Chapter 1

Introduction

1.1 History of Breadboards

The breadboard has been a staple substrate for electronic construction over the last century. At the dawn of a growing interest in amateur radio, resourceful tinkerers used planks of wood to secure and ruggedize their electrical handiwork. Conductive nodes, such as nails or copper rails, were driven into the non-conductive boards, providing anchors and contact points that were electrically isolated from the rest of the circuit. Components were soldered or wire-wrapped to the nodes, and sometimes secured by non-energized nails or screws. This construction technique provided a lot of artistic freedom in circuit construction, but was time consuming and required relatively heavy hand tools such as a hammer or drill.

1.2 Modern Breadboards

The solderless breadboard is the canonical tool given to students taking introductory courses in the field. Rather than driving nodes into arbitrary locations, component leads are inserted into contact points arranged on a grid that allow rapid semi-rigid construction of circuits with no other tools. The layout of a solderless breadboard is designed to be compatible with a plethora of powerful integrated circuits, enabling complex electronic designs. Modern solderless breadboards have come a long way from their namesake wooden ancestors, but there is still room for improvement.

The intent of breadboarding is to physically realize a circuit. Often, this involves designing or using a reference schematic to guide construction, but circuit improvisation is

not uncommon. A meticulous breadboarder can successfully realize a circuit without error by correctly placing components and jumper wires - taking care not to introduce undesired 'parasitic' components. However, for the uninitiated it is difficult to justify the additional time and care required to plan and build. Inserting components and jumper wires into contact points is straightforward, but poor contacts, broken wires (inside insulation), and misinserting leads can plague designers for hours on end and potentially destroy components. The shortcomings of solderless breadboards lie entirely within the art of breadboarding. Breadboarding is a skill that is learned over time, but small errors can lead to excessive frustration and turn students off to the field.

To satisfy the requirements of the Masters in Engineering program, I propose a solution to some of the issues with the solderless breadboard.

A confident linkage between the the electrical and mechanical domains is required to construct a circuit. On larger scales, the mechanical structures (eg. contacts, wires, components) that create electrical circuits are visible in plain sight. It is simple and reliable, then, to determine the circuit representation of a mechanical structure that has no hidden connections. Breadboards often obscure connections due to their construction. The regular nature of a breadboard, a grid of contact points arranged in rails, makes it difficult to keep track of which rail is connected to what circuit node. When combined with the opaque nature of most components and the poor reliability of breadboard contacts, visually verifying complex circuits on a breadboard becomes infeasible. Many of the inherent problems with breadboards stem from this open-loop nature of breadboarding, where visual cues are not enough to determine the electrical circuit from the mechanical structure. A symptom of open-loop construction techniques is a mismatch between the mental model and the physical realization of the system at hand. I seek to close this loop by designing and implementing a circuit-sensing breadboard.

1.3 Proposed Solution

A hardware test-bench was constructed to interface with 8 rails on a breadboard. The test-bench is composed of a pcb-mounted breadboard, an ADC and DAC multiplexing board, a microcontroller development board, firmware to control sampling and stimulating the breadboard, and software to stream the sampling data to a computer for display. Since software

is faster to prototype with, the theoretical circuit topology work was first implemented in software. A circuit simulator test-bench was written to provide the circuit-sensing algorithm in development with data. This test-bench allows the circuit-sensing algorithm to probe, stimulate, and ground every node of a randomly generated circuit, as if the circuit were built on a breadboard attached to the hardware test bench mentioned above. The software test-bench also had the ability to print out a circuit network solution to a schematic display.

1.4 Implementation to Date

Made the software test bench, an 8-rail hardware test bench, and got the network sensing algorithm working for resistors with resistance between 100 and 5K Ω.

Chapter 2

Theory

The design of a Network Sensing Algorithm requires an understanding of the networks to be analyzed. In this thesis, the networks of interest are composed of three circuit elements: resistors, capacitors, and inductors. The constitutive relations for these electrical circuit elements relate the voltage across an element to the current passing through it as a function of time. In this application, it is useful to reinterpret the constitutive relations as a function of frequency by taking the Laplace Transform.

2.1 RLC Elements

Constitutive relations: $v = iR, v = L\frac{di}{dt}$, and $i = C\frac{dv}{dt}$

2.2 Frequency Domain Perspective

Laplace transform of:

$v = iR, v = L\frac{di}{dt}$, and $i = C\frac{dv}{dt}$

$V = IR, V = LsI, I = CsV$

Impedance:

$Z_R = R, Z_L = Ls, Z_C = \frac{1}{Cs}$

2.3 Network Analysis

KVL KCL

Chapter 3

Network Sensing Algorithm

Given access to the set of nodes in an electrical network, the objective of a network sensing algorithm is to determine the set of branches and the elements that compose them. In order to illustrate how a network sensing algorithm operates, it is necessary to begin with a simple example and then build in complexity.

3.1 Grounding Clause

Blah Blah

3.2 Two Node Network

Take the example of a network with two nodes below:

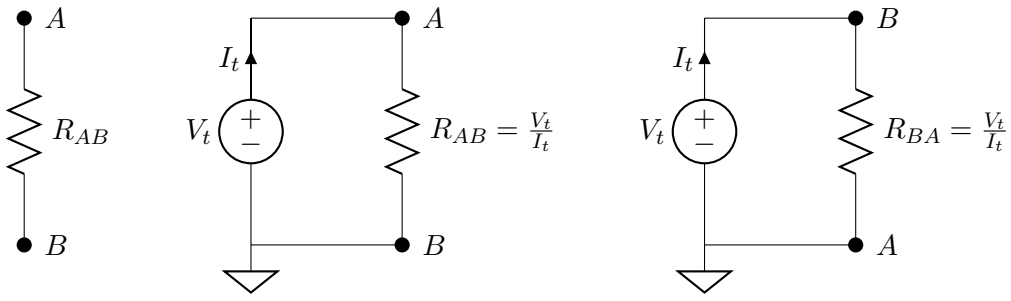


Figure 3-1: Finding R_{AB} in a two node network

In the case of a resistive network with two nodes, there is only one possible branch in the network and thus one possible element to characterize. From elementary circuit theory, the

resistance between two nodes equal to the voltage across the nodes divided by the current through the nodes when power is applied. Here, the resistance R_{AB} is found by placing a test voltage V_t across the nodes and measuring the resulting current, I_t, then taking the ratio $\frac{V_t}{I_t}$. This measurement is called the driving point impedance [1]. If there are no circuit elements between the two nodes, the driving point impedance test will find zero current in the test voltage source, resulting in an infinite resistance between two nodes. An infinite resistance between two nodes in a circuit indicates that there are no elements in that branch of the network. Thus, the network can subsequently be simplified by removing that branch from the network.

3.3 Three Node Network

A resistive network with two nodes is simple, but provides an introduction to the methodology used in analyzing larger networks. In the case of a resistive network with three nodes, it is insufficient to utilize driving point impedance measurements alone, because each node has more than one path to any other node.

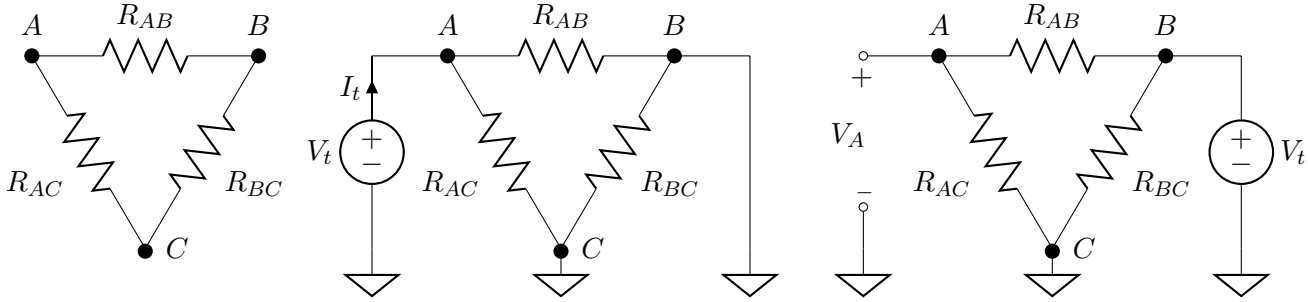


Figure 3-2: Finding R_{AB} in a three node network

Consider a resistive network with three nodes: A, B, and C. In order to determine the resistance in branch AB, the driving point impedance at node A is measured with nodes B and C grounded. This provides the resistance of the parallel combination of the branches with an endpoint at node A,
$R_{A_{||}} = R_{AB}||R_{AC}$. [2] Next, a test voltage source is applied to node B, node C is grounded,

[1] To measure a driving point impedance, a test voltage is applied between the node of interest and ground, and the resulting current in the voltage source is measured. The driving point impedance is then computed by dividing the test voltage by the resulting current.

[2] $R_1||R_2 = \frac{R_1 R_2}{R_1 + R_2}$

and the voltage at node A, V_A, is observed. $V_A = V_t \frac{R_{AC}}{R_{AB} + R_{AC}} = V_t \frac{R_{AB}||R_{AC}}{R_{AB}} = V_t \frac{R_{A_{||}}}{R_{AB}}$
The branch resistance of interest, R_{AB} is calculated using the known quantities V_t, $V_{A||}$, and V_A. $R_{AB} = V_t \frac{R_{A||}}{V_A}$
This procedure is repeated for the remaining branches to determine the entire network.

3.4 N Node Network

A resistive network with any number of nodes can be reduced to a resistive network with three nodes by grounding the nodes that are not of interest. The resulting network does not modify the branch of interest, but connects the remaining branches attached to the nodes of interest in parallel. This collapses the network into a three node network or three branch equivalent circuit. Consider a resistive network with five nodes: A-E. To determine

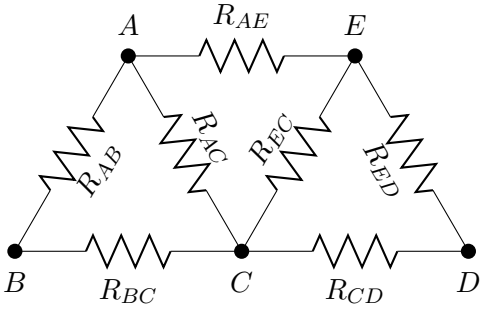


Figure 3-3: Five node network

the resistance between nodes A and E, the network is reduced to a three-node network by connecting all nodes except nodes A and E to ground. The three resistances of the branches that remain are R_{AE}, $R_{AB}||R_{AC}$, and $R_{EC}||R_{ED}$. The reduced network is then solved using the three node network method, and this procedure is repeated for all branches in the network.

3.5 Element Identification

Networks composed of elements with complex impedance can be analyzed with the same algorithm. By replacing the test DC voltage sources with AC voltage sources, the imaginary reactance of capacitors and inductors can be measured in addition to the real resistance of resistors.

3.5.1 From Resistance to Impedance

Chapter 2 described the use of the Laplace Transform to characterize the behavior of circuit elements in the frequency domain. Here, frequency-domain complex impedance is useful for identifying circuit elements based on the change in branch impedance over inputs of various frequencies.

MAKE NOTE ABOUT USING THE AMPLITUDE OF THE MEASURED SIGNALS

3.5.2 Parallel RLC Branches

To determine if there are multiple element types [in parallel] between two nodes, we can select a few frequencies to scan through and analyze the resulting change in impedance.

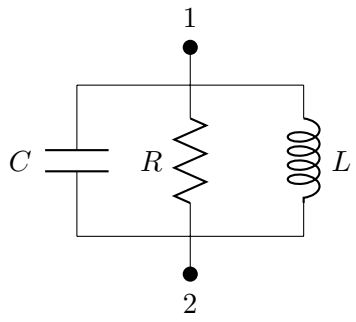


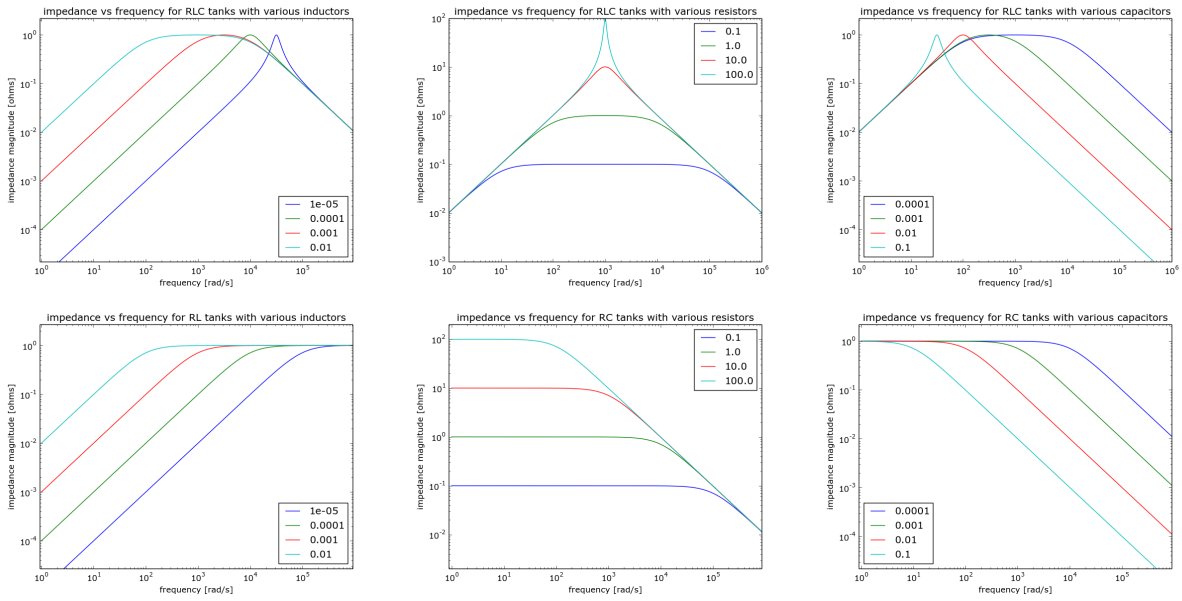
Figure 3-4: Example RLC Tank Circuit

The impedance of a parallel RLC 'tank' circuit can be characterized and analyzed over all frequencies.

$$Z_C = \frac{1}{j\omega C} \qquad Z_R = R \qquad Z_L = j\omega L \tag{3.1}$$

$$Z_{RLC} = Z_C||Z_R||Z_L = \frac{1}{j\omega C + \frac{1}{R} + \frac{1}{j\omega L}} = \frac{j\omega RL}{-\omega^2 RLC + j\omega L + R} \tag{3.2}$$

Z_{RLC} has a zero at $\frac{1}{RL}$ and two poles at $\frac{-L^{+}_{-}\sqrt{L^2+4R^2LC}}{-2RLC}$



If we look at the plots above, we can see that varying the resistance changes the maximum impedance over all frequencies, varying the inductance drops the asymptotic impedance at low frequencies, and varying the capacitance drops the asymptotic impedance at high frequencies. We can imagine there are three regimes on the impedance vs frequency plot that divide the operation of an RLC tank into three elements:
When the slope is +1, the tank behaves like an inductor $L = |Z|/(j\omega)$. When the slope is -1, the tank behaves like a capacitor $C = j\omega|Z|$. When the slope is 0, the tank behaves like a resistor $R = |Z|$.

3.5.3 Finite Difference Stencil

To reliably determine the regions of effective resistance, inductance, and capacitance, the slope of the $log|Z_{nm}|$ vs. $log(\omega)$ is computed via the Midpoint Method. The Midpoint Finite Difference Stencil takes the two points on either side of the point of interest and computes the slope between them. Z'[n]=$\frac{1}{2}$(Z[n+1]-Z[n-1]).

3.5.4 Component Value Calculation

Once the regions of the impedance plot are identified and assigned to component, the component value is calculated for each sample taken on the impedance plot:
$R = Z[\omega]$

$L = \frac{Z[\omega]}{\omega}$

$C = \frac{1}{\omega Z[\omega]}$

The resulting resistances, inductances, and capacitances are averaged to counter noisy data.

3.6 Reconstructing the Network

With a record of all of the elements and their connections in the network, the network is reconstructed.

Chapter 4

Simulation

A simulation was built to test the Network Sensing Algorithm before embarking on hardware design. In addition to the rapid prototyping cycle, the implementation of NSA and automated schematic drawing were directly used later on. The simulation generates random networks of resistors, inductors, and capacitors, and proceeds to analyze and reconstruct the network using NSA.

4.1 NgSpice and Netlists

The open source software package NgSpice was used to simulate the networks and test circuits applied to the network. NGspice operates on text files called netlists, where each component in the network is specified on a single line. An example netlist is shown below.

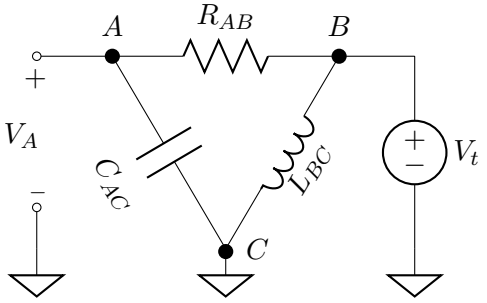


Figure 4-1: Five node network

```
fiveNodeNetlist
Vt 0 B 1
```

```
Rab A B 10k
Cac A C 1e-6
Lbc B C 1e-3
Vcg 0 C 0
.control
op
print(v(A))
.endc
.end
```

The first letter of each element line designates the element to simulate:
V start stop value→ Voltage Source between start and stop nodes, value Volts [V]
R start stop value→ Resistor between start and stop nodes, value Ohms [Ω]
L start stop value→ Inductor between start and stop nodes , value Henries [H]
C start stop value→ Capacitor between start and stop nodes, value Farads [F]

Node 0 is always designated as ground, and all simulations require a ground node.

The control commands are as follows:
op → Operating Point Simulation print() → Print the relevant data passed as an argument
v(N) → The voltage at node N i(E) → The current through element E

4.2 Methods

The NSA simulator was written in python and uses the methods below.

4.2.1 Generate Random Netlist

`writeRandomNet(netlist,num,elements):`
Generates `num`-node random graph with no self-linking nodes (symmetric matrix with zeros on the diagonal) for each `[elements]` type (R,L,C). Subsequently assigns random values between two realistic limits for each element and writes the network to netlist `netlist`. 1-1k ohms, 10nF-10uF, 100uH-100mH.

When writing the capacitive and inductive elements, care must be taken to prevent a DC operating point simulation from failing. The infinite resistance across a capacitor and

zero resistance across an inductor are responsible for DC operating point simulation failure, and can be fixed by including a small resistor in series with inductors and a large resistor in parallel with capacitors.

`L0 1 2 1mH`	→	`L0 1 tl0 1mH` `Rl0 tl0 2 1e-5`	`C0 1 2 1e-6F`	→	`C0 1 2 1e-6F` `Rc0 1 2 1e8`

4.2.2 Inserting Voltage Sources, Grounds and Probes

`insertProbe2(target,nodes,groundNodes,probes,source='DC'):`

Inserts a 1V voltage source from ground to each node in `[nodes]`, grounds each node in `[groundNodes]`, and adds a voltage print statement for each node in `[probes]`. By default, the voltage sources are written as DC sources, but if 'AC' is passed into the last argument the sources are written as AC sources and the AC control statement is added.

`AC dec 1 1 100000`

Which runs a small-signal AC simulation and returns the amplitudes of the resulting voltage and current waveforms, one sample point per decade from 1Hz to 100kHz.

4.2.3 Run Simulation

`def runSim(target,results,source='DC'):`

Makes a system call to NgSpice in batch mode with netlist `target` and outputs the result to text file `results`. The last argument indicates how to parse the resulting data, as DC data is returned in the following format:

```
No.  of Data Rows :  1
i(v) = -1.18295e-01
v(5) = 2.532846e-07
```

and AC data is returned in this format:

```
No.  of Data Rows :  6
mynetlist
AC Analysis Sun Aug 30 18:35:07 2015
--------------------------------------------------------------------------------
Index frequency i(v)
--------------------------------------------------------------------------------
0 1.000000e+00 -1.72598e+00, 3.978810e+02
```

```
1 1.000000e+01 -1.58689e-01, 3.978827e+01
2 1.000000e+02 -1.43015e-01, 3.974287e+00
3 1.000000e+03 -1.42859e-01, 3.520201e-01
4 1.000000e+04 -1.42857e-01, -4.18884e-01
5 1.000000e+05 -1.42857e-01, -4.58275e+00

mynetlist
AC Analysis Sun Aug 30 18:35:07 2015
--------------------------------------------------------------------------------
Index frequency v(3)
--------------------------------------------------------------------------------
0 1.000000e+00 1.000000e+00, 0.000000e+00
1 1.000000e+01 1.000000e+00, 0.000000e+00
2 1.000000e+02 1.000000e+00, 0.000000e+00
3 1.000000e+03 1.000000e+00, 0.000000e+00
4 1.000000e+04 1.000000e+00, 0.000000e+00
5 1.000000e+05 1.000000e+00, 0.000000e+00
```

4.2.4 Print Matrix

`def printMatrix(m):` Prints matrix `m` in nice command-line output.

4.3 Executing NSA

Wouldn't we all?

4.3.1 Calculate $Z_{||}(f)$

4.3.2 Calculate $V_n(f)$

4.3.3 Calculate $Z_{nm}(f)$

4.3.4 Finite Difference

4.3.5 Element Identification

4.3.6 Network Reconstruction

4.4 Output to JSON

Blah Blah

4.5 D3?

???? maybe