
>>> quit()
```

[4] Using version 3 of Python, which is the latest at the time of this writing.

More advanced mathematical functions are accessible in Python by first entering the line `from math import *` which "imports" these functions from Python's math *library* (with functions identical to those available for the C programming language, and included on any computer with Python installed). Some examples show some of these functions in use, demonstrating how the Python interpreter may be used as a scientific calculator:

```
>>> from math import *
>>> sin(30.0)
-0.98803162409286183
>>> sin(radians(30.0))
0.49999999999999994
>>> pow(2.0, 5.0)
32.0
>>> log10(10000.0)
4.0
>>> e
2.7182818284590451
>>> pi
3.1415926535897931
>>> log(pow(e,6.0))
6.0
>>> asin(0.7071068)
0.78539819000368838
>>> degrees(asin(0.7071068))
45.000001524425265
>>> quit()
```

Note how trigonometric functions assume angles expressed in *radians* rather than *degrees*, and how Python provides convenient functions for translating between the two. Logarithms assume a base of e unless otherwise stated (e.g. the `log10` function for common logarithms).

The interpreted (versus compiled) nature of Python, as well as its relatively simple syntax, makes it a good choice as a person's first programming language. For complex applications, interpreted languages such as Python execute slower than compiled languages such as C++, but for the very simple examples used in these learning modules speed is not a concern.

Another Python math library is `cmath`, giving Python the ability to perform arithmetic on complex numbers. This is very useful for AC circuit analysis using *phasors*[5] as shown in the following example. Here we see Python's interpreter used as a scientific calculator to show series and parallel impedances of a resistor, capacitor, and inductor in a 60 Hz AC circuit:

```
>>> from math import *
>>> from cmath import *
>>> r = complex(400,0)
>>> f = 60.0
>>> xc = 1/(2 * pi * f * 4.7e-6)
>>> zc = complex(0,-xc)
>>> xl = 2 * pi * f * 1.0
>>> zl = complex(0,xl)
>>> r + zc + zl
(400-187.38811239154882j)
>>> 1/(1/r + 1/zc + 1/zl)
(355.837695813625+125.35793777619385j)
>>> polar(r + zc + zl)
(441.717448903332, -0.4381072059213295)
>>> abs(r + zc + zl)
441.717448903332
>>> phase(r + zc + zl)
-0.4381072059213295
>>> degrees(phase(r + zc + zl))
-25.10169387356105
```

When entering a value in rectangular form, we use the `complex()` function where the arguments are the real and imaginary quantities, respectively. If we had opted to enter the impedance values in polar form, we would have used the `rect()` function where the first argument is the magnitude and the second argument is the angle in radians. For example, we could have set the capacitor's impedance (`zc`) as $X_C \angle -90^o$ with the command `zc = rect(xc,radians(-90))` rather than with the command `zc = complex(0,-xc)` and it would have worked the same.

Note how Python defaults to rectangular form for complex quantities. Here we defined a 400 Ohm resistance as a complex value in rectangular form (400 +j0 Ω), then computed capacitive and inductive reactances at 60 Hz and defined each of those as complex (phasor) values ($0 - jX_c$ Ω and $0 + jX_l$ Ω, respectively). After that we computed total impedance in series, then total impedance in parallel. Polar-form representation was then shown for the series impedance (441.717 Ω $\angle -25.102^o$). Note the use of different functions to show the polar-form series impedance value: `polar()` takes the complex quantity and returns its polar magnitude and phase angle in *radians*; `abs()` returns just the polar magnitude; `phase()` returns just the polar angle, once again in radians. To find the polar phase angle in degrees, we nest the `degrees()` and `phase()` functions together.

The utility of Python's interpreter environment as a scientific calculator should be clear from these examples. Not only does it offer a powerful array of mathematical functions, but also unlimited

[5] A "phasor" is a voltage, current, or impedance represented as a complex number, either in rectangular or polar form.

assignment of variables as well as a convenient text record[6] of all calculations performed which may be easily copied and pasted into a text document for archival.

It is also possible to save a set of Python commands to a text file using a text editor application, and then instruct the Python interpreter to execute it at once rather than having to type it line-by-line in the interpreter's shell. For example, consider the following Python program, saved under the filename `myprogram.py`:

```
x = 200
y = -560.5

print("Sum")
print(x + y)

print("Difference")
print(x - y)

print("Product")
print(x * y)

print("Quotient")
print(x / y)
```

As with C++, the interpreter will read this source code from left-to-right, top-to-bottom, just the same as you or I would read a document written in English. Interestingly, whitespace *is* significant in the Python language (unlike C++), but this simple example program makes no use of that.

To execute this Python program, I would need to type `python myprogram.py` and then press the Enter key at my computer console's prompt, at which point it would display the following result:

```
Sum
-360.5
Difference
760.5
Product
-112100.0
Quotient
-0.35682426405
```

As you can see, syntax within the Python programming language is simpler than C++, which is one reason why it is often a preferred language for beginning programmers.

[6] Like many command-line computing environments, Python's interpreter supports "up-arrow" recall of previous entries. This allows quick recall of previously typed commands for editing and re-evaluation.

If you are interested in learning more about computer programming in *any* language, you will find a wide variety of books and free tutorials available on those subjects. Otherwise, feel free to learn by the examples presented in these modules.

5.3 Modeling series RLC circuits using C++

The following program written in C++ analyzes a series RLC circuit energized by a voltage source:

```
#include <iostream>
#include <complex>
#include <cmath>
using namespace std;

int main (void)
{
  double vsrc = 30.0;
  double f = 55.0;
  double r = 30.0;
  double l = 110e-3;
  double c = 100.0e-6;

  complex <double> zc, zl, zseries, vc, vl, vr, i;

  cout << "+--V---R---L---C--+ " << " V_source = " << vsrc << " Volts " << endl;
  cout << "|                 | " << " f_source = " << f << " Hertz " << endl;
  cout << "+-----------------+ " << " R = " << r << " Ohms " << endl;
  cout << "                     L = " << l << " Henrys " << endl;
  cout << "                     C = " << c << " Farads " << endl << endl;

  zl = complex <double> (0.0, 2 * M_PI * f * l);
  zc = complex <double> (0.0, -1 / (2 * M_PI * f * c));

  zseries = zc + zl + r;
  i = vsrc / zseries;
  vr = i * r;
  vl = i * zl;
  vc = i * zc;

  cout << "Z_L = " << abs (zl) << " Ohms @ " << arg (zl) * 180 /
    M_PI << " deg " << endl;
  cout << "Z_C = " << abs (zc) << " Ohms @ " << arg (zc) * 180 /
    M_PI << " deg " << endl;
  cout << "Z_total = " << abs (zseries) << " Ohms @ " << arg (zseries) * 180 /
    M_PI << " deg " << endl;

  cout << endl;

  cout << "Current = " << abs (i) * 1e3 << " mA @ " << arg (i) * 180 /
```

```
    M_PI << " deg " << endl;
  cout << "Resistor voltage = " << abs (vr) << " V @ " << arg (vr) * 180 /
    M_PI << " deg " << endl;
  cout << "Inductor voltage = " << abs (vl) << " V @ " << arg (vl) * 180 /
    M_PI << " deg " << endl;
  cout << "Capacitor voltage = " << abs (vc) << " V @ " << arg (vc) * 180 /
    M_PI << " deg " << endl;

  cout << endl;

  cout << "Apparent power = " << abs (i) * abs (vsrc) << " VA " << endl;
  cout << "True power = " << abs (i) * abs (vsrc) *
    cos (arg (zseries)) << " W " << endl;
  cout << "Reactive power = " << abs (i) * abs (vsrc) *
    abs (sin (arg (zseries))) << " VAR " << endl;
  cout << "Power factor = " << cos (arg (zseries));

  if (arg (zseries) < 0)
    cout << " leading";
  else
    cout << " lagging";
  cout << endl;

  return 0;
}
```

When run, the output displays as follows:

```
+--V---R---L---C--+  V_source = 30 Volts
|                 |  f_source = 55 Hertz
+-----------------+  R = 30 Ohms
                     L = 0.11 Henrys
                     C = 0.0001 Farads

Z_L = 38.0133 Ohms @ 90 deg
Z_C = 28.9373 Ohms @ -90 deg
Z_total = 31.3428 Ohms @ 16.8323 deg

Current = 957.156 mA @ -16.8323 deg
Resistor voltage = 28.7147 V @ -16.8323 deg
Inductor voltage = 36.3846 V @ 73.1677 deg
Capacitor voltage = 27.6975 V @ -106.832 deg

Apparent power = 28.7147 VA
True power = 27.4844 W
Reactive power = 8.31497 VAR
Power factor = 0.957156 lagging
```

This next C++ program analyzes a series RLC circuit energized by a current source:

```
#include <iostream>
#include <complex>
#include <cmath>
using namespace std;

int main (void)
{
  double i = 7.5e-3;
  double f = 1.8e3;
  double r = 1.5e3;
  double l = 81e-3;
  double c = 0.047e-6;

  complex <double> zc, zl, zseries, vc, vl, vr, vsrc;

  cout << "+--I---R---L---C--+ " << " I_source = " << i << " Amperes " << endl;
  cout << "|                 | " << " f_source = " << f << " Hertz " << endl;
  cout << "+-----------------+ " << " R = " << r << " Ohms " << endl;
  cout << "                     L = " << l << " Henrys " << endl;
  cout << "                     C = " << c << " Farads " << endl << endl;

  zl = complex <double> (0.0, 2 * M_PI * f * l);
  zc = complex <double> (0.0, -1 / (2 * M_PI * f * c));

  zseries = zc + zl + r;
  vsrc = i * zseries;
  vr = i * r;
  vl = i * zl;
  vc = i * zc;

  cout << "Z_L = " << abs (zl) << " Ohms @ " << arg (zl) * 180 /
    M_PI << " deg " << endl;
  cout << "Z_C = " << abs (zc) << " Ohms @ " << arg (zc) * 180 /
    M_PI << " deg " << endl;
  cout << "Z_total = " << abs (zseries) << " Ohms @ " << arg (zseries) * 180 /
    M_PI << " deg " << endl;

  cout << endl;

  cout << "Source voltage = " << abs (vsrc) << " V @ " << arg (vsrc) * 180 /
    M_PI << " deg " << endl;
  cout << "Resistor voltage = " << abs (vr) << " V @ " << arg (vr) * 180 /
```

```
    M_PI << " deg " << endl;
  cout << "Inductor voltage = " << abs (vl) << " V @ " << arg (vl) * 180 /
    M_PI << " deg " << endl;
  cout << "Capacitor voltage = " << abs (vc) << " V @ " << arg (vc) * 180 /
    M_PI << " deg " << endl;

  cout << endl;

  cout << "Apparent power = " << abs (i) * abs (vsrc) << " VA " << endl;
  cout << "True power = " << abs (i) * abs (vsrc) *
    cos (arg (zseries)) << " W " << endl;
  cout << "Reactive power = " << abs (i) * abs (vsrc) *
    abs (sin (arg (zseries))) << " VAR " << endl;
  cout << "Power factor = " << cos (arg (zseries));

  if (arg (zseries) < 0)
    cout << " leading";
  else
    cout << " lagging";
  cout << endl;

  return 0;
}
```

When run, the output displays as follows:

```
+--I---R---L---C--+  I_source = 0.0075 Amperes
|                 |  f_source = 1800 Hertz
+-----------------+  R = 1500 Ohms
                     L = 0.081 Henrys
                     C = 4.7e-08 Farads

Z_L = 916.088 Ohms @ 90 deg
Z_C = 1881.26 Ohms @ -90 deg
Z_total = 1783.69 Ohms @ -32.7593 deg

Source voltage = 13.3777 V @ -32.7593 deg
Resistor voltage = 11.25 V @ 0 deg
Inductor voltage = 6.87066 V @ 90 deg
Capacitor voltage = 14.1095 V @ -90 deg

Apparent power = 0.100333 VA
True power = 0.084375 W
Reactive power = 0.0542911 VAR
Power factor = 0.840951 leading
```

Chapter 6

Questions

This learning module, along with all others in the ModEL collection, is designed to be used in an inverted instructional environment where students independently read[1] the tutorials and attempt to answer questions on their own *prior* to the instructor's interaction with them. In place of lecture[2], the instructor engages with students in Socratic-style dialogue, probing and challenging their understanding of the subject matter through inquiry.

Answers are not provided for questions within this chapter, and this is by design. Solved problems may be found in the Tutorial and Derivation chapters, instead. The goal here is *independence*, and this requires students to be challenged in ways where others cannot think for them. Remember that you always have the tools of *experimentation* and *computer simulation* (e.g. SPICE) to explore concepts!

The following lists contain ideas for Socratic-style questions and challenges. Upon inspection, one will notice a strong theme of *metacognition* within these statements: they are designed to foster a regular habit of examining one's own thoughts as a means toward clearer thinking. As such these sample questions are useful both for instructor-led discussions as well as for self-study.

[1] Technical reading is an essential academic skill for any technical practitioner to possess for the simple reason that the most comprehensive, accurate, and useful information to be found for developing technical competence is in textual form. Technical careers in general are characterized by the need for continuous learning to remain current with standards and technology, and therefore any technical practitioner who cannot read well is handicapped in their professional development. An excellent resource for educators on improving students' reading prowess through intentional effort and strategy is the book textitReading For Understanding – How Reading Apprenticeship Improves Disciplinary Learning in Secondary and College Classrooms by Ruth Schoenbach, Cynthia Greenleaf, and Lynn Murphy.

[2] Lecture is popular as a teaching method because it is easy to implement: any reasonably articulate subject matter expert can talk to students, even with little preparation. However, it is also quite problematic. A good lecture always makes complicated concepts seem easier than they are, which is bad for students because it instills a false sense of confidence in their own understanding; reading and re-articulation requires more cognitive effort and serves to verify comprehension. A culture of teaching-by-lecture fosters a debilitating dependence upon direct personal instruction, whereas the challenges of modern life demand independent and critical thought made possible only by gathering information and perspectives from afar. Information presented in a lecture is ephemeral, easily lost to failures of memory and dictation; text is forever, and may be referenced at any time.

General challenges following tutorial reading

- Summarize as much of the text as you can in one paragraph of your own words. A helpful strategy is to explain ideas as you would for an intelligent child: as simple as you can without compromising too much accuracy.
- Simplify a particular section of the text, for example a paragraph or even a single sentence, so as to capture the same fundamental idea in fewer words.
- Where did the text make the most sense to you? What was it about the text's presentation that made it clear?
- Identify where it might be easy for someone to misunderstand the text, and explain why you think it could be confusing.
- Identify any new concept(s) presented in the text, and explain in your own words.
- Identify any familiar concept(s) such as physical laws or principles applied or referenced in the text.
- Devise a proof of concept experiment demonstrating an important principle, physical law, or technical innovation represented in the text.
- Devise an experiment to disprove a plausible misconception.
- Did the text reveal any misconceptions you might have harbored? If so, describe the misconception(s) and the reason(s) why you now know them to be incorrect.
- Describe any useful problem-solving strategies applied in the text.
- Devise a question of your own to challenge a reader's comprehension of the text.

General follow-up challenges for assigned problems

- Identify where any fundamental laws or principles apply to the solution of this problem, especially before applying any mathematical techniques.
- Devise a thought experiment to explore the characteristics of the problem scenario, applying known laws and principles to mentally model its behavior.
- Describe in detail your own strategy for solving this problem. How did you identify and organized the given information? Did you sketch any diagrams to help frame the problem?
- Is there more than one way to solve this problem? Which method seems best to you?
- Show the work you did in solving this problem, even if the solution is incomplete or incorrect.
- What would you say was the most challenging part of this problem, and why was it so?
- Was any important information missing from the problem which you had to research or recall?
- Was there any extraneous information presented within this problem? If so, what was it and why did it not matter?
- Examine someone else's solution to identify where they applied fundamental laws or principles.
- Simplify the problem from its given form and show how to solve this simpler version of it. Examples include eliminating certain variables or conditions, altering values to simpler (usually whole) numbers, applying a limiting case (i.e. altering a variable to some extreme or ultimate value).
- For quantitative problems, identify the real-world meaning of all intermediate calculations: their units of measurement, where they fit into the scenario at hand. Annotate any diagrams or illustrations with these calculated values.
- For quantitative problems, try approaching it qualitatively instead, thinking in terms of "increase" and "decrease" rather than definite values.
- For qualitative problems, try approaching it quantitatively instead, proposing simple numerical values for the variables.
- Were there any assumptions you made while solving this problem? Would your solution change if one of those assumptions were altered?
- Identify where it would be easy for someone to go astray in attempting to solve this problem.
- Formulate your own problem based on what you learned solving this one.

General follow-up challenges for experiments or projects

- In what way(s) was this experiment or project easy to complete?
- Identify some of the challenges you faced in completing this experiment or project.

- Show how thorough documentation assisted in the completion of this experiment or project.
- Which fundamental laws or principles are key to this system's function?
- Identify any way(s) in which one might obtain false or otherwise misleading measurements from test equipment in this system.
- What will happen if (component X) fails (open/shorted/etc.)?
- What would have to occur to make this system unsafe?

6.1 Conceptual reasoning

These questions are designed to stimulate your analytic and synthetic thinking[3]. In a Socratic discussion with your instructor, the goal is for these questions to prompt an extended dialogue where assumptions are revealed, conclusions are tested, and understanding is sharpened. Your instructor may also pose additional questions based on those assigned, in order to further probe and refine your conceptual understanding.

Questions that follow are presented to challenge and probe your understanding of various concepts presented in the tutorial. These questions are intended to serve as a guide for the Socratic dialogue between yourself and the instructor. Your instructor's task is to ensure you have a sound grasp of these concepts, and the questions contained in this document are merely a means to this end. Your instructor may, at his or her discretion, alter or substitute questions for the benefit of tailoring the discussion to each student's needs. The only absolute requirement is that each student is challenged and assessed at a level equal to or greater than that represented by the documented questions.

It is far more important that you convey your *reasoning* than it is to simply convey a correct answer. For this reason, you should refrain from researching other information sources to answer questions. What matters here is that *you* are doing the thinking. If the answer is incorrect, your instructor will work with you to correct it through proper reasoning. A correct answer without an adequate explanation of how you derived that answer is unacceptable, as it does not aid the learning or assessment process.

You will note a conspicuous lack of answers given for these conceptual questions. Unlike standard textbooks where answers to every other question are given somewhere toward the back of the book, here in these learning modules students must rely on other means to check their work. The best way by far is to debate the answers with fellow students and also with the instructor during the Socratic dialogue sessions intended to be used with these learning modules. Reasoning through challenging questions with other people is an excellent tool for developing strong reasoning skills.

Another means of checking your conceptual answers, where applicable, is to use circuit simulation software to explore the effects of changes made to circuits. For example, if one of these conceptual questions challenges you to predict the effects of altering some component parameter in a circuit, you may check the validity of your work by simulating that same parameter change within software and seeing if the results agree.

[3] *Analytical* thinking involves the "disassembly" of an idea into its constituent parts, analogous to dissection. *Synthetic* thinking involves the "assembly" of a new idea comprised of multiple concepts, analogous to construction. Both activities are high-level cognitive skills, extremely important for effective problem-solving, necessitating frequent challenge and regular practice to fully develop.

6.1.1 Reading outline and reflections

"Reading maketh a full man; conference a ready man; and writing an exact man" – Francis Bacon

Francis Bacon's advice is a blueprint for effective education: reading provides the learner with knowledge, writing focuses the learner's thoughts, and critical dialogue equips the learner to confidently communicate and apply their learning. Independent acquisition and application of knowledge is a powerful skill, well worth the effort to cultivate. To this end, students should read these educational resources closely, journal their own reflections on the reading, and discuss in detail their findings with classmates and instructor(s). You should be able to do all of the following after reading any instructional text:

☑ Briefly SUMMARIZE THE TEXT in the form of a journal entry documenting your learning as you progress through the course of study. Share this summary in dialogue with your classmates and instructor. Journaling is an excellent self-test of thorough reading because you cannot clearly express what you have not read or did not comprehend.

☑ Demonstrate ACTIVE READING STRATEGIES, including verbalizing your impressions as you read, simplifying long passages to convey the same ideas using fewer words, annotating text and illustrations with your own interpretations, working through mathematical examples shown in the text, cross-referencing passages with relevant illustrations and/or other passages, identifying problem-solving strategies applied by the author, etc. Technical reading is a special case of problem-solving, and so these strategies work precisely because they help solve any problem: paying attention to your own thoughts (metacognition), eliminating unnecessary complexities, identifying what makes sense, paying close attention to details, drawing connections between separated facts, and noting the successful strategies of others.

☑ Identify IMPORTANT THEMES, especially GENERAL LAWS and PRINCIPLES, expounded in the text and express them in the simplest of terms as though you were teaching an intelligent child. This emphasizes connections between related topics and develops your ability to communicate complex ideas to anyone.

☑ Form YOUR OWN QUESTIONS based on the reading, and then pose them to your instructor and classmates for their consideration. Anticipate both correct and incorrect answers, the incorrect answer(s) assuming one or more plausible misconceptions. This helps you view the subject from different perspectives to grasp it more fully.

☑ Devise EXPERIMENTS to test claims presented in the reading, or to disprove misconceptions. Predict possible outcomes of these experiments, and evaluate their meanings: what result(s) would confirm, and what would constitute disproof? Running mental simulations and evaluating results is essential to scientific and diagnostic reasoning.

☑ Specifically identify any points you found CONFUSING. The reason for doing this is to help diagnose misconceptions and overcome barriers to learning.

6.1.2 Foundational concepts

Correct analysis and diagnosis of electric circuits begins with a proper understanding of some basic concepts. The following is a list of some important concepts referenced in this module's full tutorial. Define each of them in your own words, and be prepared to illustrate each of these concepts with a description of a practical example and/or a live demonstration.

Energy

Conservation of Energy

Voltage

Current

Source

Load

Frequency

RMS

Phasor

Resistance

Reactance

Impedance

Ohm's Law

Kirchhoff's Voltage Law

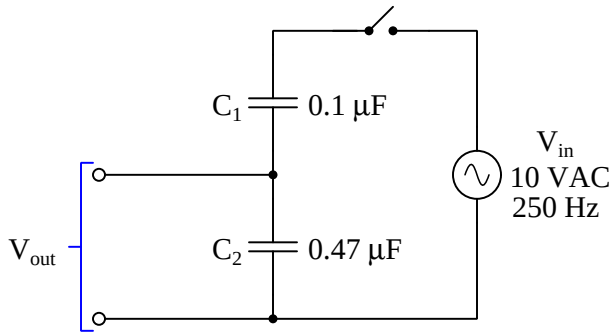
Kirchhoff's Current Law

Series network

Parallel network

6.1.3 Capacitive voltage divider

Voltage divider circuits may be constructed from reactive components just as easily as they may be constructed from resistors. Take this capacitive voltage divider, for instance:



Describe what advantages a capacitive voltage divider might have over a resistive voltage divider.

Challenges

- Calculate V_{out} in this circuit.
- Will the capacitors introduce a phase shift to the divided voltage signal?

6.2 Quantitative reasoning

These questions are designed to stimulate your computational thinking. In a Socratic discussion with your instructor, the goal is for these questions to reveal your mathematical approach(es) to problem-solving so that good technique and sound reasoning may be reinforced. Your instructor may also pose additional questions based on those assigned, in order to observe your problem-solving firsthand.

Mental arithmetic and estimations are strongly encouraged for all calculations, because without these abilities you will be unable to readily detect errors caused by calculator misuse (e.g. keystroke errors).

You will note a conspicuous lack of answers given for these quantitative questions. Unlike standard textbooks where answers to every other question are given somewhere toward the back of the book, here in these learning modules students must rely on other means to check their work. My advice is to use circuit simulation software such as SPICE to check the correctness of quantitative answers. Refer to those learning modules within this collection focusing on SPICE to see worked examples which you may use directly as practice problems for your own study, and/or as templates you may modify to run your own analyses and generate your own practice problems.

Completely worked example problems found in the Tutorial may also serve as "test cases[4]" for gaining proficiency in the use of circuit simulation software, and then once that proficiency is gained you will never need to rely[5] on an answer key!

[4] In other words, set up the circuit simulation software to analyze the same circuit examples found in the Tutorial. If the simulated results match the answers shown in the Tutorial, it confirms the simulation has properly run. If the simulated results disagree with the Tutorial's answers, something has been set up incorrectly in the simulation software. Using every Tutorial as practice in this way will quickly develop proficiency in the use of circuit simulation software.

[5] This approach is perfectly in keeping with the instructional philosophy of these learning modules: *teaching students to be self-sufficient thinkers*. Answer keys can be useful, but it is even more useful to your long-term success to have a set of tools on hand for checking your own work, because once you have left school and are on your own, there will no longer be "answer keys" available for the problems you will have to solve.

6.2.1 Miscellaneous physical constants

Note: constants shown in **bold** type are *exact*, not approximations. Values inside of parentheses show one standard deviation (σ) of uncertainty in the final digits: for example, the magnetic permeability of free space value given as $1.25663706212(19) \times 10^{-6}$ H/m represents a center value (i.e. the location parameter) of $1.25663706212 \times 10^{-6}$ Henrys per meter with one standard deviation of uncertainty equal to $0.0000000000019 \times 10^{-6}$ Henrys per meter.

Avogadro's number (N_A) = **6.02214076** $\times 10^{23}$ **per mole** (mol^{-1})

Boltzmann's constant (k) = **1.380649** $\times 10^{-23}$ **Joules per Kelvin** (J/K)

Electronic charge (e) = **1.602176634** $\times 10^{-19}$ **Coulomb** (C)

Faraday constant (F) = **96,485.33212...** $\times 10^{4}$ **Coulombs per mole** (C/mol)

Magnetic permeability of free space (μ_0) = $1.25663706212(19) \times 10^{-6}$ Henrys per meter (H/m)

Electric permittivity of free space (ϵ_0) = $8.8541878128(13) \times 10^{-12}$ Farads per meter (F/m)

Characteristic impedance of free space (Z_0) = 376.730313668(57) Ohms (Ω)

Gravitational constant (G) = $6.67430(15) \times 10^{-11}$ cubic meters per kilogram-seconds squared ($m^3/kg\text{-}s^2$)

Molar gas constant (R) = **8.314462618... Joules per mole-Kelvin** (J/mol-K) = 0.08205746(14) liters-atmospheres per mole-Kelvin

Planck constant (h) = **6.62607015** $\times 10^{-34}$ **joule-seconds** (J-s)

Stefan-Boltzmann constant (σ) = **5.670374419...** $\times 10^{-8}$ **Watts per square meter-Kelvin**^{4} ($W/m^2 \cdot K^4$)

Speed of light in a vacuum (c) = **299,792,458 meters per second** (m/s) = 186282.4 miles per second (mi/s)

Note: All constants taken from NIST data "Fundamental Physical Constants – Complete Listing", from `http://physics.nist.gov/constants`, National Institute of Standards and Technology (NIST), 2018 CODATA Adjustment.

6.2.2 Introduction to spreadsheets

A powerful computational tool you are encouraged to use in your work is a *spreadsheet*. Available on most personal computers (e.g. Microsoft Excel), *spreadsheet* software performs numerical calculations based on number values and formulae entered into cells of a grid. This grid is typically arranged as lettered columns and numbered rows, with each cell of the grid identified by its column/row coordinates (e.g. cell `B3`, cell `A8`). Each cell may contain a string of text, a number value, or a mathematical formula. The spreadsheet automatically updates the results of all mathematical formulae whenever the entered number values are changed. This means it is possible to set up a spreadsheet to perform a series of calculations on entered data, and those calculations will be re-done by the computer any time the data points are edited in any way.

For example, the following spreadsheet calculates average speed based on entered values of distance traveled and time elapsed:

	A	B	C	D
1	Distance traveled	46.9	Kilometers	
2	Time elapsed	1.18	Hours	
3	Average speed	= B1 / B2	km/h	
4				
5				

Text labels contained in cells `A1` through `A3` and cells `C1` through `C3` exist solely for readability and are not involved in any calculations. Cell `B1` contains a sample distance value while cell `B2` contains a sample time value. The formula for computing speed is contained in cell `B3`. Note how this formula begins with an "equals" symbol (`=`), references the values for distance and speed by lettered column and numbered row coordinates (`B1` and `B2`), and uses a forward slash symbol for division (`/`). The coordinates `B1` and `B2` function as *variables*[6] would in an algebraic formula.

When this spreadsheet is executed, the numerical value `39.74576` will appear in cell `B3` rather than the formula `= B1 / B2`, because 39.74576 is the computed speed value given 46.9 kilometers traveled over a period of 1.18 hours. If a different numerical value for distance is entered into cell `B1` or a different value for time is entered into cell `B2`, cell `B3`'s value will automatically update. All you need to do is set up the given values and any formulae into the spreadsheet, and the computer will do all the calculations for you.

Cell `B3` may be referenced by other formulae in the spreadsheet if desired, since it is a variable just like the given values contained in `B1` and `B2`. This means it is possible to set up an entire chain of calculations, one dependent on the result of another, in order to arrive at a final value. The arrangement of the given data and formulae need not follow any pattern on the grid, which means you may place them anywhere.

[6] Spreadsheets may also provide means to attach text labels to cells for use as variable names (Microsoft Excel simply calls these labels "names"), but for simple spreadsheets such as those shown here it's usually easier just to use the standard coordinate naming for each cell.

Common[7] arithmetic operations available for your use in a spreadsheet include the following:

- Addition (`+`)
- Subtraction (`-`)
- Multiplication (`*`)
- Division (`/`)
- Powers (`^`)
- Square roots (`sqrt()`)
- Logarithms (`ln()` , `log10()`)

Parentheses may be used to ensure[8] proper order of operations within a complex formula. Consider this example of a spreadsheet implementing the *quadratic formula*, used to solve for roots of a polynomial expression in the form of $ax^2 + bx + c$:

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

	A	B
1	x_1	= (-B4 + sqrt((B4^2) - (4*B3*B5))) / (2*B3)
2	x_2	= (-B4 - sqrt((B4^2) - (4*B3*B5))) / (2*B3)
3	a =	9
4	b =	5
5	c =	-2

This example is configured to compute roots[9] of the polynomial $9x^2 + 5x - 2$ because the values of 9, 5, and -2 have been inserted into cells `B3`, `B4`, and `B5`, respectively. Once this spreadsheet has been built, though, it may be used to calculate the roots of *any* second-degree polynomial expression simply by entering the new a, b, and c coefficients into cells `B3` through `B5`. The numerical values appearing in cells `B1` and `B2` will be automatically updated by the computer immediately following any changes made to the coefficients.

[7] Modern spreadsheet software offers a bewildering array of mathematical functions you may use in your computations. I recommend you consult the documentation for your particular spreadsheet for information on operations other than those listed here.

[8] Spreadsheet programs, like text-based programming languages, are designed to follow standard order of operations by default. However, my personal preference is to use parentheses even where strictly unnecessary just to make it clear to any other person viewing the formula what the intended order of operations is.

[9] Reviewing some algebra here, a *root* is a value for x that yields an overall value of zero for the polynomial. For this polynomial ($9x^2 + 5x - 2$) the two roots happen to be $x = 0.269381$ and $x = -0.82494$, with these values displayed in cells `B1` and `B2`, respectively upon execution of the spreadsheet.

Alternatively, one could break up the long quadratic formula into smaller pieces like this:

$$y = \sqrt{b^2 - 4ac} \qquad z = 2a$$

$$x = \frac{-b \pm y}{z}$$

	A	B	C
1	x_1	= (-B4 + C1) / C2	= sqrt((B4^2) - (4*B3*B5))
2	x_2	= (-B4 - C1) / C2	= 2*B3
3	a =	9	
4	b =	5	
5	c =	-2	

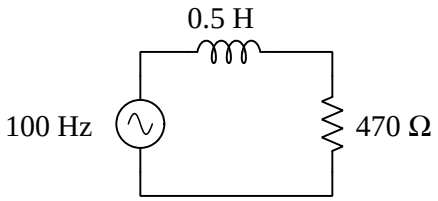
Note how the square-root term (y) is calculated in cell C1, and the denominator term (z) in cell C2. This makes the two final formulae (in cells B1 and B2) simpler to interpret. The positioning of all these cells on the grid is completely arbitrary[10] – all that matters is that they properly reference each other in the formulae.

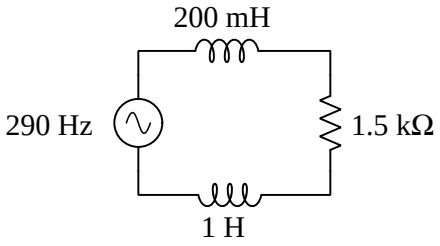
Spreadsheets are particularly useful for situations where the same set of calculations representing a circuit or other system must be repeated for different initial conditions. The power of a spreadsheet is that it automates what would otherwise be a tedious set of calculations. One specific application of this is to simulate the effects of various components within a circuit failing with abnormal values (e.g. a shorted resistor simulated by making its value nearly zero; an open resistor simulated by making its value extremely large). Another application is analyzing the behavior of a circuit design given new components that are out of specification, and/or aging components experiencing drift over time.

[10] My personal preference is to locate all the "given" data in the upper-left cells of the spreadsheet grid (each data point flanked by a sensible name in the cell to the left and units of measurement in the cell to the right as illustrated in the first distance/time spreadsheet example), sometimes coloring them in order to clearly distinguish which cells contain entered data versus which cells contain computed results from formulae. I like to place all formulae in cells below the given data, and try to arrange them in logical order so that anyone examining my spreadsheet will be able to figure out *how* I constructed a solution. This is a general principle I believe all computer programmers should follow: *document and arrange your code to make it easy for other people to learn from it.*

6.2.3 Series RL impedances

Calculate the total impedances (complete with phase angles) for each of the following inductor-resistor circuits:



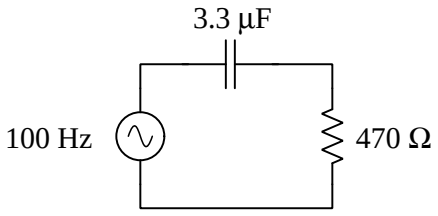


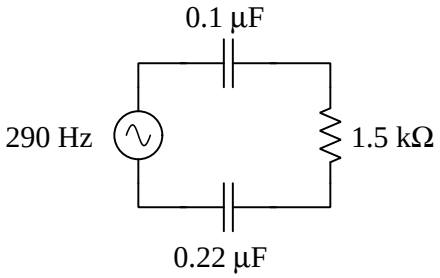
Challenges

- Predict the effect on impedance magnitude and phase angle if the frequency of each source decreases.

6.2.4 Series RC impedances

Calculate the total impedances (complete with phase angles) for each of the following capacitor-resistor circuits:



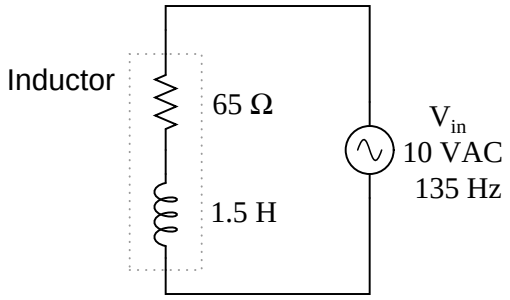


Challenges

- Predict the effect on impedance magnitude and phase angle if the frequency of each source decreases.

6.2.5 Inductor with winding resistance

Calculate the magnitude and phase shift of the current through this inductor, taking into consideration its intrinsic winding resistance:

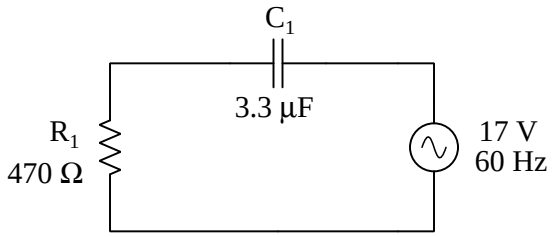


Challenges

- Why are inductors generally considered less "pure" than capacitors?

6.2.6 VIZ table for series RC circuit

Complete the table of values for this circuit, representing all quantities in complex-number form (either polar or rectangular, your choice):



	R_1	C_1	Total
V			$17\ V \angle 0^o$
I			
Z			

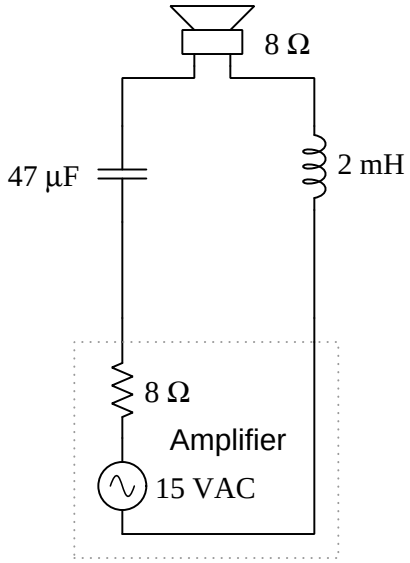
Lastly, calculate power factor, apparent power, reactive power, and true power for this circuit.

Challenges

- Sketch a phasor diagram for all additive quantities in this circuit, showing graphically how their sum equals the sum shown in the table.
- At what source frequency will the phase angle in this circuit be 45^o?

6.2.7 Sound system calculations

Calculate the voltage dropped across the inductor, the capacitor, and the 8-Ohm speaker in this sound system at the following frequencies, given a constant source voltage of 15 Volts:



- $f = 200$ Hz
- $f = 550$ Hz
- $f = 900$ Hz

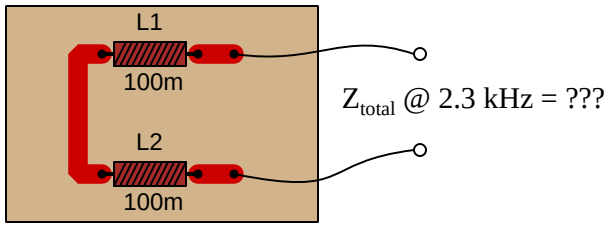
Regard the speaker as nothing more than an 8-Ohm resistor.

Challenges

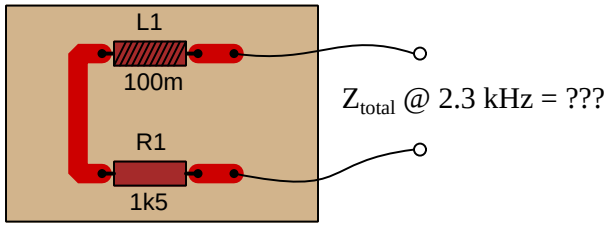
- If the source frequency were higher, would we need a smaller or larger capacitor to achieve the same total impedance?

6.2.8 LL and RL phasor diagrams

Calculate the total impedance for these two PCB-mounted 100 mH inductors at 2.3 kHz, and draw a phasor diagram showing all circuit impedances:



Now, re-calculate impedance and re-draw the phasor impedance diagram supposing the second inductor is replaced by a 1.5 kΩ resistor:

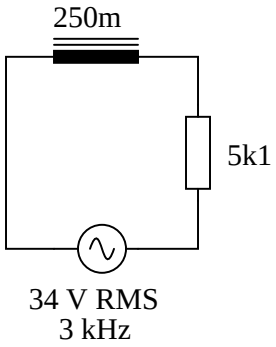


Challenges

- What is the significance of an impedance phasor pointing in a vertical direction?

6.2.9 RL network and phasor diagram

Calculate the total impedance of this series LR circuit and then calculate the total circuit current:



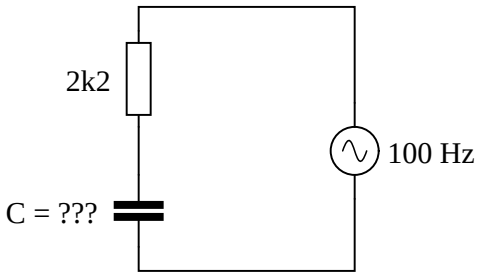
Also, draw a phasor diagram showing how the individual component impedances relate to the total impedance.

Challenges

- If the source frequency were higher, would we need a smaller or larger capacitor to achieve the same total impedance?

6.2.10 Unknown capacitance

Calculate the necessary size of the capacitor to give this circuit a total impedance (Z_{total}) of 4 kΩ, at a power supply frequency of 100 Hz:



Challenges

- If the source frequency were higher, would we need a smaller or larger capacitor to achieve the same total impedance?

6.3 Diagnostic reasoning

These questions are designed to stimulate your deductive and inductive thinking, where you must apply general principles to specific scenarios (deductive) and also derive conclusions about the failed circuit from specific details (inductive). In a Socratic discussion with your instructor, the goal is for these questions to reinforce your recall and use of general circuit principles and also challenge your ability to integrate multiple symptoms into a sensible explanation of what's wrong in a circuit. Your instructor may also pose additional questions based on those assigned, in order to further challenge and sharpen your diagnostic abilities.

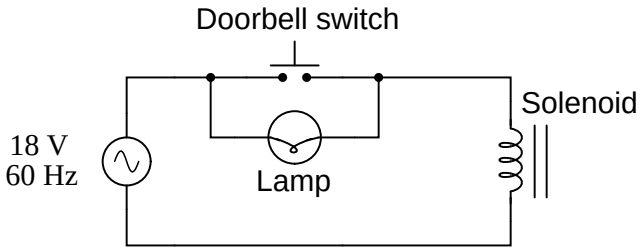
As always, your goal is to fully *explain* your analysis of each problem. Simply obtaining a correct answer is not good enough – you must also demonstrate sound reasoning in order to successfully complete the assignment. Your instructor's responsibility is to probe and challenge your understanding of the relevant principles and analytical processes in order to ensure you have a strong foundation upon which to build further understanding.

You will note a conspicuous lack of answers given for these diagnostic questions. Unlike standard textbooks where answers to every other question are given somewhere toward the back of the book, here in these learning modules students must rely on other means to check their work. The best way by far is to debate the answers with fellow students and also with the instructor during the Socratic dialogue sessions intended to be used with these learning modules. Reasoning through challenging questions with other people is an excellent tool for developing strong reasoning skills.

Another means of checking your diagnostic answers, where applicable, is to use circuit simulation software to explore the effects of faults placed in circuits. For example, if one of these diagnostic questions requires that you predict the effect of an open or a short in a circuit, you may check the validity of your work by simulating that same fault (substituting a very high resistance in place of that component for an open, and substituting a very low resistance for a short) within software and seeing if the results agree.

6.3.1 Failed doorbell

Doorbell circuits connect a small lamp in parallel with the doorbell pushbutton so that there is light at the button when it is *not* being pressed. The lamp's filament resistance is such that there is not enough current going through it to energize the solenoid coil when lit, which means the doorbell will ring only when the pushbutton switch shorts past the lamp:



Suppose that such a doorbell circuit suddenly stops working one day, and the home owner assumes the power source has quit since the bell will not ring when the button is pressed and the lamp never lights. Although a dead power source is certainly possible, it is not the only possibility. Identify another possible failure in this circuit which would result in no doorbell action (no sound) and no light at the lamp.

Challenges

- If you have identified multiple fault possibilities, determine which of these might be the most likely based on your knowledge of this circuit's construction. In other words, which of those faults might be the most likely, based on no knowledge other than how each component is made.

Appendix A

Problem-Solving Strategies

The ability to solve complex problems is arguably one of the most valuable skills one can possess, and this skill is particularly important in any science-based discipline.

- Study principles, not procedures. Don't be satisfied with merely knowing how to compute solutions – learn *why* those solutions work.
- Identify what it is you need to solve, identify all relevant data, identify all units of measurement, identify any general principles or formulae linking the given information to the solution, and then identify any "missing pieces" to a solution. Annotate all diagrams with this data.
- Sketch a diagram to help visualize the problem. When building a real system, always devise a plan for that system and analyze its function *before* constructing it.
- Follow the units of measurement and meaning of every calculation. If you are ever performing mathematical calculations as part of a problem-solving procedure, and you find yourself unable to apply each and every intermediate result to some aspect of the problem, it means you don't understand what you are doing. Properly done, every mathematical result should have practical meaning for the problem, and not just be an abstract number. You should be able to identify the proper units of measurement for each and every calculated result, and show where that result fits into the problem.
- Perform "thought experiments" to explore the effects of different conditions for theoretical problems. When troubleshooting real systems, perform *diagnostic tests* rather than visually inspecting for faults, the best diagnostic test being the one giving you the most information about the nature and/or location of the fault with the fewest steps.
- Simplify the problem until the solution becomes obvious, and then use that obvious case as a model to follow in solving the more complex version of the problem.
- Check for exceptions to see if your solution is incorrect or incomplete. A good solution will work for *all* known conditions and criteria. A good example of this is the process of testing scientific hypotheses: the task of a scientist is not to find support for a new idea, but rather to *challenge* that new idea to see if it holds up under a battery of tests. The philosophical

principle of *reductio ad absurdum* (i.e. disproving a general idea by finding a specific case where it fails) is useful here.

- Work "backward" from a hypothetical solution to a new set of given conditions.
- Add quantities to problems that are qualitative in nature, because sometimes a little math helps illuminate the scenario.
- Sketch graphs illustrating how variables relate to each other. These may be quantitative (i.e. with realistic number values) or qualitative (i.e. simply showing increases and decreases).
- Treat quantitative problems as qualitative in order to discern the relative magnitudes and/or directions of change of the relevant variables. For example, try determining what happens if a certain variable were to increase or decrease before attempting to precisely calculate quantities: how will each of the dependent variables respond, by increasing, decreasing, or remaining the same as before?
- Consider limiting cases. This works especially well for qualitative problems where you need to determine which direction a variable will change. Take the given condition and magnify that condition to an extreme degree as a way of simplifying the direction of the system's response.
- Check your work. This means regularly testing your conclusions to see if they make sense. This does *not* mean repeating the same steps originally used to obtain the conclusion(s), but rather to use some other means to check validity. Simply repeating procedures often leads to *repeating the same errors* if any were made, which is why alternative paths are better.

Appendix B

Instructional philosophy

"The unexamined circuit is not worth energizing" – Socrates (if he had taught electricity)

These learning modules, although useful for self-study, were designed to be used in a formal learning environment where a subject-matter expert challenges students to digest the content and exercise their critical thinking abilities in the answering of questions and in the construction and testing of working circuits.

The following principles inform the instructional and assessment philosophies embodied in these learning modules:

- The first goal of education is to enhance clear and independent thought, in order that every student reach their fullest potential in a highly complex and inter-dependent world. Robust reasoning is *always* more important than particulars of any subject matter, because its application is universal.
- Literacy is fundamental to independent learning and thought because text continues to be the most efficient way to communicate complex ideas over space and time. Those who cannot read with ease are limited in their ability to acquire knowledge and perspective.
- Articulate communication is fundamental to work that is complex and interdisciplinary.
- Faulty assumptions and poor reasoning are best corrected through challenge, not presentation. The rhetorical technique of *reductio ad absurdum* (disproving an assertion by exposing an absurdity) works well to discipline student's minds, not only to correct the problem at hand but also to learn how to detect and correct future errors.
- Important principles should be repeatedly explored and widely applied throughout a course of study, not only to reinforce their importance and help ensure their mastery, but also to showcase the interconnectedness and utility of knowledge.

These learning modules were expressly designed to be used in an "inverted" teaching environment[1] where students first read the introductory and tutorial chapters on their own, then individually attempt to answer the questions and construct working circuits according to the experiment and project guidelines. The instructor never lectures, but instead meets regularly with each individual student to review their progress, answer questions, identify misconceptions, and challenge the student to new depths of understanding through further questioning. Regular meetings between instructor and student should resemble a Socratic[2] dialogue, where questions serve as scalpels to dissect topics and expose assumptions. The student passes each module only after consistently demonstrating their ability to logically analyze and correctly apply all major concepts in each question or project/experiment. The instructor must be vigilant in probing each student's understanding to ensure they are truly *reasoning* and not just *memorizing*. This is why "Challenge" points appear throughout, as prompts for students to think deeper about topics and as starting points for instructor queries. Sometimes these challenge points require additional knowledge that hasn't been covered in the series to answer in full. This is okay, as the major purpose of the Challenges is to stimulate analysis and synthesis on the part of each student.

The instructor must possess enough mastery of the subject matter and awareness of students' reasoning to generate their own follow-up questions to practically any student response. Even completely correct answers given by the student should be challenged by the instructor for the purpose of having students practice articulating their thoughts and defending their reasoning. Conceptual errors committed by the student should be exposed and corrected not by direct instruction, but rather by reducing the errors to an absurdity[3] through well-chosen questions and thought experiments posed by the instructor. Becoming proficient at this style of instruction requires time and dedication, but the positive effects on critical thinking for both student and instructor are spectacular.

An inspection of these learning modules reveals certain unique characteristics. One of these is a bias toward thorough explanations in the tutorial chapters. Without a live instructor to explain concepts and applications to students, the text itself must fulfill this role. This philosophy results in lengthier explanations than what you might typically find in a textbook, each step of the reasoning process fully explained, including footnotes addressing common questions and concerns students raise while learning these concepts. Each tutorial seeks to not only explain each major concept in sufficient detail, but also to explain the logic of each concept and how each may be developed

[1] In a traditional teaching environment, students first encounter new information via *lecture* from an expert, and then independently apply that information via *homework*. In an "inverted" course of study, students first encounter new information via *homework*, and then independently apply that information under the scrutiny of an expert. The expert's role in lecture is to simply *explain*, but the expert's role in an inverted session is to *challenge*, *critique*, and if necessary *explain* where gaps in understanding still exist.

[2] Socrates is a figure in ancient Greek philosophy famous for his unflinching style of questioning. Although he authored no texts, he appears as a character in Plato's many writings. The essence of Socratic philosophy is to leave no question unexamined and no point of view unchallenged. While purists may argue a topic such as electric circuits is too narrow for a true Socratic-style dialogue, I would argue that the essential thought processes involved with scientific reasoning on *any* topic are not far removed from the Socratic ideal, and that students of electricity and electronics would do very well to challenge assumptions, pose thought experiments, identify fallacies, and otherwise employ the arsenal of critical thinking skills modeled by Socrates.

[3] This rhetorical technique is known by the Latin phrase *reductio ad absurdum*. The concept is to expose errors by counter-example, since only one solid counter-example is necessary to disprove a universal claim. As an example of this, consider the common misconception among beginning students of electricity that voltage cannot exist without current. One way to apply *reductio ad absurdum* to this statement is to ask how much current passes through a fully-charged battery connected to nothing (i.e. a clear example of voltage existing without current).

from "first principles". Again, this reflects the goal of developing clear and independent thought in students' minds, by showing how clear and logical thought was used to forge each concept. Students benefit from witnessing a model of clear thinking in action, and these tutorials strive to be just that.

Another characteristic of these learning modules is a lack of step-by-step instructions in the Project and Experiment chapters. Unlike many modern workbooks and laboratory guides where step-by-step instructions are prescribed for each experiment, these modules take the approach that students must learn to closely read the tutorials and apply their own reasoning to identify the appropriate experimental steps. Sometimes these steps are plainly declared in the text, just not as a set of enumerated points. At other times certain steps are implied, an example being assumed competence in test equipment use where the student should not need to be told *again* how to use their multimeter because that was thoroughly explained in previous lessons. In some circumstances no steps are given at all, leaving the entire procedure up to the student.

This lack of prescription is not a flaw, but rather a feature. Close reading and clear thinking are foundational principles of this learning series, and in keeping with this philosophy all activities are designed to *require* those behaviors. Some students may find the lack of prescription frustrating, because it demands more from them than what their previous educational experiences required. This frustration should be interpreted as an unfamiliarity with autonomous thinking, a problem which must be corrected if the student is ever to become a self-directed learner and effective problem-solver. Ultimately, the need for students to read closely and think clearly is more important both in the near-term and far-term than any specific facet of the subject matter at hand. If a student takes longer than expected to complete a module because they are forced to outline, digest, and reason on their own, so be it. The future gains enjoyed by developing this mental discipline will be well worth the additional effort and delay.

Another feature of these learning modules is that they do not treat topics in isolation. Rather, important concepts are introduced early in the series, and appear repeatedly as stepping-stones toward other concepts in subsequent modules. This helps to avoid the "compartmentalization" of knowledge, demonstrating the inter-connectedness of concepts and simultaneously reinforcing them. Each module is fairly complete in itself, reserving the beginning of its tutorial to a review of foundational concepts.

This methodology of assigning text-based modules to students for digestion and then using Socratic dialogue to assess progress and hone students' thinking was developed over a period of several years by the author with his Electronics and Instrumentation students at the two-year college level. While decidedly unconventional and sometimes even unsettling for students accustomed to a more passive lecture environment, this instructional philosophy has proven its ability to convey conceptual mastery, foster careful analysis, and enhance employability so much better than lecture that the author refuses to ever teach by lecture again.

Problems which often go undiagnosed in a lecture environment are laid bare in this "inverted" format where students must articulate and logically defend their reasoning. This, too, may be unsettling for students accustomed to lecture sessions where the instructor cannot tell for sure who comprehends and who does not, and this vulnerability necessitates sensitivity on the part of the "inverted" session instructor in order that students never feel discouraged by having their errors exposed. *Everyone* makes mistakes from time to time, and learning is a lifelong process! Part of the instructor's job is to build a culture of learning among the students where errors are not seen as shameful, but rather as opportunities for progress.

To this end, instructors managing courses based on these modules should adhere to the following principles:

- Student questions are always welcome and demand thorough, honest answers. The only type of question an instructor should refuse to answer is one the student should be able to easily answer on their own. Remember, *the fundamental goal of education is for each student to learn to think clearly and independently.* This requires hard work on the part of the student, which no instructor should ever circumvent. Anything done to bypass the student's responsibility to do that hard work ultimately limits that student's potential and thereby does real harm.

- It is not only permissible, but encouraged, to answer a student's question by asking questions in return, these follow-up questions designed to guide the student to reach a correct answer through their own reasoning.

- All student answers demand to be challenged by the instructor and/or by other students. This includes both correct and incorrect answers – the goal is to practice the articulation and defense of one's own reasoning.

- No reading assignment is deemed complete unless and until the student demonstrates their ability to accurately summarize the major points in their own terms. Recitation of the original text is unacceptable. This is why every module contains an "Outline and reflections" question as well as a "Foundational concepts" question in the Conceptual reasoning section, to prompt reflective reading.

- No assigned question is deemed answered unless and until the student demonstrates their ability to consistently and correctly apply the concepts to *variations* of that question. This is why module questions typically contain multiple "Challenges" suggesting different applications of the concept(s) as well as variations on the same theme(s). Instructors are encouraged to devise as many of their own "Challenges" as they are able, in order to have a multitude of ways ready to probe students' understanding.

- No assigned experiment or project is deemed complete unless and until the student demonstrates the task in action. If this cannot be done "live" before the instructor, video-recordings showing the demonstration are acceptable. All relevant safety precautions must be followed, all test equipment must be used correctly, and the student must be able to properly explain all results. The student must also successfully answer all Challenges presented by the instructor for that experiment or project.

Students learning from these modules would do well to abide by the following principles:

- No text should be considered fully and adequately read unless and until you can express every idea *in your own words, using your own examples.*
- You should always articulate your thoughts as you read the text, noting points of agreement, confusion, and epiphanies. Feel free to print the text on paper and then write your notes in the margins. Alternatively, keep a journal for your own reflections as you read. This is truly a helpful tool when digesting complicated concepts.
- Never take the easy path of highlighting or underlining important text. Instead, *summarize* and/or *comment* on the text using your own words. This actively engages your mind, allowing you to more clearly perceive points of confusion or misunderstanding on your own.
- A very helpful strategy when learning new concepts is to place yourself in the role of a teacher, if only as a mental exercise. Either explain what you have recently learned to someone else, or at least *imagine* yourself explaining what you have learned to someone else. The simple act of having to articulate new knowledge and skill forces you to take on a different perspective, and will help reveal weaknesses in your understanding.
- Perform each and every mathematical calculation and thought experiment shown in the text on your own, referring back to the text to see that your results agree. This may seem trivial and unnecessary, but it is critically important to ensuring you actually understand what is presented, especially when the concepts at hand are complicated and easy to misunderstand. Apply this same strategy to become proficient in the use of *circuit simulation software*, checking to see if your simulated results agree with the results shown in the text.
- Above all, recognize that learning is hard work, and that a certain level of frustration is unavoidable. There are times when you will struggle to grasp some of these concepts, and that struggle is a natural thing. Take heart that it will yield with persistent and varied[4] effort, and never give up!

Students interested in using these modules for self-study will also find them beneficial, although the onus of responsibility for thoroughly reading and answering questions will of course lie with that individual alone. If a qualified instructor is not available to challenge students, a workable alternative is for students to form study groups where they challenge[5] one another.

To high standards of education,

Tony R. Kuphaldt

[4] As the old saying goes, "Insanity is trying the same thing over and over again, expecting different results." If you find yourself stumped by something in the text, you should attempt a different approach. Alter the thought experiment, change the mathematical parameters, do whatever you can to see the problem in a slightly different light, and then the solution will often present itself more readily.

[5] Avoid the temptation to simply share answers with study partners, as this is really counter-productive to learning. Always bear in mind that the answer to any question is far less important in the long run than the method(s) used to obtain that answer. The goal of education is to empower one's life through the improvement of clear and independent thought, literacy, expression, and various practical skills.

Appendix C

Tools used

I am indebted to the developers of many open-source software applications in the creation of these learning modules. The following is a list of these applications with some commentary on each.

You will notice a theme common to many of these applications: a bias toward *code*. Although I am by no means an expert programmer in any computer language, I understand and appreciate the flexibility offered by code-based applications where the user (you) enters commands into a plain ASCII text file, which the software then reads and processes to create the final output. Code-based computer applications are by their very nature *extensible*, while WYSIWYG (What You See Is What You Get) applications are generally limited to whatever user interface the developer makes for you.

The `GNU/Linux` computer operating system

> There is so much to be said about Linus Torvalds' `Linux` and Richard Stallman's `GNU` project. First, to credit just these two individuals is to fail to do justice to the *mob* of passionate volunteers who contributed to make this amazing software a reality. I first learned of `Linux` back in 1996, and have been using this operating system on my personal computers almost exclusively since then. It is *free*, it is completely *configurable*, and it permits the continued use of highly efficient `Unix` applications and scripting languages (e.g. shell scripts, Makefiles, `sed`, `awk`) developed over many decades. `Linux` not only provided me with a powerful computing platform, but its open design served to inspire my life's work of creating open-source educational resources.

Bram Moolenaar's `Vim` text editor

> Writing code for any code-based computer application requires a *text editor*, which may be thought of as a word processor strictly limited to outputting plain-ASCII text files. Many good text editors exist, and one's choice of text editor seems to be a deeply personal matter within the programming world. I prefer `Vim` because it operates very similarly to `vi` which is ubiquitous on `Unix`/`Linux` operating systems, and because it may be entirely operated via keyboard (i.e. no mouse required) which makes it fast to use.

Donald Knuth's TEX typesetting system

Developed in the late 1970's and early 1980's by computer scientist extraordinaire Donald Knuth to typeset his multi-volume magnum opus *The Art of Computer Programming*, this software allows the production of formatted text for screen-viewing or paper printing, all by writing plain-text code to describe how the formatted text is supposed to appear. TEX is not just a markup language for documents, but it is also a Turing-complete programming language in and of itself, allowing useful algorithms to be created to control the production of documents. Simply put, TEX *is a programmer's approach to word processing*. Since TEX is controlled by code written in a plain-text file, this means anyone may read that plain-text file to see exactly how the document was created. This openness afforded by the code-based nature of TEX makes it relatively easy to learn how other people have created their own TEX documents. By contrast, examining a beautiful document created in a conventional WYSIWYG word processor such as Microsoft `Word` suggests nothing to the reader about *how* that document was created, or what the user might do to create something similar. As Mr. Knuth himself once quipped, conventional word processing applications should be called WYSIAYG (What You See Is *All* You Get).

Leslie Lamport's LATEX extensions to TEX

Like all true programming languages, TEX is inherently extensible. So, years after the release of TEX to the public, Leslie Lamport decided to create a massive extension allowing easier compilation of book-length documents. The result was LATEX, which is the markup language used to create all ModEL module documents. You could say that TEX is to LATEX as `C` is to `C++`. This means it is permissible to use any and all TEX commands within LATEX source code, and it all still works. Some of the features offered by LATEX that would be challenging to implement in TEX include automatic index and table-of-content creation.

Tim Edwards' `Xcircuit` drafting program

This wonderful program is what I use to create all the schematic diagrams and illustrations (but not photographic images or mathematical plots) throughout the ModEL project. It natively outputs PostScript format which is a true vector graphic format (this is why the images do not pixellate when you zoom in for a closer view), and it is so simple to use that I have never had to read the manual! Object libraries are easy to create for `Xcircuit`, being plain-text files using PostScript programming conventions. Over the years I have collected a large set of object libraries useful for drawing electrical and electronic schematics, pictorial diagrams, and other technical illustrations.

`Gimp` graphic image manipulation program

Essentially an open-source clone of Adobe's `PhotoShop`, I use `Gimp` to resize, crop, and convert file formats for all of the photographic images appearing in the ModEL modules. Although `Gimp` does offer its own scripting language (called `Script-Fu`), I have never had occasion to use it. Thus, my utilization of `Gimp` to merely crop, resize, and convert graphic images is akin to using a sword to slice bread.

`SPICE` circuit simulation program

`SPICE` is to circuit analysis as TEX is to document creation: it is a form of markup language designed to describe a certain object to be processed in plain-ASCII text. When the plain-text "source file" is compiled by the software, it outputs the final result. More modern circuit analysis tools certainly exist, but I prefer `SPICE` for the following reasons: it is *free*, it is *fast*, it is *reliable*, and it is a fantastic tool for *teaching* students of electricity and electronics how to write simple code. I happen to use rather old versions of `SPICE`, version 2g6 being my "go to" application when I only require text-based output. `NGSPICE` (version 26), which is based on Berkeley `SPICE` version 3f5, is used when I require graphical output for such things as time-domain waveforms and Bode plots. In all `SPICE` example netlists I strive to use coding conventions compatible with all `SPICE` versions.

Andrew D. Hwang's `ePiX` mathematical visualization programming library

This amazing project is a `C++` library you may link to any `C/C++` code for the purpose of generating PostScript graphic images of mathematical functions. As a completely free and open-source project, it does all the plotting I would otherwise use a Computer Algebra System (CAS) such as `Mathematica` or `Maple` to do. It should be said that `ePiX` is *not* a Computer Algebra System like `Mathematica` or `Maple`, but merely a mathematical *visualization* tool. In other words, it won't determine integrals for you (you'll have to implement that in your own `C/C++` code!), but it can graph the results, and it does so beautifully. What I really admire about `ePiX` is that it is a `C++` programming library, which means it builds on the existing power and toolset available with that programming language. Mr. Hwang could have probably developed his own stand-alone application for mathematical plotting, but by creating a `C++` library to do the same thing he accomplished something much greater.

`gnuplot` mathematical visualization software

Another open-source tool for mathematical visualization is `gnuplot`. Interestingly, this tool is *not* part of Richard Stallman's GNU project, its name being a coincidence. For this reason the authors prefer "gnu" *not* be capitalized at all to avoid confusion. This is a much "lighter-weight" alternative to a spreadsheet for plotting tabular data, and the fact that it easily outputs directly to an X11 console or a file in a number of different graphical formats (including PostScript) is very helpful. I typically set my `gnuplot` output format to default (X11 on my Linux PC) for quick viewing while I'm developing a visualization, then switch to PostScript file export once the visual is ready to include in the document(s) I'm writing. As with my use of `Gimp` to do rudimentary image editing, my use of `gnuplot` only scratches the surface of its capabilities, but the important points are that it's *free* and that it *works well*.

`Python` programming language

Both Python and C++ find extensive use in these modules as instructional aids and exercises, but I'm listing Python here as a *tool* for myself because I use it almost daily as a *calculator*. If you open a Python interpreter console and type `from math import *` you can type mathematical expressions and have it return results just as you would on a hand calculator. Complex-number (i.e. *phasor*) arithmetic is similarly supported if you include the complex-math library (`from cmath import *`). Examples of this are shown in the Programming References chapter (if included) in each module. Of course, being a fully-featured programming language, Python also supports conditionals, loops, and other structures useful for calculation of quantities. Also, running in a console environment where all entries and returned values show as text in a chronologically-ordered list makes it easy to copy-and-paste those calculations to document exactly how they were performed.

Appendix D

Creative Commons License

Creative Commons Attribution 4.0 International Public License

By exercising the Licensed Rights (defined below), You accept and agree to be bound by the terms and conditions of this Creative Commons Attribution 4.0 International Public License ("Public License"). To the extent this Public License may be interpreted as a contract, You are granted the Licensed Rights in consideration of Your acceptance of these terms and conditions, and the Licensor grants You such rights in consideration of benefits the Licensor receives from making the Licensed Material available under these terms and conditions.

Section 1 – Definitions.

a. **Adapted Material** means material subject to Copyright and Similar Rights that is derived from or based upon the Licensed Material and in which the Licensed Material is translated, altered, arranged, transformed, or otherwise modified in a manner requiring permission under the Copyright and Similar Rights held by the Licensor. For purposes of this Public License, where the Licensed Material is a musical work, performance, or sound recording, Adapted Material is always produced where the Licensed Material is synched in timed relation with a moving image.

b. **Adapter's License** means the license You apply to Your Copyright and Similar Rights in Your contributions to Adapted Material in accordance with the terms and conditions of this Public License.

c. **Copyright and Similar Rights** means copyright and/or similar rights closely related to copyright including, without limitation, performance, broadcast, sound recording, and Sui Generis Database Rights, without regard to how the rights are labeled or categorized. For purposes of this Public License, the rights specified in Section 2(b)(1)-(2) are not Copyright and Similar Rights.

d. **Effective Technological Measures** means those measures that, in the absence of proper authority, may not be circumvented under laws fulfilling obligations under Article 11 of the WIPO Copyright Treaty adopted on December 20, 1996, and/or similar international agreements.

e. **Exceptions and Limitations** means fair use, fair dealing, and/or any other exception or

limitation to Copyright and Similar Rights that applies to Your use of the Licensed Material.

f. **Licensed Material** means the artistic or literary work, database, or other material to which the Licensor applied this Public License.

g. **Licensed Rights** means the rights granted to You subject to the terms and conditions of this Public License, which are limited to all Copyright and Similar Rights that apply to Your use of the Licensed Material and that the Licensor has authority to license.

h. **Licensor** means the individual(s) or entity(ies) granting rights under this Public License.

i. **Share** means to provide material to the public by any means or process that requires permission under the Licensed Rights, such as reproduction, public display, public performance, distribution, dissemination, communication, or importation, and to make material available to the public including in ways that members of the public may access the material from a place and at a time individually chosen by them.

j. **Sui Generis Database Rights** means rights other than copyright resulting from Directive 96/9/EC of the European Parliament and of the Council of 11 March 1996 on the legal protection of databases, as amended and/or succeeded, as well as other essentially equivalent rights anywhere in the world.

k. **You** means the individual or entity exercising the Licensed Rights under this Public License. **Your** has a corresponding meaning.

Section 2 – Scope.

a. License grant.

1. Subject to the terms and conditions of this Public License, the Licensor hereby grants You a worldwide, royalty-free, non-sublicensable, non-exclusive, irrevocable license to exercise the Licensed Rights in the Licensed Material to:

A. reproduce and Share the Licensed Material, in whole or in part; and

B. produce, reproduce, and Share Adapted Material.

2. Exceptions and Limitations. For the avoidance of doubt, where Exceptions and Limitations apply to Your use, this Public License does not apply, and You do not need to comply with its terms and conditions.

3. Term. The term of this Public License is specified in Section 6(a).

4. Media and formats; technical modifications allowed. The Licensor authorizes You to exercise the Licensed Rights in all media and formats whether now known or hereafter created, and to make technical modifications necessary to do so. The Licensor waives and/or agrees not to assert any right or authority to forbid You from making technical modifications necessary to exercise the Licensed Rights, including technical modifications necessary to circumvent Effective Technological Measures.

For purposes of this Public License, simply making modifications authorized by this Section 2(a)(4) never produces Adapted Material.

5. Downstream recipients.

A. Offer from the Licensor – Licensed Material. Every recipient of the Licensed Material automatically receives an offer from the Licensor to exercise the Licensed Rights under the terms and conditions of this Public License.

B. No downstream restrictions. You may not offer or impose any additional or different terms or conditions on, or apply any Effective Technological Measures to, the Licensed Material if doing so restricts exercise of the Licensed Rights by any recipient of the Licensed Material.

6. No endorsement. Nothing in this Public License constitutes or may be construed as permission to assert or imply that You are, or that Your use of the Licensed Material is, connected with, or sponsored, endorsed, or granted official status by, the Licensor or others designated to receive attribution as provided in Section 3(a)(1)(A)(i).

b. Other rights.

1. Moral rights, such as the right of integrity, are not licensed under this Public License, nor are publicity, privacy, and/or other similar personality rights; however, to the extent possible, the Licensor waives and/or agrees not to assert any such rights held by the Licensor to the limited extent necessary to allow You to exercise the Licensed Rights, but not otherwise.

2. Patent and trademark rights are not licensed under this Public License.

3. To the extent possible, the Licensor waives any right to collect royalties from You for the exercise of the Licensed Rights, whether directly or through a collecting society under any voluntary or waivable statutory or compulsory licensing scheme. In all other cases the Licensor expressly reserves any right to collect such royalties.

Section 3 – License Conditions.

Your exercise of the Licensed Rights is expressly made subject to the following conditions.

a. Attribution.

1. If You Share the Licensed Material (including in modified form), You must:

A. retain the following if it is supplied by the Licensor with the Licensed Material:

i. identification of the creator(s) of the Licensed Material and any others designated to receive attribution, in any reasonable manner requested by the Licensor (including by pseudonym if designated);

ii. a copyright notice;

iii. a notice that refers to this Public License;

iv. a notice that refers to the disclaimer of warranties;

v. a URI or hyperlink to the Licensed Material to the extent reasonably practicable;

B. indicate if You modified the Licensed Material and retain an indication of any previous modifications; and

C. indicate the Licensed Material is licensed under this Public License, and include the text of, or the URI or hyperlink to, this Public License.

2. You may satisfy the conditions in Section 3(a)(1) in any reasonable manner based on the medium, means, and context in which You Share the Licensed Material. For example, it may be reasonable to satisfy the conditions by providing a URI or hyperlink to a resource that includes the required information.

3. If requested by the Licensor, You must remove any of the information required by Section 3(a)(1)(A) to the extent reasonably practicable.

4. If You Share Adapted Material You produce, the Adapter's License You apply must not prevent recipients of the Adapted Material from complying with this Public License.

Section 4 – Sui Generis Database Rights.

Where the Licensed Rights include Sui Generis Database Rights that apply to Your use of the Licensed Material:

a. for the avoidance of doubt, Section 2(a)(1) grants You the right to extract, reuse, reproduce, and Share all or a substantial portion of the contents of the database;

b. if You include all or a substantial portion of the database contents in a database in which You have Sui Generis Database Rights, then the database in which You have Sui Generis Database Rights (but not its individual contents) is Adapted Material; and

c. You must comply with the conditions in Section 3(a) if You Share all or a substantial portion of the contents of the database.

For the avoidance of doubt, this Section 4 supplements and does not replace Your obligations under this Public License where the Licensed Rights include other Copyright and Similar Rights.

Section 5 – Disclaimer of Warranties and Limitation of Liability.

a. Unless otherwise separately undertaken by the Licensor, to the extent possible, the Licensor offers the Licensed Material as-is and as-available, and makes no representations or warranties of any kind concerning the Licensed Material, whether express, implied, statutory, or other. This includes, without limitation, warranties of title, merchantability, fitness for a particular purpose, non-infringement, absence of latent or other defects, accuracy, or the presence or absence of errors,

whether or not known or discoverable. Where disclaimers of warranties are not allowed in full or in part, this disclaimer may not apply to You.

b. To the extent possible, in no event will the Licensor be liable to You on any legal theory (including, without limitation, negligence) or otherwise for any direct, special, indirect, incidental, consequential, punitive, exemplary, or other losses, costs, expenses, or damages arising out of this Public License or use of the Licensed Material, even if the Licensor has been advised of the possibility of such losses, costs, expenses, or damages. Where a limitation of liability is not allowed in full or in part, this limitation may not apply to You.

c. The disclaimer of warranties and limitation of liability provided above shall be interpreted in a manner that, to the extent possible, most closely approximates an absolute disclaimer and waiver of all liability.

Section 6 – Term and Termination.

a. This Public License applies for the term of the Copyright and Similar Rights licensed here. However, if You fail to comply with this Public License, then Your rights under this Public License terminate automatically.

b. Where Your right to use the Licensed Material has terminated under Section 6(a), it reinstates:

1. automatically as of the date the violation is cured, provided it is cured within 30 days of Your discovery of the violation; or

2. upon express reinstatement by the Licensor.

For the avoidance of doubt, this Section 6(b) does not affect any right the Licensor may have to seek remedies for Your violations of this Public License.

c. For the avoidance of doubt, the Licensor may also offer the Licensed Material under separate terms or conditions or stop distributing the Licensed Material at any time; however, doing so will not terminate this Public License.

d. Sections 1, 5, 6, 7, and 8 survive termination of this Public License.

Section 7 – Other Terms and Conditions.

a. The Licensor shall not be bound by any additional or different terms or conditions communicated by You unless expressly agreed.

b. Any arrangements, understandings, or agreements regarding the Licensed Material not stated herein are separate from and independent of the terms and conditions of this Public License.

Section 8 – Interpretation.

a. For the avoidance of doubt, this Public License does not, and shall not be interpreted to, reduce, limit, restrict, or impose conditions on any use of the Licensed Material that could lawfully

be made without permission under this Public License.

b. To the extent possible, if any provision of this Public License is deemed unenforceable, it shall be automatically reformed to the minimum extent necessary to make it enforceable. If the provision cannot be reformed, it shall be severed from this Public License without affecting the enforceability of the remaining terms and conditions.

c. No term or condition of this Public License will be waived and no failure to comply consented to unless expressly agreed to by the Licensor.

d. Nothing in this Public License constitutes or may be interpreted as a limitation upon, or waiver of, any privileges and immunities that apply to the Licensor or You, including from the legal processes of any jurisdiction or authority.

Creative Commons is not a party to its public licenses. Notwithstanding, Creative Commons may elect to apply one of its public licenses to material it publishes and in those instances will be considered the "Licensor." Except for the limited purpose of indicating that material is shared under a Creative Commons public license or as otherwise permitted by the Creative Commons policies published at `creativecommons.org/policies`, Creative Commons does not authorize the use of the trademark "Creative Commons" or any other trademark or logo of Creative Commons without its prior written consent including, without limitation, in connection with any unauthorized modifications to any of its public licenses or any other arrangements, understandings, or agreements concerning use of licensed material. For the avoidance of doubt, this paragraph does not form part of the public licenses.

Creative Commons may be contacted at `creativecommons.org`.

Appendix E

Version history

This is a list showing all significant additions, corrections, and other edits made to this learning module. Each entry is referenced by calendar date in reverse chronological order (newest version first), which appears on the front cover of every learning module for easy reference. Any contributors to this open-source document are listed here as well.

9 September 2024 – divided the Introduction chapter into sections, one with recommendations for students, one with a listing of challenging concepts, and one with recommendations for instructors.

10 February 2023 – added power factor calculations to the "VIZ table" quantitative question.

28 November 2022 – placed questions at the top of the itemized list in the Introduction chapter prompting students to devise experiments related to the tutorial content.

28 June 2022 – minor edits to instructor answers.

23 May 2022 – added Python calculation examples to the Case Tutorial section "Example: series RL circuit".

16 May 2022 – added a Case Tutorial section showing the effective use of a triangle-wave signal generator rather than a sine-wave signal generator to provide AC excitatin in simple RLC network experiments. It's not "perfect" like a sine-wave signal generator would (ideally) be, but the results are quite close to what they ought to be, usually within the tolerances of the components.

23 December 2021 – added a Tutorial section to provide a thematic break between the AC-DC review and the rest of the Tutorial.

8 September 2021 – added a Case Tutorial example of an RLC series circuit.

1 September 2021 – added content to the Tutorial elaborating on how to analyze AC circuits using "long-hand" mathematics rather than a calculator capable of complex-number calculations.

20 August 2021 – added a Case Tutorial example where we must solve for a component value

given a desired total impedance.

8 May 2021 – commented out or deleted empty chapters.

17-18 February 2021 – added a Programming References chapter with C++ examples showing series RLC circuits energized by both voltage and current sources.

28 January 2021 – significantly edited the Introduction chapter to make it more suitable as a pre-study guide and to provide cues useful to instructors leading "inverted" teaching sessions.

6 January 2021 – added another Case Tutorial example (RC), and added SPICE simulations to each of the Case Tutorial examples.

5 January 2021 – added Derivations and Technical References chapter.

4 January 2021 – added Case Tutorial examples and Questions.

26 May 2019 – added more detail to the review section, which caused page formatting to change a bit in the final document.

October 2018 – document first created.

Index