EE20084: Structured Programming Coursework

S. R. Pennock & J. Graham-Harper-Cater

2023-2024

Contents

1 Introduction

This programming coursework is designed to allow you to develop your structured programming skills in the context of an engineering problem using Python. The main objective of the coursework is to write a program that should analyse an electrical circuit, as described below. There are a number of extensions possible to the main objective that will allow you to gain additional marks. There is an all day laboratory session where the main aim is producing a Design Report on what is to be implemented, and one and a half days laboratory session where you should implement, debug and test the code. You are, of course, free to develop and test the code in your own time. Automated tests will be used on your submission to assess a mark and so **it is critically important** that you ensure that your code complies with the syntax and file formats described in this document.

2 The Objective

Your objective is to write a program that analyses cascade circuits, such as shown in Figure 2.1, where series and shunt impedances of any value can be connected in any order between a source and a load. This is exactly the same problem that underpins expensive circuit analysis software. For these circuits the ABCD matrix analysis scheme outlined in the Appendix is a natural, **and simple**, way to analyse the circuit.

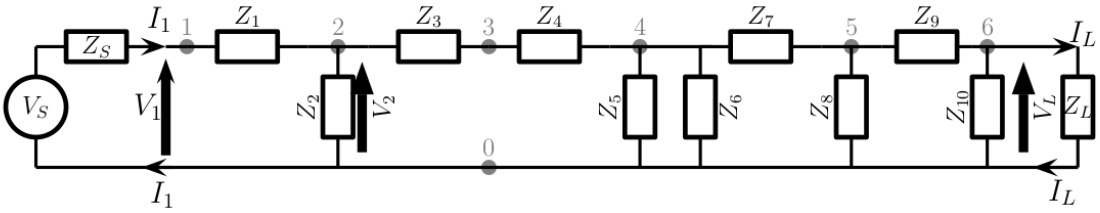


Figure 2.1: A cascade circuit of series and shunt impedances. Node numbers, 1 to 6 in this case, are used to define how components are connected. Node zero is the common (ground?) connection.

Your program must perform the following tasks:

1. Read in the circuit input file using the format specified below.
2. Analyse the circuit to evaluate the requested output variables.
3. Write the output variables to a file in the specified format.

You may extend the work in any way that you wish, so long as it is obvious and does not interfere with completion of the automated tests. For example your program may display a representation of the circuit under analysis. Your program should display a message that explains any such extensions you chose to make.

3 Running the program

The program is to read an input data file, say 'test.net', and output to a results file, say 'test.csv'. These files are to be specified in the command line, so to run 'MyCode.py' on these files the command line should be

```
python MyCode.py test.net test.csv
```

4 Input and Output file formats and syntax

The files that describe the circuit (the input) and the results (the output) are described here. Your program should be as tolerant as possible of format errors in the input files and as precise as possible in the format of the output files. *Note that because an automated tester will be used to assess your output files you must comply **exactly** with the file format described here.*

There are many examples of input and corresponding output files on the Moodle page for this course.

4.1 Input file

The input file contains a description of the circuit that is to be analysed. Lines that start with the hash symbol (#) are comments. The input file will contain three blocks, each of which is described in greater detail below:

CIRCUIT block: This is delimited by <CIRCUIT> and </CIRCUIT>. It defines the components in the circuit and how they are connected to each other.

TERMS block: This is delimited by <TERMS> and </TERMS>. It defines the source and the load.

OUTPUT block: This is delimited by <OUTPUT> and </OUTPUT>. It defines ~~the output filename to use,~~ the variables to be printed out into the file and the units to be used for each variable.

The following example is available on the Moodle site as 'a_Test_Circuit_1.net'.

4.1.1 CIRCUIT block

The start of the CIRCUIT block is indicated by the xml style text <CIRCUIT> and the end by </CIRCUIT>. Each component in the circuit is defined by two node numbers, one for each end of the component, and then its resistance (R), conductance (G=1/R), inductance (L) or capacitance (C).

```
# define a circuit between <CIRCUIT> and </CIRCUIT> delimiters
# Elements have two node numbers and component value is a
# resistance R=1/G (Ohms), a conductance G=1/R (Seimens), an inductance (Henries)
# or a capacitance (Farads).
<CIRCUIT>
n1=1 n2=2 R=8.55
n1=2 n2=0 R=141.9
n1=2 n2=3 R=8.55
n1=3 n2=4 L=1.59e-3
n1=4 n2=0 C=3.18e-9
n1=4 n2=0 L=7.96e-6
n1=4 n2=5 C=6.37e-7
n1=5 n2=0 R=150.5
# components do not have to follow their order in the circuit
n1=6 n2=0 R=150.5
n1=5 n2=6 G=0.02677
</CIRCUIT>
```

Note that components do not have to be specified in the order that they appear in the circuit, the node numbers show how the components are connected.

4.1.2 TERMS block

The TERMS block defines the source and the load attached to the circuit.

The source is implicitly connected between node 0, the common node, and node 1.

The source may be given as a Thevenin source or a Norton source.

The load is connected between the implicit node 0 and the last node specified in the CIRCUIT block.

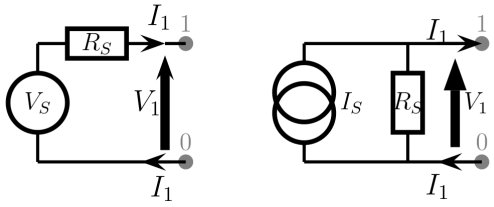


Figure 4.1: Thevenin and Norton sources. Equivalent when $V_S = I_S\, R_S$.

```
# define the terminations between <TERMS> and </TERMS> delimiters
<TERMS>
# A 5V Thevenin voltage source with RS=50 ohms connected
# between node 1 and the implicit common (0) node
VT=5 RS=50
# A 2.5 Amp Norton current source with RS=25 Ohms
#IN=2.5 RS=25
#IN=2.5 GS=0.04
# Load connected between last node (6 in this case) and the implicit common (0)
RL=75
# Frequency range and number of frequencies to evaluate at.
Fstart=10.0 Fend=10e+6 Nfreqs=10
</TERMS>
```

4.1.3 OUTPUT block

The OUTPUT block defines the desired output of the circuit analysis. ~~, including a filename where the output variables will be written.~~ It defines which variables are to be written, in what order, and the units to be written for each variable. The variables to be output can be:

- Vin - Input Voltage
- Iin - Input Current
- Pin - Input Power
- Zin - Input Impedance
- Vout - Output Voltage
- Iout - Output Current
- Pout - Output Power
- Zout - Output Impedance
- Av - Voltage Gain
- Ai - Current Gain

Note: input values are defined as being measured between nodes 0 and 1 (through node 1 for current). Output values are defined as being measured between the node 0 and the load node, which is the last node that is defined in the CIRCUIT block of the input file.

```
# define the outputs between <OUTPUT> and </OUTPUT> delimiters
# Order of parameters defines the order they appear in the columns of the
# output file and their units.
<OUTPUT>
Vin V
Vout V
Iin A
Iout A
Pin W
Zout Ohms
Pout W
Zin Ohms
Av
Ai
</OUTPUT>
```

4.1.4 Output file format

The first line of the output file should be the variable type. The second line of the output file should be the units for the output variables and the third and subsequent lines are the variable values. Apart from the frequency the output values are all complex numbers in either real and imaginary format, or when using decibel outputs in magnitude and phase. The commas following the labels and the data should all align. The units for a linear gains should be L. The following output in file 'a_Test_Circuit_1.csv' results from the example input file 'a_Test_Circuit_1.net'.

```
     Freq,    Re(Vin),    Im(Vin),   Re(Vout),   Im(Vout),    Re(Iin),    Im(Iin),   Re(Iout),   I
       Hz,          V,          V,          V,          V,          A,          A,          A,
1.000e+01,  1.247e+00,  5.032e-03, -4.487e-08,  1.894e-10,  7.506e-02, -1.006e-04, -5.983e-10,  2.
1.111e+06,  3.753e+00,  1.130e-02, -2.138e-03, -9.601e-03,  2.495e-02, -2.260e-04, -2.850e-05, -1.
2.222e+06,  3.753e+00,  5.649e-03, -2.041e-03, -1.054e-03,  2.494e-02, -1.130e-04, -2.721e-05, -1.
3.333e+06,  3.753e+00,  3.764e-03, -9.215e-04, -2.779e-04,  2.494e-02, -7.528e-05, -1.229e-05, -3.
4.444e+06,  3.753e+00,  2.822e-03, -5.175e-04, -1.122e-04,  2.494e-02, -5.645e-05, -6.899e-06, -1.
5.556e+06,  3.753e+00,  2.258e-03, -3.306e-04, -5.624e-05,  2.494e-02, -4.515e-05, -4.407e-06, -7.
6.667e+06,  3.753e+00,  1.881e-03, -2.293e-04, -3.218e-05,  2.494e-02, -3.763e-05, -3.057e-06, -4.
7.778e+06,  3.753e+00,  1.612e-03, -1.683e-04, -2.012e-05,  2.494e-02, -3.225e-05, -2.244e-06, -2.
8.889e+06,  3.753e+00,  1.411e-03, -1.288e-04, -1.342e-05,  2.494e-02, -2.822e-05, -1.717e-06, -1.
1.000e+07,  3.753e+00,  1.254e-03, -1.017e-04, -9.397e-06,  2.494e-02, -2.508e-05, -1.356e-06, -1.
```

5 Additional extension features

The following additional extension features can be added in order to gain more marks.

5.1 Exponent prefixes

Add the exponent prefixes (p, n, u, m, k, M, G) and use them in reading and writing all input and output variables.

Symbol	Prefix	Factor
p	pico	10^{-12}
n	nano	10^{-9}
u	micro (μ)	10^{-6}
m	milli	10^{-3}
k	kilo	10^{3}
M	mega	10^{6}
G	giga	10^{9}

Decibels and phase

Add the ability to express the outputs in decibels. The <OUTPUT> block would be of the form:

```
<OUTPUT>
Vin V
Vout dBV
Iin A
Iout uA
Pin dBW
Pout dBmW
Zin Ohms
Zout kOhms
Av dB
Ai dB
</OUTPUT>
```

The output should be decibels and the argument or phase in radians of the complex value. The separator between the magnitude in dB and the phase should be `/_`, i.e. a slash and an underscore. Example output is:

```
     Freq,    Re(Vin),    Im(Vin),     |Vout|,     /_Vout,    Re(Iin),    Im(Iin),   Re(Iout),   I
       Hz,          V,          V,        dBV,       Rads,          A,          A,         uA,
1.000e+01,  1.247e+00,  5.032e-03, -1.470e+02,  3.137e+00,  7.506e-02, -1.006e-04, -5.983e-04,  2.
4.642e+01,  3.753e+00,  1.130e-02, -4.014e+01, -1.790e+00,  2.495e-02, -2.260e-04, -2.850e+01, -1.
2.154e+02,  3.753e+00,  5.649e-03, -5.278e+01, -2.665e+00,  2.494e-02, -1.130e-04, -2.721e+01, -1.
1.000e+03,  3.753e+00,  3.764e-03, -6.033e+01, -2.849e+00,  2.494e-02, -7.528e-05, -1.229e+01, -3.
4.642e+03,  3.753e+00,  2.822e-03, -6.552e+01, -2.928e+00,  2.494e-02, -5.645e-05, -6.899e+00, -1.
2.154e+04,  3.753e+00,  2.258e-03, -6.949e+01, -2.973e+00,  2.494e-02, -4.515e-05, -4.407e+00, -7.
1.000e+05,  3.753e+00,  1.881e-03, -7.271e+01, -3.002e+00,  2.494e-02, -3.763e-05, -3.057e+00, -4.
4.642e+05,  3.753e+00,  1.612e-03, -7.542e+01, -3.023e+00,  2.494e-02, -3.225e-05, -2.244e+00, -2.
2.154e+06,  3.753e+00,  1.411e-03, -7.776e+01, -3.038e+00,  2.494e-02, -2.822e-05, -1.717e+00, -1.
1.000e+07,  3.753e+00,  1.254e-03, -7.982e+01, -3.049e+00,  2.494e-02, -2.508e-05, -1.356e+00, -1.
```

Logarithmic frequency sweep

Add the ability to calculate at logarithmically spaced frequencies as specified in the <TERMS> block as

```
LFstart=10.0 LFend=10e+6 Nfreqs=10
```

The 10 values of $\log(f)$ are then evenly spaced, but the linear frequencies, f, are to be output. Such output is shown in the last example of output.

Program execution time

Programs will be tested for speed of execution. The execution time is the time taken for the complete Python program to execute from the command line to the program terminating. Best results will be obtained for a program that produces valid output files in the shortest execution time.

6 Workplan

6.1 LOIL and PC Laboratory Sessions

There are LOIL and PC Laboratory sessions on the timetable. In the earlier weeks of the semester these are directed towards stand alone tasks that are designed to help you understand particular parts of the lecture course. These tasks are available to you on the Moodle site. The later sessions in weeks 9, 10 and 11 are left for discussions on particular issues that you may need to raise while you do exploratory work on your analysis program.

6.2 First Laboratory session, week 5

During the first all-day laboratory session you should develop your Design Document, and attempt to create the basic I/O framework for your program. At the end of this session you should have the majority of the Design Document and a program that is reads in the input file and writes out test data correctly. You should include in your program a dummy analysis that produces a dummy solution for output.

6.3 Second Laboratory session, week 9 and 10

In the second laboratory session you should try to implement the analysis program itself and record all the needed test data. You can also work in your own time, as all campus computers will have Visual Studio or Anaconda/Spyder installed.

7 Assessment

There are three assessments for the coursework in this unit, and the Marking Rubric that will be used can be seen on the Moodle site:

Design Document: The design document for your circuit analysis program should be submitted via Moodle by **4pm on March 15th 2024**.
It should not include blocks of Python code, but it should include:

- Analysis of the problem so that your program will analyse any circuit defined in the input file format, the outline code structure you are proposing in terms of functions or functional blocks and the data flow through them. Flowcharts can be a simple tool to describe the structure and flow of the program.
- A description of every function or functional block you plan to implement as a part of your program, and how each one is to be tested.
- A description of every data structure, array and enumerated variable you plan to implement as a part of your analysis program.

Typically the design can be expressed in 3 or 4 pages. 20% of the coursework mark (10% of the Unit mark) will be awarded for this.

Circuit analysis program - Code: Your circuit analysis Python program that is executable within the automated test mark. This is due in via Moodle by **4pm on May 3rd 2024**.

You will need to submit your Python files which are executable in a single directory or folder, as needed by the auto tester. Do NOT submit your whole project directory (marks will be deducted if you do this).

Your submitted executable will run through a set of automated tests, and the results will form part of your marks. Make sure that your program runs as expected on the computers in the undergraduate labs. Source code will be inspected for good readability, commenting and coding standards according to the EE20084 coding standards document (available on Moodle).

40% of the coursework marks (20% of the Unit marks) will be awarded for this.

Circuit analysis program - Final Report: This should be submitted via Moodle by **4pm on May 3rd 2024**. The report should cover the description of how the finished code operates, documentation of the code, the testing strategy and test records used on components of the code, and the testing strategy and test records used on the complete code.

40% of the coursework marks (20% of the Unit marks) will be awarded for this.

8 Opportunities for feedback

Various opportunities for feedback exist during the unit. There are tutorial sessions and laboratory sessions where you can gain feedback from lecturers and demonstrators. In addition the will be feedback on the design document that you submit via Moodle.

Labs/tutorials: At the end of the lab/tutorial session, consider whether you have successfully completed the tasks for that session or not:

Struggling: “The code didn’t work and I don’t know why”

Basic: “The code compiled and sort of worked”.

Advanced: “ I was able to spot out algorithmic and syntax errors before compilation. I tested the code in detail and it always works the way that it should”.

Software: We will provide feedback on software submitted.

A ABCD or Chain Matrix Analysis

The ABCD matrix relates voltages and currents at the input and output connection ports of a circuit. The ABCD matrix is also known as the chain matrix or voltage transmission matrix. The ABCD matrix formulation is particularly useful for the cascade connection of components, which is a very common and natural way for components to be connected.

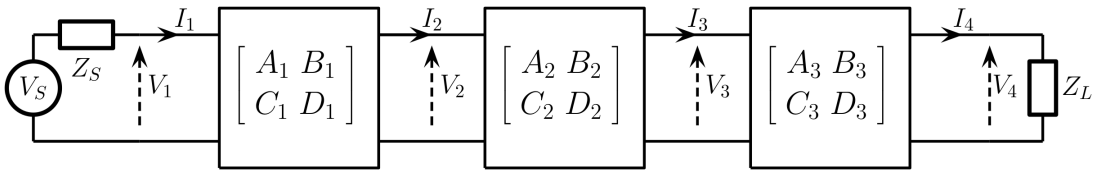


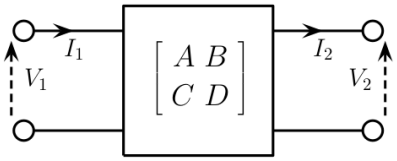
Figure A.1: Cascade of three 2-port circuits.

Figure A.1 shows three components connected in cascade. The ABCD matrix of the whole circuit is simply found by multiplying the ABCD matrices of the elements in the order that they appear in the circuit:

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} = \begin{bmatrix} A_1 & B_1 \\ C_1 & D_1 \end{bmatrix} \begin{bmatrix} A_2 & B_2 \\ C_2 & D_2 \end{bmatrix} \begin{bmatrix} A_3 & B_3 \\ C_3 & D_3 \end{bmatrix} \tag{1}$$

A.1 Definition of the ABCD matrix

Consider a linear two-port circuit with voltages V_1 and V_2 and currents I_1 and I_2 at the two ports.



$$\begin{aligned} V_1 &= AV_2 + BI_2 \\ I_1 &= CV_2 + DI_2 \end{aligned}$$

The equation set is cast into the ABCD matrix form as

$$\begin{bmatrix} V_1 \\ I_1 \end{bmatrix} = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} V_2 \\ I_2 \end{bmatrix} = [T] \begin{bmatrix} V_2 \\ I_2 \end{bmatrix} \tag{2}$$

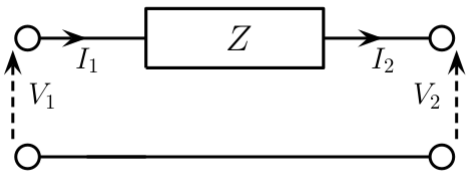
This can be simply inverted to find:

$$\begin{bmatrix} V_2 \\ I_2 \end{bmatrix} = [T]^{-1} \begin{bmatrix} V_1 \\ I_1 \end{bmatrix} = \frac{1}{AD - BC} \begin{bmatrix} D & -B \\ -C & A \end{bmatrix} \begin{bmatrix} V_1 \\ I_1 \end{bmatrix} \tag{3}$$

The four parameters of the matrix can be determined by examination of the circuit with open and short circuit load conditions.

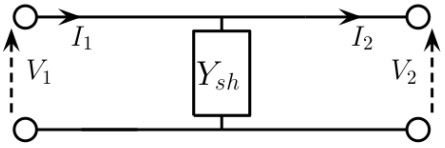
A.1.1 Examples of ABCD Matrices

Series Impedance and Shunt Admittance



In the case of a series impedance, Z, the ABCD matrix is:

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} = \begin{bmatrix} 1 & Z \\ 0 & 1 \end{bmatrix} \tag{4}$$



In the case where the shunt element is an admittance $Y_{sh} = 1/Z_{sh}$:

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ Y_{sh} & 1 \end{bmatrix} \tag{5}$$

A.1.2 Cascade of several circuit elements

The cascade connection is a *very common* way of connecting components. Consider the case where there are several two-port circuits in cascade.

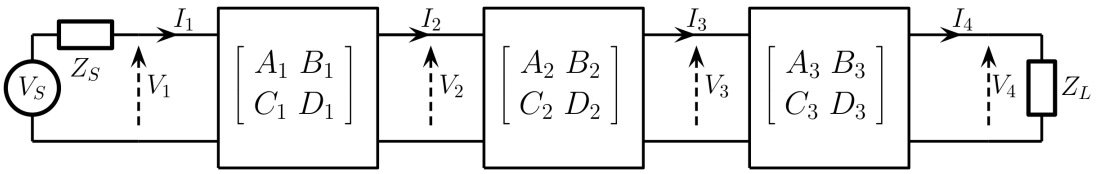


Figure A.2: Cascade of three 2-port circuits.

The behaviour of the overall circuit is determined by the ABCD matrices of the elements of the cascade:

$$\begin{aligned}
\begin{bmatrix} V_1 \\ I_1 \end{bmatrix} &= \begin{bmatrix} A_1 & B_1 \\ C_1 & D_1 \end{bmatrix} \begin{bmatrix} V_2 \\ I_2 \end{bmatrix} &= [T_1] \begin{bmatrix} V_2 \\ I_2 \end{bmatrix} \\
\begin{bmatrix} V_2 \\ I_2 \end{bmatrix} &= \begin{bmatrix} A_2 & B_2 \\ C_2 & D_2 \end{bmatrix} \begin{bmatrix} V_3 \\ I_3 \end{bmatrix} &= [T_2] \begin{bmatrix} V_3 \\ I_3 \end{bmatrix} \\
\begin{bmatrix} V_3 \\ I_3 \end{bmatrix} &= \begin{bmatrix} A_3 & B_3 \\ C_3 & D_3 \end{bmatrix} \begin{bmatrix} V_4 \\ I_4 \end{bmatrix} &= [T_3] \begin{bmatrix} V_4 \\ I_4 \end{bmatrix}
\end{aligned} \tag{6}$$

Relating the voltages and currents from left to right:

$$\begin{aligned}\begin{bmatrix} V_1 \\ I_1 \end{bmatrix} &= [T_1]\begin{bmatrix} V_2 \\ I_2 \end{bmatrix} \\ &= [T_1]\,[T_2]\begin{bmatrix} V_3 \\ I_3 \end{bmatrix} \\ &= [T_1]\,[T_2]\,[T_3]\begin{bmatrix} V_4 \\ I_4 \end{bmatrix} = \begin{bmatrix} A_C & B_C \\ C_C & D_C \end{bmatrix}\begin{bmatrix} V_4 \\ I_4 \end{bmatrix} \end{aligned} \tag{7}$$

So the ABCD matrix of the cascade network is determined by multiplying together the ABCD matrices of the elements in the order they appear in the cascade circuit. This is why the matrix is sometimes referred to as the chain or voltage transmission matrix.

Observation

For a general cascade circuit the overall ABCD matrix is simply the multiple of the ABCD matrices of the individual components of the cascade, in the order that they occur in the cascade:

$$[T_C] = [T_1]\,[T_2]\,[T_3]\,\ldots[T_M] \tag{8}$$

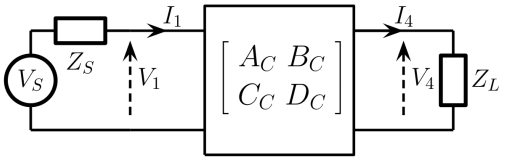


Figure A.3: Loaded 2-port circuit described by ABCD matrix.

From the ABCD matrix of the complete cascade network various features of the complete network are easily found, such as input impedance, voltage gain, current gain, power gain.....

A.1.3 Input and output impedance

When a load, Z_L, is connected at the end of the cascade $\frac{V_4}{I_4} = Z_L$ and the input impedance as seen by the source, Z_{in} is

$$Z_{in} = \frac{V_1}{I_1} = \frac{A_C V_4 + B_C I_4}{C_C V_4 + D_C I_4} = \frac{A_C \frac{V_4}{I_4} + B_C}{C_C \frac{V_4}{I_4} + D_C} = \frac{A_C Z_L + B_C}{C_C Z_L + D_C} \tag{9}$$

When a source, Z_S, is connected the output impedance as seen at the load end, Z_{out} is

$$Z_{out} = \frac{D_C Z_S + B_C}{C_C Z_S + A_C} \tag{10}$$

A.1.4 Voltage, current and power gain

As $\frac{V_4}{I_4} = Z_L = \frac{1}{Y_L}$ using the base definition of the ABCD matrix:

$$V_1 = A_C V_4 + B_C I_4 = A_C V_4 + B_C V_4 Y_L \tag{11}$$

the voltage gain is:

$$A_V \;=\; \frac{V_4}{V_1} \;=\; \frac{1}{A_C \;+\; B_C Y_L} \;=\; \frac{Z_L}{A_C\, Z_L \;+\; B_C} \tag{12}$$

Likewise

$$I_1 \;=\; C_C V_4 \;+\; D_C I_4 \;=\; C_C I_4 Z_L \;+\; D_C I_4 \tag{13}$$

and the current gain is:

$$A_I \;=\; \frac{I_4}{I_1} \;=\; \frac{1}{C_C Z_L \;+\; D_C} \tag{14}$$

The power gain is:

$$A_P \;= A_V A_I^* \tag{15}$$

where the complex conjugate of the current gain is needed.

A.1.5 Transmittance

The transmittance from source to load is:

$$T \;=\; \frac{2}{A_C Z_L \;+\; B_C \;+\; C_C Z_L Z_S \;+\; D_C Z_S} \tag{16}$$

A.1.6 Familiarisation Exercises

It is often easier to develop functions or programs to implement an algorithm if you can perform the operations yourself 'by-hand'. Examples that you are convinced are correct often (always?) form test cases for you to prove your software programs against.

These familiarisation exercises illustrate the basic steps that are needed to perform an ABCD matrix circuit analysis.

Example 1

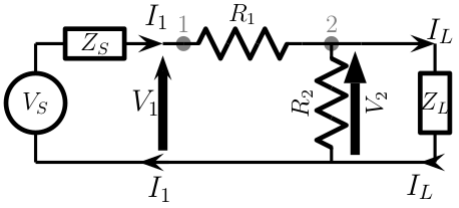


Figure A.4: A two element cascade circuit of series and shunt resistances.

Question 1

In the case where $Z_S = R_1 = R_2 = Z_L = 50\Omega$ calculate the ABCD matrix, Z_{in} and A_v.

Question 2

In the case where $Z_S = Z_L = 50\Omega$ and $R_1 = R_2 = 150\Omega$ calculate the ABCD matrix, Z_{in} and A_v.

Question 3

In the case where $Z_S = Z_L = 50\Omega$, $R_1 = R_2 = 150\Omega$ and $V_s = 5\ Volts$ calculate $V_1,\ I_1,\ V_2$ and I_L.

Example 2

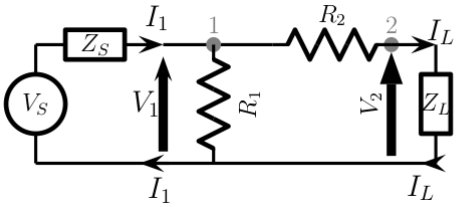


Figure A.5: A two element cascade circuit of shunt and series resistances.

Question 4

In the case where $Z_S = R_1 = R_2 = Z_L = 50\Omega$ calculate the ABCD matrix, Z_{in} and A_v.

Question 5

In the case where $Z_S = Z_L = 50\Omega$ and $R_1 = R_2 = 150\Omega$ calculate the ABCD matrix, Z_{in} and A_v.

Question 6

In the case where $Z_S = Z_L = 50\Omega$, $R_1 = R_2 = 150\Omega$ and $V_s = 5\ Volts$ calculate $V_1,\ I_1,\ V_2$ and I_L.