EngSci Year 3 Fall 2022 Notes

Brian Chen

Division of Engineering Science
University of Toronto
https://chenbrian.ca

brianchen.chen@mail.utoronto.ca

Contents

I ECE349: Introduction to Energy Systems 1

1 Admin stuff 1
1.1 Lecture 1 1
1.1.1 Mark breakdown 1

2 AC Steady State Analysis 1
2.1 Lecture 2 1
2.1.1 TODO 1
2.2 Lecture 3 3

3 AC Power 4
3.1 Lecture 4 5
3.1.1 Root Mean Squared (RMS) Values 6
3.2 Lecture 5: Multi-Phase AC 7
3.3 Lecture 6: Y and Δ connections 10

II ECE352: Computer Organization 11

4 Admin stuff 11
4.1 Lecture 1 11
4.1.1 Mark breakdown 11

5 Preliminary 12
5.1 Lecture 2: Using binary quantities to represent other things 12
5.1.1 Floating Point Numbers 12

6 NIOS II 13
6.1 Lecture 3: Behavioural Model of Memory 14
6.1.1 Memory 14
6.1.2 Physical Interface 15
6.2 Lecture 4: NIOS II Programming Model 16
6.2.1 Adding Two Numbers 17
6.2.2 Adding two numbers using memory 19
6.3 Lecture 5: Simple Control Flow 21

III ECE355: Signal Analysis and Communication 22

7 Admin and Preliminary **22**
7.1 Lecture 1 22
7.1.1 Mark Breakdown 23

8 Transformations **23**
8.1 Lecture 2 23
8.2 Lecture 3 24
8.2.1 General Continuous Complex Exponential Signals 24
8.3 Lecture 4: Step and Impulse Functions 25

IV ECE360: Electronics **27**

9 Admin and Preliminary **27**
9.1 Lecture 1 27
9.1.1 Mark Breakdown 27
9.1.2 Diodes 27

10 Diodes **28**
10.1 Lecture 2 29
10.2 Lecture 3 31

11 Lecture 4: Forward conducting diodes **32**
11.0.1 Small-Signal Model 34

V ECE358: Foundations of Computing **35**

12 Admin and Preliminary **35**
12.1 Lecture 1 35
12.1.1 Mark Breakdown 36

13 Complexities **36**
13.1 Lecture 2 36
13.2 Lecture 3: Logs & Sums 38
13.2.1 Functional Iteration 38
13.3 Lecture 4: Induction & Contradiction 40
13.3.1 Induction 40
13.3.2 Contradiction 41

VI MAT389: Complex Analysis **41**

14 Complex Numbers **42**
14.1 Lecture 1 42
14.2 Lecture 2 44
14.2.1 Functions on complex planes 45
14.2.2 Exponential Functions 45

VII ECE444: Software Engineering **48**

15 Preliminary **48**
15.1 Lecture 1, 2 48

16 Project Management **48**
16.1 Lecture 3 48
16.1.1 Agile 50
16.2 I dropped this course 50

VIII ESC301: Seminar **50**

17 Preliminary **50**
17.1 Seminar 1 50

PART

I

ECE349: Introduction to Energy Systems

Taught by Prof. P. Lehn

Section 1

Admin stuff

Subsection 1.1

Lecture 1

First lecture was logistical info + a speil about how power systems are one of the great modern wonders. Course will cover sinusoidal AC power systems (1, 3 phase), power systems (dc-dc, dc-ac conversion), and magnetic systems (transformers, actuators, and synchronous machines)

1.1.1 Mark breakdown

- 50 % Final
- 25 % Midterm
- 5 % Quiz
- 15 % Labs
- 5 % Assignments

Section 2

AC Steady State Analysis

Subsection 2.1

Lecture 2

2.1.1 TODO

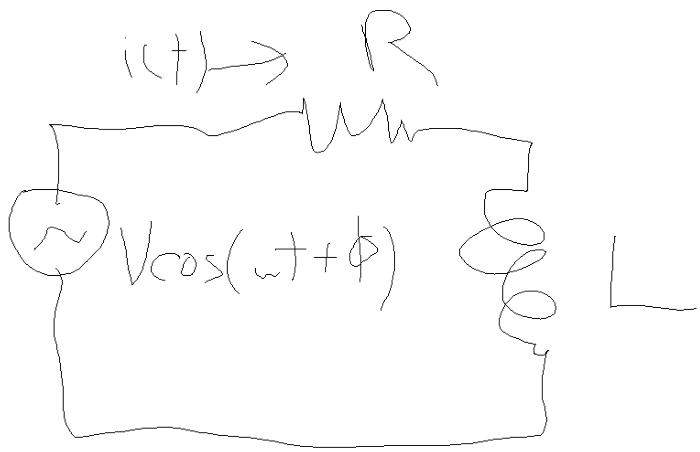
- Review Thomas 669-600

Recall, for a general phasor $\hat{P}$

- $\frac{d\hat{P}}{dt} = jw\hat{P}$
- $\int \hat{P} = \frac{1}{jw}\hat{P}$

What we have learnt prior for differential equations enables us to arrive at analytical solutions to linear stable AC systems with phasors. A homogeneous and particular solution will be produced. If there's a stable homogeneous solution, $\rightarrow 0$ as $t \rightarrow \infty$. The full solution would be the addition of the two via super position.

We generally use this approach to solve circuits since it's an efficient way to solve circuits and make them into essentially DC circuits.



$$Ri + L\frac{di}{dt} = V\cos(\omega t + \phi) \tag{2.1}$$

But this is a pain to solve. It can be made simpler by applying phasors

$$V\cos(\omega t + \phi) = Re\{Ve^{j(\omega t+\phi)}\} \tag{2.2}$$

Take the real part of $\hat{I}$:

$$R\hat{I} + L\frac{d\hat{I}}{dt} = Ve^{j(\omega t+\phi)} \tag{2.3}$$

And therefore by inspection the solution is of format $\hat{I}e^{j\omega t}$, where $\hat{I}$ is a phasor. Noting that $\hat{I}$ contains only amplitude and phase,

$$\begin{gathered} R\hat{I}e^{j\omega t} + L\frac{d}{dt}(\hat{I}e^{j\omega t}) = Ve^{j\omega+\phi} \\ R\hat{I} + L\hat{I}j\omega = Ve^{j\phi} \end{gathered} \tag{2.4}$$

And now reconstructing:

$$\begin{aligned} \hat{I} &= \frac{V}{\sqrt{R^2(\omega L)^2}}e^{j(\omega t+\phi-\tan^{-1}\frac{wL}{R})} \\ i(t) &= Re\left\{\hat{I}\right\} \end{aligned} \tag{2.5}$$

And therefore

$$\hat{I} = \frac{V}{\sqrt{R^2(\omega L)^2}}\cos(\omega t + \phi - \tan^{-1}(\frac{wL}{R})) \tag{2.6}$$

The steps to solving a phasor problem are:

Notation: $Xe^{j\phi} \leftrightarrow X\lfloor\phi$

- Define phasor: $V\cos(\omega t + \phi) \leftrightarrow V\lfloor\phi$
- Map L, C into phasor domain; find impedences
  - $v = L\frac{di}{dt} \leftrightarrow \hat{V} = j\omega L\hat{I}$

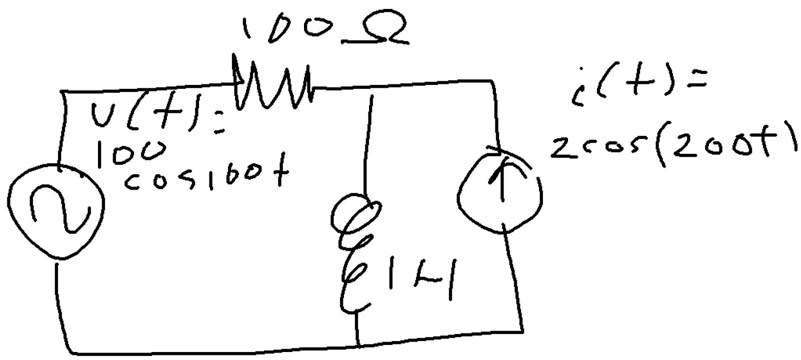
  - $i = C\frac{dv}{dt} \leftrightarrow \hat{V} = \frac{1}{j\omega C}\hat{I}$

- Do mesh analysis to find $\hat{I}$; $\hat{I} = \frac{\hat{V}}{\sum \text{impedances}}$
- Reconstruct $i(t)$ from $\hat{I}$

Subsection 2.2

Lecture 3

Phasors allow us to solve circuits with multiple sources of differing frequencies.



To find the current $i(t)$ over the inductor we can find it's response due to the voltage and current sources and then apply superposition.

- $I_1 = \frac{100\lfloor 0}{1oo+j100} = 0.707\lfloor -45^o \rightarrow i_1(t) = 0.707cos(100t - 45^o)$
- $I_2 = \frac{100}{100+j200}2\lfloor 0 = 0.894\lfloor -65^o \rightarrow i_2(t) = 0.894cos(200t - 63^o)$
- $i(t) = i_1(t) + i_2(t) = 0.707cos(100t - 45^o) + 0.894cos(200t - 63^o)$

Non-sinusoidal stimulus may be solved by decomposing the signal with Fourier transforms. For example, square waves:

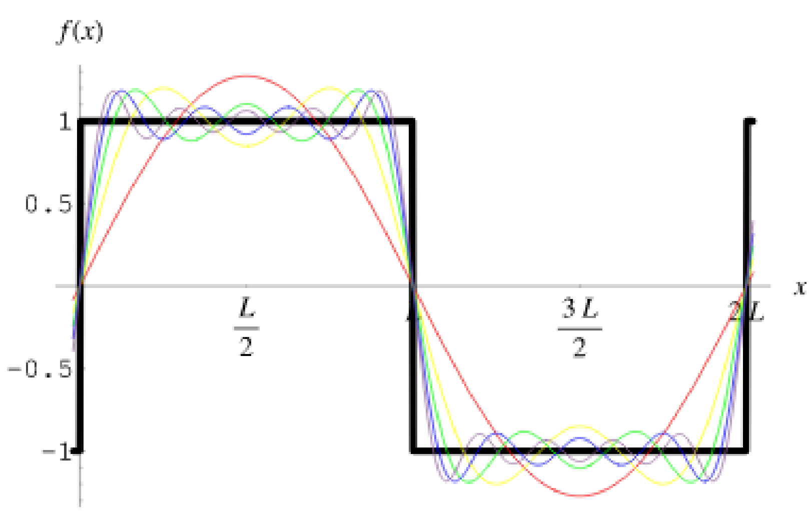


Figure 1. square waves with Fourier series superimposed

The general form of a Fourier transform is given as:

$$v_{equiv}(t) = a_o + \sum_{n=1}^{\infty} a_k cos(nw_o t) + b_k cos(nw_o t) \tag{2.7}$$

Where:

$$\begin{aligned} a_o &= \frac{1}{T}\int_0^T v(t)dt \\ a_k &= \frac{2}{T}\int_0^T v(t)cos(nw_o t)dt \\ b_k &= \frac{2}{T}\int_0^T v(t)sin(nw_o t)dt \end{aligned} \tag{2.8}$$

Armed with Fourier series and superposition we may now model a non-sinusoidal signal as a superposition of an infinite sum of sources. About half the work can be cut in half by recognizing that sin lags cos by 90^o, so

$$\begin{aligned} a_o &= \frac{1}{T}\int_0^T v(t)dt \\ a_k &= \frac{2}{T}\int_0^T v(t)cos(nw_o t)dt \\ b_k &= \frac{2}{T}\int_0^T v(t)cos(nw_o t - 90^o)dt \end{aligned} \tag{2.9}$$

Section 3

AC Power

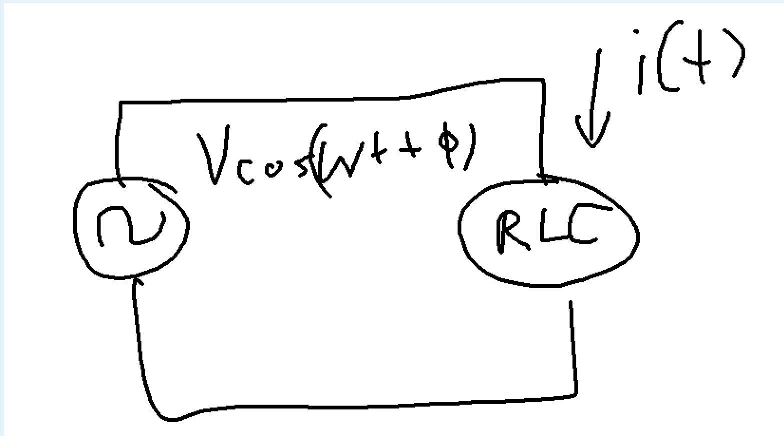
Definition 1 **Instantaneous Power**: $p(t) = v(t) \times i(t)[W, \frac{J}{s}]$

Example For a circuit with a voltage source, $v(t) = Vcos(\omega t)$ and a resistor Ω, $i(t) = Icos(\omega t)$, $p(t) = VIcos^2(\omega t) = \frac{VI}{2}(1 + cos(2\omega t))$

Definition 2 **Average Power over Cycle**: $P(t) = \frac{1}{T}\int_o^T p(t)dt = \frac{VI}{2}$

If we were to plot the instantaneous power we see that due to the sinusoidal response there are times where 0 power is supplied. This will always be true for a single phase power supply; real-world supplies always have multiple phases; this is why computer PSUs always contain a ton of capacitors.

Definition 3 **Reactive Power**: Q



If $\phi_i = 0$ and taking $\phi = \phi_v$,

$\phi = \phi_v - \phi_i$

$$\begin{aligned} p(t) &= V\cos(\omega t + \phi) * I\cos(\omega t) \\ &= \frac{VI}{2}\cos(\phi) + \cos(2\omega t + \phi) \\ &= \underbrace{\frac{VI}{2}\cos(\phi)(1 + \cos(2\omega t)}_{\text{real power e.g. heat}} - \underbrace{\frac{VI}{2}(\sin\phi\sin 2\omega t)}_{\text{stored and released back to source}} \end{aligned} \tag{3.1}$$

Taking the average power of the reactive power we get

$$P_{avg} = \frac{VI}{2}\cos\phi \tag{3.2}$$

Another quantity, reactive power, can be defined with regards to the energy sloshing back and forth:

$$Q = \frac{VI}{2}\sin\phi \tag{3.3}$$

Subsection 3.1

Lecture 4

Definition 4

Displacement factor

$DF \equiv \cos\phi$

Where ϕ is the angle measured from the $\hat{V}$ to the $\hat{I}$ phasors. A $DF = 1$ means the system is transferring the most power possible.

- Lagging DF: ϕ is -'ve
- Leading DF: ϕ is +'ve'

We can also write P, Q from phasors.

$$
\begin{aligned}
P &= \frac{\hat{V}\hat{I}}{2}\cos(\phi_v - \phi_i) \\
&= \frac{1}{2}\hat{V}\hat{I}Re\{e^{j\phi_v - \phi_i}\} \\
&= \frac{1}{2}Re\{Ve^{j\phi v} \cdot Ie^{-j\phi_i}\} \\
&= \frac{1}{2}Re\{\hat{V}\hat{I}^*\}
\end{aligned} \tag{3.4}
$$

$$
\begin{aligned}
Q &= \frac{\hat{V}\hat{I}}{2}\sin(\phi_v - \phi_i) \\
&\vdots \\
&= \frac{1}{2}Im\{\hat{V}\hat{I}^*\}
\end{aligned} \tag{3.5}
$$

The expressions for Q and P are basically the same so we define a new quantity, complex power, which incorporates both the imaginary and real components.

Reference: Thomas 16.1, 16.2, 16.3

Definition 5 **Complex Power**

$$S = \frac{1}{2}\hat{V}\hat{I}* = P + jQ \tag{3.6}$$

3.1.1 Root Mean Squared (RMS) Values

A RMS value measures the average power of a periodic signal. Consider a circuit with an AC voltage source with $v(t)$ flowing through a resistor.

The power is:

$$p(t) = v(t)i(t) = v(t)\frac{v(t)}{R} = \frac{1}{R}v(t)^2 \tag{3.7}$$

The average power is therefore

$$P = \frac{1}{R}\int_0^T v(t)^2 dt \tag{3.8}$$

Evaluating this becomes easier by defining an useful quantity, RMS voltage

$$v_{rms} = \sqrt{\frac{1}{T}\int_0^T v(t)^2 dt} \tag{3.9}$$

And using v_{rms} we get a nice expression for average power,

$$P = \frac{1}{R}v_{rms}^2 \tag{3.10}$$

More generally speaking we can define RMS values of sinusoidal signals

Definition 6

$$v_{rms} = \sqrt{\frac{1}{2\pi}\int_0^{2\pi}(\hat{V}\cos wt)^2 dt} = \frac{1}{\sqrt{2}}\hat{V} \tag{3.11}$$

Plugging this into the expressions for complex power:

$$S = \underline{V} \cdot \underline{I^*} \tag{3.12}$$

Where $\underline{V}$, $\underline{I^*}$ are RMS phasors at a given common frequency.

And more generally yet, the RMS values of non-sinusoidal signals can be found with help of a Fourier expansion

Definition 7 Let $v(t) = \hat{v}_0 + \hat{v}_1 \cos(wt + \phi_1) + \dots$
Then,

$$v_{rms} = \sqrt{\hat{v}^2 + \frac{\hat{v}_1}{\sqrt{2}}^2 + \frac{\hat{v}_2}{\sqrt{2}}^2 \dots} = \sqrt{v_0^2 + \sum_{n=1}^{\infty}(\frac{V_m^2}{2})} \tag{3.13}$$

And if V_m is the rms value,

$$v_{rms} = \sqrt{v_0^2 + \sum_{n=1}^{\infty} V_m^2} \tag{3.14}$$

One thing to watch out for is that we could have a system with high voltage but near-zero current transferring little to no power. To account for this we look at the power factor 'PF'

Definition 8 **Power Factor**

$$PF \equiv \frac{\text{average power}}{\text{rms voltage} \cdot \text{rms current}} \tag{3.15}$$

For a sinusoidal V, I:

$$\begin{aligned} PF &= \frac{\frac{1}{2}\hat{V}\hat{I}\cos\phi}{\frac{\hat{V}}{\sqrt{2}} \cdot \frac{\hat{I}}{\sqrt{2}}} \\ &= \cos\phi \end{aligned} \tag{3.16}$$

If signals are not harmonics $PF = DF$. This can be a source of confusion.

For non-sinusoidal systems this becomes more difficult because, unlike pure signals, they may contain harmonics. In these systems either V, I, or both may contain harmonics. Generally in household power V is clean but I contains harmonics.

The effect of harmonics in currents is that $I_{\text{harmonics}}$ causes a higher I_{rms}; there is more current but no higher power! This reduces PF and causes $PF < DF$.

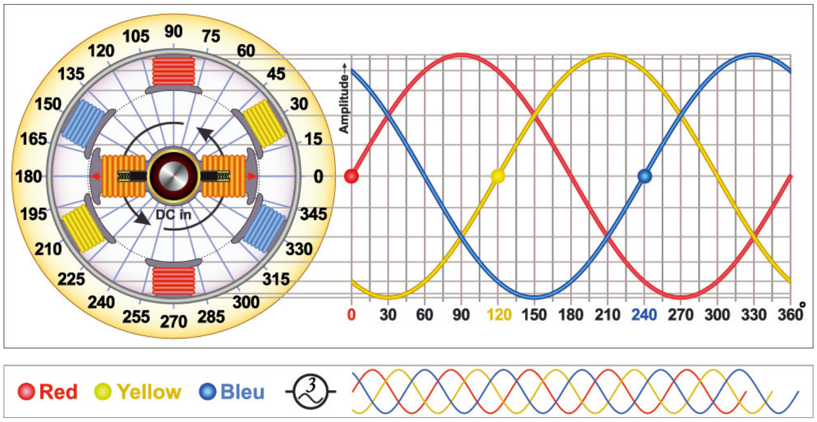
To summarize,

- In ideal systems, $PF = DF$ if all V and I are at one frequency.
- $PF = 1$ means no energy sloshing between load and source.

Harmonics are bad because they reduce the usefulness of the system There are very tight standards for how many harmonics one is allowed to inject into the system via generators or loads. See: textbook 16.6

Subsection 3.2
Lecture 5: Multi-Phase AC

AC power is generated by spinning a magnet between some coils. Some pixies get excited and by some Maxwell's equations and EMF and ECE259 we get a voltage induced in the coils.



Current from a generator with a single pair of coils is *single phase*. Most generators, like the one in the picture, have three pairs of coils and therefore generate three phases. For a typical three-phase setup with coils arranged at $0^o, 120^o, 240^o$, the voltages can be found with a little bit of trigonometry.

$$\begin{aligned} v_a(t) &= \sqrt{2}V\cos\omega t \rightarrow \underline{V_a} = V\lfloor 0 \\ v_b(t) &= \sqrt{2}V\cos(\omega t - 120) \rightarrow \underline{V_b} = V\lfloor -120^o \\ v_c(t) &= \sqrt{2}V\cos(\omega t - 240) \rightarrow \underline{V_c} = V\lfloor 240^o \end{aligned} \tag{3.17}$$

And similar expressions may be derived for single-phase and two-phase power.

The reason why three-phase power is typically used has to do with efficiency of power transfer relative to the amount of wires and copper needed. Whereas single-phase power requires two wires to carry power and two-phase power requires four wires, three-phase power can be transmitted over 6, 4, or three wires. This saves a lot of copper as three-phase 3ϕ power can carry 50% more power than 2ϕ power for less copper.

In 3ϕ systems the voltages sum to zero; $v_a(t) + v_b(t) + v_c(t) = 0\forall t$

Four-wire three-phase power:

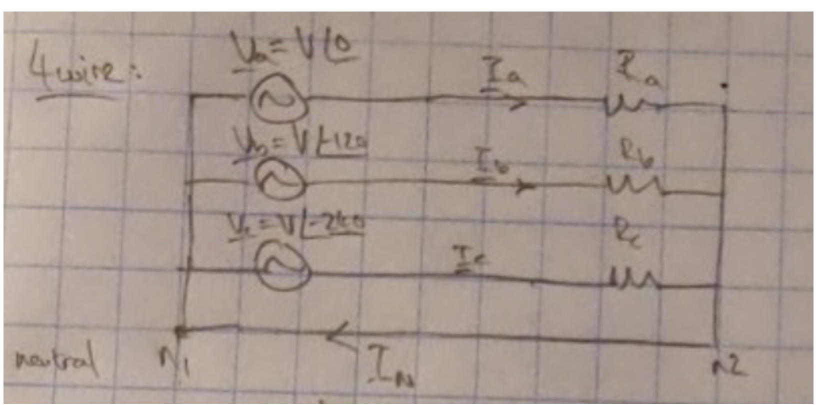


Figure 2. A four wire system for three phase power

If $R_a = R_b = R_c = R$, we have a balanced load and then the currents are related by $i_n = i_a(t) + i_b(t) + i_c(t) = 0$.

Proof: just substitute the condition earlier that voltages sum to zero and the fact that all resistances are equal into $I = \frac{V}{R}$

Three-wire systems drop the fourth neutral wire since it carries no current. This can be problematic if the load is not balanced; though $I_a + I_b + I_c = 0$ will still hold true since there is no return path, the nodes at either end of the AC source and resistor pair would have differing voltages.

If the system is balanced then to save ourselves from drawing everything out all the time, only a single diagram is drawn for a characteristic phase and then solved once.

Example

$$Z_a = Z_b = Z_c = Z \tag{3.18}$$

$$\underline{V_b V_a} e^{\frac{-j2n}{3}} \tag{3.19}$$

$$\underline{V_c V_a} e^{\frac{-j4n}{3}} \tag{3.20}$$

Then,

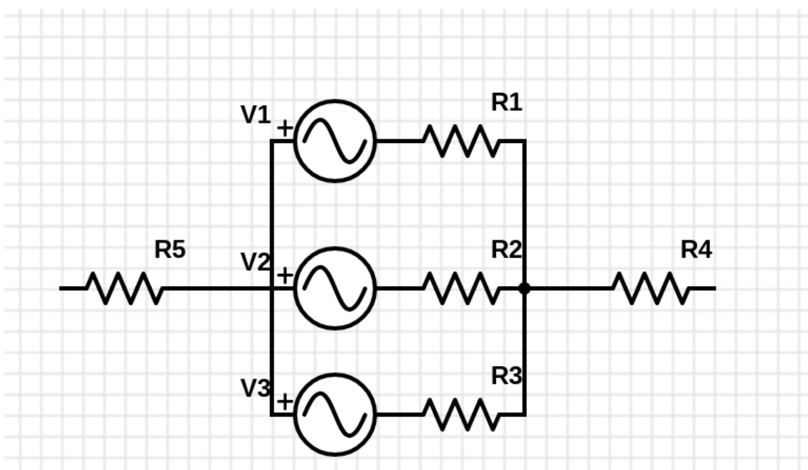
$$\underline{I_a} = \frac{\underline{V_a}}{Z} \tag{3.21}$$

$$\underline{I_b} = \frac{\underline{V_b}}{Z} = \frac{\underline{V_a}}{Z} e^{\frac{-j2n}{3}} \tag{3.22}$$

$$\underline{I_b} = \frac{\underline{V_c}}{Z} = \frac{\underline{V_a}}{Z} e^{\frac{-j4n}{3}} \tag{3.23}$$

And then the solutions for phase b and c are the same as that for phase a except for the $120^o, 240^o$ offsets.

Because of this property, instead of drawing three separate diagrams for each phase, we can just draw one diagram and then solve for the other two phases.



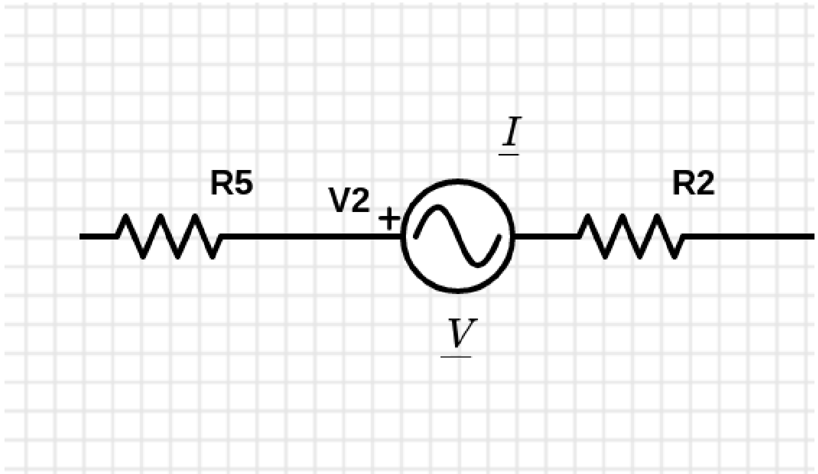


Figure 3. Condensed diagram for three phase power

SUBSECTION 3.3
Lecture 6: Y and $\triangle$ connections

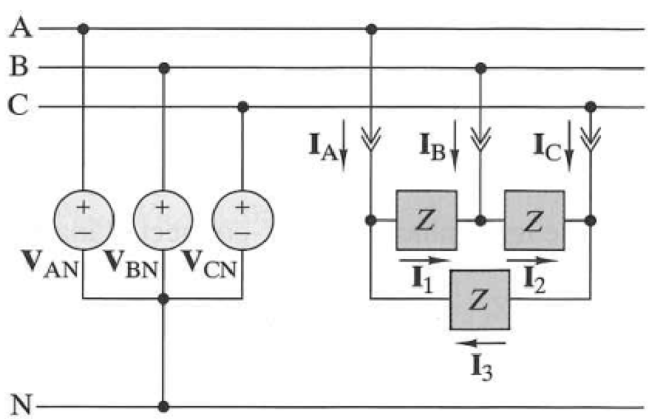
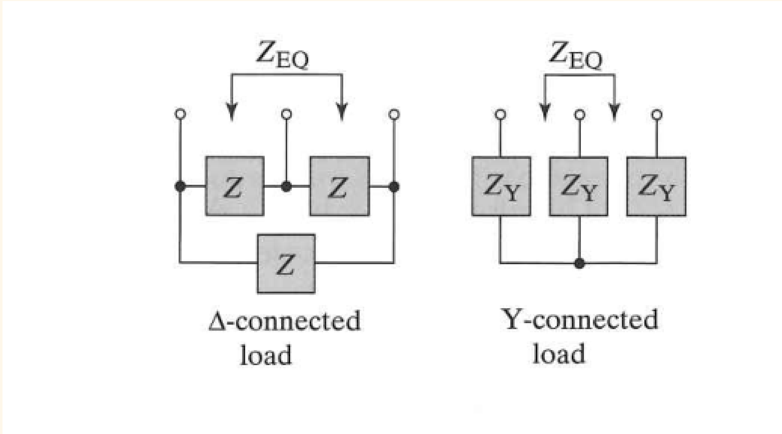


Figure 4. Three-phase system with a Y connected source and a Δ connected load

The current to ground I_N is always 0 for a Δ connected load. I_N is zero for a Y cnonnected load if Y has no neutral connection and the load/source are balanced.

Theorem 1 $Y - \Delta$ **conversion**
Any Δ load can be converted to a equivalent Y connected load



The derivation is in the textbook.

PART II

ECE352: Computer Organization

Taught by Prof. Andreas Moshovos

SECTION 4

Admin stuff

SUBSECTION 4.1

Lecture 1

- Lecture recordings on YouTube
- Online notes: https://www.eecg.utoronto.ca/ moshovos/ECE352-2022/
- Course will cover the following:
  - C to assembly
  - How to build a processor that works
  - Intro to processor optimizations
  - Peripherals
  - OS support (Maybe)
  - (Maybe) Arithmetic circuits
  - Use NIOS II and cover a little bit of RISC-V

4.1.1 Mark breakdown

- Labs 15%
- project 5%
- midterm 30%
- Final 50%
- All exams will be open notes/book/whatever except another person/service helping you.

Section 5

Preliminary

Subsection 5.1

Lecture 2: Using binary quantities to represent other things

Computers can represent information in bits; 0/1. Though they don't necessarily know or care what bits are, we may assign our own arbitrary meaning to them – usually numbers with the help of positioning; the LSB represents 2^0 and so forth.

C types

Or, just `#include <stdint.h>`...

- int: 32b (word)
- char: 8b (byte)
- short: 16b (half word)
- long: 32b (word)
- long long: 64b

Signed numbers may be represented in a number of ways.

- Sign bit (make MSB represent positive or negative numbers and then the remaining $n-1$ bits represent the number. Con: hardware impl sucks because requires if/else
- Two's complement [1]. Pro: only need to implement adders on hardware and then negative numbers will work just like any other except must be interpreted differently. Positive numbers would always start with a 0 and negatives would start with 1. So the range of possible values becomes $-(2^{n-1}-1), +2^{n-1}-1$

[1] Flip bits, add one. Intuition; in 3 bit system, adding 7 to 1 would result in 8 which would get truncated to 0.

Adding together binary numbers can also cause overflow; $(A+B) \geq A, (A+B) \geq B$ may not always be true. Also, when we work with these types we always use all the bits. This has implications when working with values of different lengths.

- `char b = -1` (1111 1111))
- `short int c = -1` (0000 0000 0000 0001))
- `a = b + c` 0000 0001 0000 0000
- In order to deal with this we must `cast` the `char` to a `short int`. This is done via sign extension which prepends 0s or 1s [2] to the `char` so that math can be done on it.

[2] two's complement

5.1.1 Floating Point Numbers

Whereas fixed point numbers i.e. $\$5.25$ can be represented just as how an integer would be represented but with the understanding that the user would interpret it as having a decimal point somewhere that indicates the position of 2^0. This decimal point would be the same for all numbers of that type, i.e. we could have a six bit number that has places $2^2 2^1 2^0 2^{-1} 2^{-2}$ This is common in embedded systems and how it is formatted isn't super clearly standardized.

Lemma 1 Reference: What Every Computer Scientist Should Know About Floats

Definition 9 **IEEE 754 Floating Point**
This is a single precision 32 bit float

$$\texttt{S EEEEEEEE MMMM MMMM MMMM MMMM MMMM MMM} \quad (5.1)$$

The most significant `S` bit is the sign bit, bits 30 through 23 `E` form the exponent which is an unsigned integer, and 22 through 0 form the (`M`)antissa. The number being represented can be found using the following:

$$(-1)^S \times 2^{(E-127)} \times 1.\text{Mantissa} \quad (5.2)$$

Example

For example, given the following float:

1 10000001 10000000000000000000000

So S = 1, E = 10000001 = 129 and Mantissa = 10000000000000000000000. The number is therefore

$$(-1^1) \times 2^{(129-127)} \times 1.10000000000000000000000 = -6.0 \quad (5.3)$$

IEEE754 also defines 64 bit floating-point numbers. They behave the same except for now having an 11 bit exponent, the bias being 2047[3], and the mantissa having 52 bits.
A few special cases are also available to represent other quantities

- If E=0, M non-zero, value=$(-1)^S \times 2^{-126} \times 0.M$ (denormals)
- If E=0, M zero and S=1, value=-0
- If E=0, M zero and S=0, value=0
- If E=1...1, M non-zero, value=NaN 'not a number'
- If E=1...1, M zero and S=1, value=-infinity
- If E=1...1, M zero and S=0, value=infinity

[3] *Instead of 1023*

In c, `float` is a 32 bit float and `double` is 64

Floating-point numbers are inherently imprecise. Addition and subtract are inherently lossy; the mantissa window may not be large enough to capture the decimal points. Multiplication and division just creates a ton of numbers.
Converting real numbers to IEEE754 floats, here using 37.64 as an example, can be done as follows

- Repeatedly divide the part of the number > 0 by 2 and get the remainders, i.e. 37/2 = 18, rem = 1 -> 18/2 = 0, rem = 0 -> 4/ 2 = 2, rem = 0, 2/2 = 1, rem = 0, 1/2 = 0, rem = 1; E = 100101
- Do the same for the part of the number past the decimal, but multiplying by two and checking if > 1: 0.64 * 2 = 1.28 -> 1, 0.28 * 2 = 0.56 -> 0, 0.56 * 2 = 1.12 -> 1 ... and so forth. At some point we will hit a cycle but we'll just take the N_{mantissa} of digits.

So the full number is 01000010000101101000111101011111

There are more floating point formats introduced by nvidia and google such as a half-precision or 8-bit float designed to reduce memory use for machine learning

Section 6

NIOS II

Subsection 6.1

Lecture 3: Behavioural Model of Memory

Computers can be described as a set of units, each of which interact with each other and the outside world in a specified way. For example, modern computers tend to have memory units, processing units, display units, and so forth. Each unit has a set of inputs and outputs, and a set of rules that govern how the unit behaves. This gives the manufacturer flexibility in how they want to implement a unit, as long as the unit behaves as specified. When designing these operational units it is important to strike a balance between functionality and specificity; if the unit is too specific it will be difficult to implement, but if it is too general it will be difficult to use.

Specification is the description of what an unit should do, and **implementation** is how it actually does it. For example, an `OR` gate can be specified as a truth table and then implemented via transistors or a person in a box.

6.1.1 Memory

Memory is a unit that stores information and is usually represented as a vector of elements, usually a byte (8 bites). Each element, or memory location, contains a binary quantity and has an associated *address*. The address is a number that uniquely identifies the location of the element in the vector, and is **permanently fixed** at time of manufacture. Most systems are byte-addressable, meaning that there is an unique address for each byte in the memory. The collection of all addresses is called the **address space** of the memory, which is typically a power of two. Modern systems tend to be 32 or 64 bit, meaning that the address space is 2^{32} or 2^{64} elements long.

For each memory location there there are two operations available

- **Read**: Read the value stored at a given address
- **Write**: Write a value to a given address

Typical memory behaviour models define that the order of memory operations matters, i.e.

1. `store 0x10, 0xf0`
2. `store 0x20, 0xf0`

We would see that `0xf0` contains the value `0x20` and not `0x10` due to the sequential execution model. Memory that adheres to the sequential model offers operations that are **atomic**; the operations are performed on its own with no interaction or overlap with anything else.

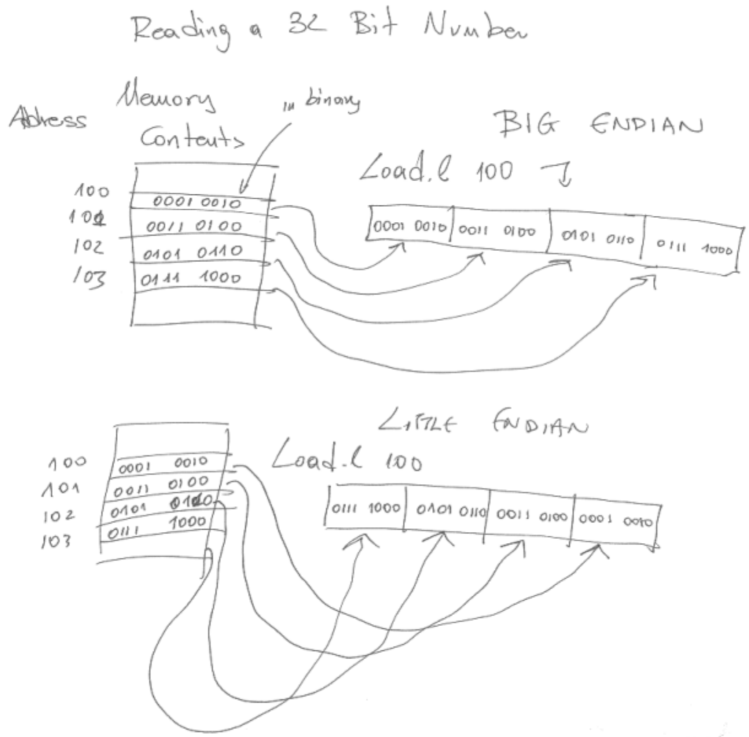
`In this case it is convenient to draw memory as an array where each row comprises four consecutive bytes.`

0x00 0000	0x11	0x22	0x33	0x44
0x00 0004	0xff	0x88	0x62	0x51
. . .				
0xff ffff				

Systems are generally also addressable by words, halfwords, and bytes. Different architectures have different constraints on allowing unaligned access[4]

[4] Aligned access only means to allow [only] reads or writes for a data size i.e. halfword to an address divisible by the size of said data type. For example an longword access on our development board would be at an address divisible by 4

Endianness refers to the order in which bytes are stored in memory. Though some processors are big-endian, most modern processors are little-endian. The NIOS II used for this course is little-endian.



6.1.2 Physical Interface

What physical interfaces would be necessary to implement this behavioural model?

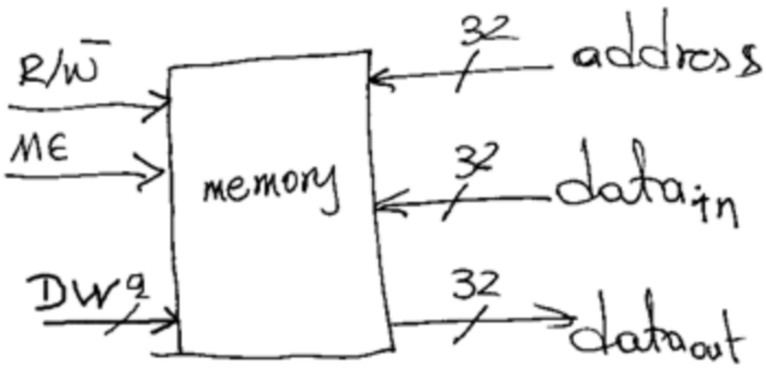
Given a summary of requirements as follows:

1. Read and write operations
2. Addressable by byte, word, longword
3. 24 bit address
4. 32 bit for writing
5. 32 bit for reading
6. signal for do nothing

A single bit signal can be used to indicate whether the memory is reading or writing, and a two bit signal can be used to specify if we're interested in addressing by byte, word, or longword. The address is 24 bits, so we need 24 address lines. As for reading/writing data, we have the option of having two 32 bit data lines, or multiplexing a single 32 bit line. A single bit signal can be used to indicate to do nothing or not.

The use of a single bit signal to indicate 'do nothing' is necessary because a physical device won't be able to change all signals instantaneously, so we use it to tell the memory to wait until these transient effects die off

One way of multiplexing the data lines is to use a tri-state buffer, which is a buffer that can be enabled or disabled. When enabled, the buffer acts as a normal buffer, but when disabled, the output is disconnected from the input. On the other hand this means that our memory chip would not support simultaneous reads or writes.



Subsection 6.2

Lecture 4: NIOS II Programming Model

The NIOS II assumes a `32-bit` address space where each address holds a single byte. Each byte is addressable, and three data types are supported. Halfword and word accesses must be aligned.

- **Byte**: 8 bits
- **Halfword**: 16 bits
- **Word**: 32 bits

The NIOS II also has a set of registers

- 32 general purpose 32 bit registers
  - `r0` is always zero
- 6 control registers, 32 bits each
- Program counter (PC), 32 bits

Many operations can be synthesized using another operation involving zero, i.e. assignment `A=B` can be implemented as `A = B + 0`

There are certain conventions for the use of registers, which are as follows:

Register	Name	Function	Register	Name	Function
r0	zero	0x00000000	r16		
r1	at	Assembler Temporary	r17		
r2		Return Value	r18		
r3		Return Value	r19		
r4		Register Arguments	r20		
r5		Register Arguments	r21		
r6		Register Arguments	r22		
r7		Register Arguments	r23		
r8		Caller-Saved Register	r24	et	Exception Temporary
r9		Caller-Saved Register	r25	bt	Breakpoint Temporary (1)
r10		Caller-Saved Register	r26	gp	Global Pointer
r11		Caller-Saved Register	r27	sp	Stack Pointer
r12		Caller-Saved Register	r28	fp	Frame Pointer
r13		Caller-Saved Register	r29	ea	Exception Return Address
r14		Caller-Saved Register	r30	ba	Breakpoint Return Address (1)
r15		Caller-Saved Register	r31	ra	Return Address

Notes to Table 3–1:
(1) This register is used exclusively by the JTAG debug module.

6.2.1 Adding Two Numbers

As an exercise, let's see how we can implement the following piece of code in NIOS II assembly

```
unsigned int a = 0x00000000;
unsigned int b = 0x00000001;
unsigned int c = 0x00000002;

a = b + c;
```

Register-only version

```
addi r9, r0, 0x1
addi r10, r0, 0x2
add r9, r10, r11
```

`addi` stands for 'add intermediate', the only difference being that the second operand is a number. It is used to set a constant

In general, most instructions take the form of `operation destination, source1, source2`.

Breaking it down even further we can see that these assembly instructions actually perform a number of steps

```
addi r9, r0, 0x1
        ; 1. read r0
        ; 2. Add value read in step 1 with 0x1
        ; 3. Write result of step 2 to r9
        ; 4. increment PC to next instruction
addi r10, r0, 0x2
        ; 1. read r0
        ; 2. Add value read in step 1 with 0x2
        ; 3. Write result of step 2 to r10
        ; 4. increment PC to next instruction
add r9, r10, r11
        ; 1. read r10
        ; 2. read r11
        ; 3. Add values read in steps 1 and 2
        ; 4. Write result of step 3 to r8
        ; 5. increment PC to next instruction
```

What about 32 bit constants? An unfortunate quirk is that `addi` only supports 16 bit constants, so we need to use `ori` to set the upper 16 bits of the register.

```
movhi r9, 0x1122
        ; Sets the upper 16 bits of r9 to 0x1122
        ; and the lower 16 bits to zero
ori r9, r9, 0x3344
        ; bitwise OR the value in r9 with 0x3344
        ; which will set the lower 16 bits to 0x3344
```

This is a PITA so NIOS II offers a few pseudo-instructions to make this easier

```
movi rX, Imm16
        ; sets rX to the sign-extended (signed) 16 bit immediate
movui rX, Imm16
        ; sets rX to a zero-extended unsigned 16 bit immediate
movia rX, Imm32
        ; sets rX to a 32 bit immediate
```

- Footnote1: `movia` does not use the `movhi` and `ori` instructions to create a 32-bit immediate but rather a `movhi` and a `addia`. `addi` will sign extend it's 16-bit field so some adjustment might be needed for whatever is being passed to `movhi`.
- Footnote2: movhi r9, %hi(0x11223344) is equivalent to movhi r9, 0x1122. Ori r9, %lo(0x11223344) is equivalent to ori 9, 0x3344. That is, %hi(Imm32) returns the upper 16-bits of Imm32 and %lo(Imm32) the lower 16 bits.
- Footnote3: movhi r9, %hiadj(0x11223344) followed by addi r1, %lo(0x11223344) is the correct way of creating a 32-bit immediate using movhi and addi. %hiadj(Imm32) returns the upper 16 bits of the immediate as-is or incremented by 1 if bit 15 is 1. Think why this is necessary based on footnote 1.
- Footnote4: %hi(), %lo(), and %hiadj() are macros supported by the assembler. They are not NIOS II instructions. They get parsed during compile time.

6.2.2 Adding two numbers using memory

NIOS II is a load/store architecture which means that all data manipulation happens only in registers.

```
; read b from memory into r9
movhi r11, 0x0020
ori   r11, r11, 0x0004
ldw   r9, 0x0(r11)

; read c from memory into r10
movhi r11, 0x0020
ori   r11, 0x0008
ldw   r10, 0x0(r11)

; add, then store into r8
add   r8, r9, r10

; store r8 into memory
movhi r11, 0x0020
ori   r11, r11, 0x0000
stw   r8, 0x0(r11)
```

The new instructions introduced here are

```
ldw rX, Imm16(rY) ;; 'load word' from memory
;; rX, rY registers, Imm16 is a 16 bit immediate
;; TLDR; Rx = mem[rY + sign-extended(Imm16)]
; 1. read rY
; 2. sign-extend Imm16 to 32bits
; 3. adds the result of step 1 and 2
; 4. reads from memory a word (32 bit) using the result of step 3 as the address
; 5. write the result of step 4 to rX
```

```
stw rX, Imm16(rY) ;; 'store word' to memory
;; rX, rY registers, Imm16 is a 16 bit immediate
;; TLDR; mem[rY + sign-extended(Imm16)] = rX
; 1. read rY
; 2. sign-extend Imm16 to 32bits
; 3. adds the result of step 1 and 2
; 4. write to memory rX using the result of step 3 as the address
```

This can be simplified using the `movia` macro

```
movia r11, 0x200004
ldw   r9, 0x0(r11)
movia r11, 0x200008
ldw   r10, 0x0(r11)
add   r8, r9, r10
movia r11, 0x200000
stw   r8, 0x0(r11)
```

In this lecture so far we have seen three addressing modes

1. Register addressing, i.e. `rX`
2. Immediate addressing, i.e. `Imm16`
3. Register indirect addressing with displacement, i.e. `Imm16(rY)`. This is how we calculate the referenced memory address. Register indirect refers to using a register's value to refer to memory, and 'displacement' refers to adding a constant prior to using the register value to access memory. Register indirect addressing is where we use a displacement of 0.

We can exploit register indirect addressing with displacement.

```
movhi r11, 0x0020
ori   r11, r11, 0x0004
ldw   r9, 0x0(r11)
;; can be replaced with
movhi r11, 0x0020
ldw r9, 0x4(r11)
```

Note that the value of `r11` does not change since the subsequent operations use an offset to that value.

Generally when we want to read memory from `A` we can use

```
movhi r11, (upper 16 bits of A)
ori   r9, r11, (lower 16 bits of A)
```

Care must be taken when the 16th bit of `A` is 1 since the addition that `ldw` performs will sign extend it to be a negative number, i.e.

```
movhi r11, 0x0020
ldw r9, 0x8000(r11)
;; this is incorrect because
;; will extend to 0xFFFF8000, which would result
;; in a final address of 0x001F800
```

This is where the macros `%hiadj(Imm32)` and `%lo(Imm32)` come in handy, since they will add 1 to the values if bit 15 of Imm32 is 1. This results in code that looks like this:

```
movhi r11, %hiadj(0x208000)
ldw r9, %lo(0x2080000)(r11)
;; will extend to 0xFFFF8000, which would result
;; in a final address of 0x001F800
```

Subsection 6.3

Lecture 5: Simple Control Flow

We have prior worked with straight-line sequences. In this lecture we will look at how to add control flow to our programs, i.e if-then-else, etc.

A pseudo-c program will be rewritten in assembly to illustrate the concepts.

```
unsigned int a = 0x00000000;
unsigned int b = 0x11223344;
unsigned int c = 0x22334455;

if (b == 0)
then a = b + c;
else a = b - c;
```

The `data` section contains stuff that you want to be initialized for you before the entry point of the program is called, e.g. global variables. This segment as a fixed size. The `text` or `code` segment contains executable instructions (typically read-only, unless the architecture allows self-modifying code) and typically resides in the lower parts of memory. `bss` contains static and global variables which are zero-initialized.

```
      .section .data
va:   .long 0x0
vb:   .long 0x11223344
vc:   .long 0x55667788


main:
      movia r11, va
      ldw   r9, 4(r11)
      beq   r9, r0, then
else:
      ldw   r10, 8(r11)
      sub   r8, r9, r10
      stw   r8, 0(r11)
      beq   r0, r0, after

then:
      ldw   r10, 8(r11)
      add   r8, r9, r10
      stwio r8, 0(r11)
after:
```

Definition 10 We encounter two new instructions in this snippet.

- `sub` is a subtraction instruction
- `beq`: a branch-if-equals instruction

The `beq` instruction takes the general form

`beq RX, rY, label`.
This instruction will compare the values of rX and rY and if the condition is true then the program counter will jump to the destination label. When the branch changes the program counter it is called a taken branch, otherwise it is non-taken. Non-taken branches fall through to the next instructions.
Other branch instructions include

- `br`: always/unconditional branch
- `bne`: branch if not equal
- `blt`: branch if less than, w/ signed comparison
- `bltu`: branch if less than, w/ unsigned comparison
- `bgt`: branch if greater than, w/ signed comparison
- `bgtu`: branch if greater than, w/ unsigned comparison

```
        .data
        .align 4 ;; Align to word size addresses which are faster to access
a: .word 0
b: .word 0x11223344
c: .word 0x55667788
        .text
        movia r11, a  ;; moves the address of a into r11
        ldw r9, 4(r11) ;; loads the value at address a+4 into r9
        ldw r10, 8(r11)
        add r8, r9, r10
        stw r8, 0(r11)
```

PART III

ECE355: Signal Analysis and Communication

Taught by Prof. Sunila Akbar

Section 7

Admin and Preliminary

Subsection 7.1

Lecture 1

- CT and DT signals
- A ton of LTI (Linear time invariant) systems
- Processing of signals via LTI systems
- Fourier transforms

- Sampling

7.1.1 Mark Breakdown

Table 1. Mark Breakdown

Homework	20
MT1	20
MT2	20
Final	40

- Continuous enclose in (), independent is t
- Discrete: enclose in [], independent is n

In many systems we are interested in power and energy of signals over an infinite time interval; $-\infty < \{t, n\} < \infty$

Theorem 2 **Energy for Complex Signals**

$$E_{[t_1,t_2]} = \int_{t_1}^{t_2} |x(t)|^2 dt \tag{7.1}$$

$$E_{[t_1,t_2]} = \sum_{n=n_1}^{n_2} |x(t)|^2 dt \tag{7.2}$$

Average Power for Complex Signals

$$P_{avg,[t_1,t_2]} = \frac{1}{t_2 - t_1} \int_{t_1}^{t_2} |x(t)|^2 dt \tag{7.3}$$

$$P_{avg,[t_1,t_2]} = \frac{1}{n_2 - n_1 + 1} \sum_{n=n_1}^{n_2} |x(n)|^2 \tag{7.4}$$

Section 8

Transformations

Subsection 8.1

Lecture 2

Most of this lecture was review. When applying transforms just note to always scale, *then* shift, i.e.

1. $y(t) = x(\alpha t)$
2. $y(t) = x(\alpha t + \frac{\beta}{\alpha})$

Definition 11 **Fundamental Period**

$$x_t = x(t + mT), m \in \mathbb{Z} \tag{8.1}$$

The fundamental period, T_o is the smallest positive value of T for which this holds true

Definition 12 **Even signals**

$$x(t) = x(-t) \tag{8.2}$$

Definition 13 **Odd signals**

$$x(t) = -x(t) \tag{8.3}$$

Theorem 3 Any signal can be broken into an even and odd component

$$\begin{aligned} x(t) = Ev\{x(t)\} = \frac{1}{2}[x(t) + x(-t)] \\ x(t) = Od\{x(t)\} = \frac{1}{2}[x(t) + -x(-t)] \end{aligned} \tag{8.4}$$

SUBSECTION 8.2

Lecture 3

Again, most of this lecture was review from the waves portion of PHY293 from last year or some other course prior.

A complex exponential and sinusoidal system can be represented as

$$x(t) = Ce^{at} \tag{8.5}$$

Where C, a are complex numbers.

Two cases may occur.

If a imaginary and C is real we have, depending on ω, either a constant signal or a periodic sinusoidal system.

$$x(t_{=}e^{j\omega_0 t}) \tag{8.6}$$

- Important property: this is periodic, i.e. $Ce^{j\theta_0 t} = Ce^{j\theta(t+T)}$
- Implies that $e^{j\omega_0 T} = 1$
- Implies that for $\omega \neq 0 \to T_0 = \frac{2\overline{n}}{|\omega_0|}$

On the other hand, if a imaginary and C complex, we have a periodic signal with $T = \frac{2\overline{n}}{\omega_0}$

$$x(t) = Ce^{j\omega_0 t} = |C|e^{j\omega_0 t + \phi} = |C|\cos(\omega_o + \phi) + j|C|\sin(\omega_0 + \phi) \tag{8.7}$$

The energy of the signal is given by (7.1), or

$$E_{period} = \int_0^{T_0} |e^{j\omega_0 t}|^2 dt = \int_0^{T_0} 1 dt = T_0 \tag{8.8}$$

$$P_{period} = \frac{E_{period}}{T_0} = 1 \tag{8.9}$$

Recall: implication that $e^{j\omega_0 T} = 1$, therefore the quantity inside the integral evaluates to 1

8.2.1 General Continuous Complex Exponential Signals

The most general case of a complex exponential can be represented as a combination of the real exponential and the periodic complex exponential;

$$C = Ce^{at} \tag{8.10}$$

and

$$a = r + j\omega_0 \tag{8.11}$$

can be combined to give

Definition 14

$$Ce^{at} = |C|e^{rt}e^{j\omega_0 t+\theta} \tag{8.12}$$

Euler's relation can be used to simplify this to

$$Ce^{at} = |C|e^{rt}\left(\cos(\omega_0 t + \theta) + j\sin(\omega_0 t + \theta)\right) \tag{8.13}$$

By inspection we can see that the signal has the following properties:

1. $r = 0$: real and imaginary parts of sinusoidal
2. $r > 0$: sinusoidal signal with exponential growth
3. $r < 0$: sinusoidal signal with exponential decay

I will be skipping notes on the discrete case as it is essentially the same as the continuous case, but with the following differences

Table 2. Comparison of continuous and discrete complex exponential signals

$e^{j\omega_0 t}$	$e^{j\omega_0 n}$
Distinct signals for distinct ω_0	identical signals for distinct $\omega_0 \in \{\omega_0 \pm 2\pi i, i \in \mathbb{Z}\}$
Periodic for any ω_0	Periodic only if $\omega_0 = \frac{2\pi m}{N}$ for integers $N > 0, m$
Fundamental frequency ω_0	Fundamental frequency $\frac{\omega_0}{m}$
Fundamental period $\omega_0 = 0 \rightarrow$ undefined, otherwise $T_0 = \frac{2\pi}{\omega_0}$	Fundamental period $\omega_0 = 0 \rightarrow$ undefined, otherwise $T_0 = m\frac{2\pi}{\omega_0}$
	Since unique ω does not mean unique signal, pick $0 \leq \omega_0 \leq 2\pi$ or $-\pi \leq \omega_0 \leq \pi$

SUBSECTION 8.3

Lecture 4: Step and Impulse Functions

One of the simplest discrete-time signals is the **unit impulse**[5] function, $\delta[n]$

[5] or unit sample

Definition 15

$$\delta[n] = \begin{cases} 0 & n \neq 0 \\ 1 & n = 0 \end{cases} \tag{8.14}$$

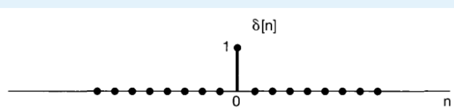


Figure 1.28 Discrete-time unit impulse (sample).

Another basic signal is the **unit step** function, $u[n]$

Definition 16

$$u[n] = \begin{cases} 0 & n < 0 \\ 1 & n \geq 0 \end{cases} \tag{8.15}$$

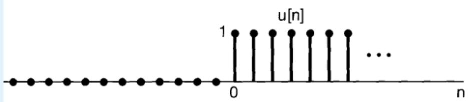


Figure 1.29 Discrete-time unit step sequence.

The unit impulse function is the first difference of the discrete time step function, i.e.

$$\delta[n] = u[n] - u[n-1] \tag{8.16}$$

And the unit step function is the running sum of the unit impulse function, i.e.

$$u[n] = \sum_{m=-\infty}^{n} \delta[m] \tag{8.17}$$

This can be rewritten with $k = n - m$ to make a more convenient expression for moving the function along $-\infty \ldots 0 \ldots \infty$

$$u[n] = \sum_{k=0}^{\infty} \delta[n-k] \tag{8.18}$$

Theorem 4 The unit impulse function $\delta[n - n_0]$ can be used to sample a function at a specific $n = n_0$ since the impulse function will take on the value 0 for all values of $n \neq n_0$

$$x[n]\delta[n-n_0] = x[n_0]\delta[n-n_0] \tag{8.19}$$

The continuous equivalents of the unit impulse and unit step functions are defined similarly

Definition 17

$$u(t) = \begin{cases} 0 & t < 0 \\ 1 & t > 0 \end{cases} \tag{8.20}$$

Likewise, the continuous unit step function is a running ~~sum~~ integral of the continuous unit impulse function

$$u(t) = \int_{-\infty}^{t} \delta(\tau) d\tau \tag{8.21}$$

A relationship analogous to the discrete case can be found for the continuous case; the continuous unit impulse function can be thought of as the first derivative of the continuous-time unit step function

Definition 18

$$\delta(t) = \frac{du(t)}{dt} \tag{8.22}$$

(8.22) is discontinuous at $t = 0$ so it is non-differentiable. We can address this by considering an approximation of (8.22) for a Δ short enough to not matter for any practical purpose

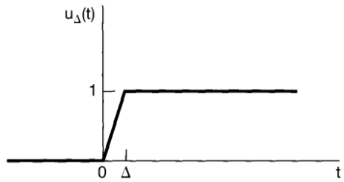


Figure 1.33 Continuous approximation to the unit step, $u_\Delta(t)$.

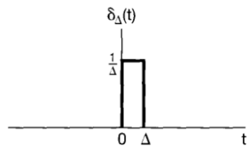


Figure 1.34 Derivative of $u_\Delta(t)$.

(8.21) can be rewritten as follows to make it more convenient to use along $\sigma \in -\infty \ldots 0 \ldots \infty$.

$$u(t) = \int_0^\infty \delta(t - \sigma) d\sigma \tag{8.23}$$

Theorem 5 And by the same argument as for the discrete case, the continuous impulse function has an important sampling property.
For any arbitrary point t_0,

$$x(t)\delta t(t - t_0) = x(t_0)\delta(t - t_0) \tag{8.24}$$

PART

IV

ECE360: Electronics

Taught by Prof. Khoman Phang

Section 9

Admin and Preliminary

Subsection 9.1

Lecture 1

9.1.1 Mark Breakdown

Table 3. Mark Breakdown

Test 1	15
Test 2	20
Homework	10
Labs	12
Final	43

9.1.2 Diodes

Diodes are an electronic valve which causes current to only flow in one direction. An ideal diode is an open circuit in the closed direction and a closed circuit in the other, so the current is always in the direction of the arrow (+'ve @ arrow base, -'ve at arrow point)[6].

[6] *recall: passive sign convention*

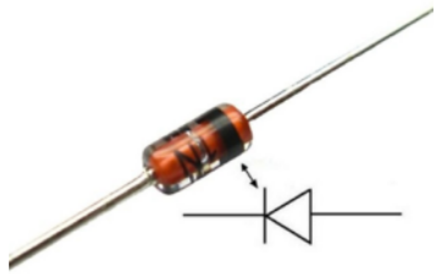
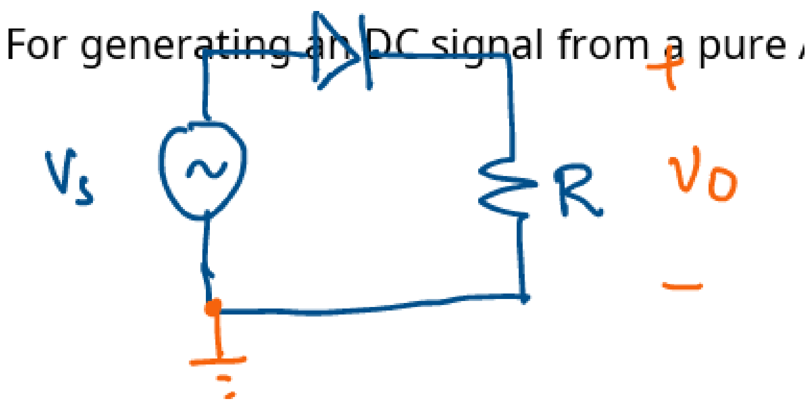
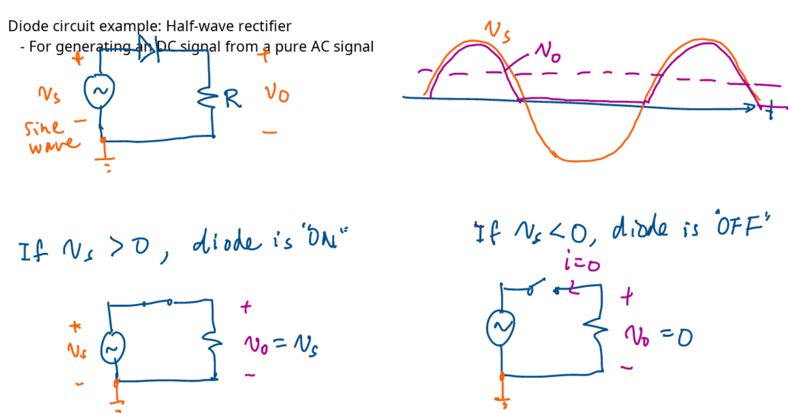


Figure 5. A diode and its symbol

An example of a diode circuit is the half-wave rectifier which turns an AC signal to a DC signal

Can take oscilloscope over resistor to see that a pure DC signal has been generated





Section 10

Diodes

Subsection 10.1

Lecture 2

More formally, off/on for diodes should be referred to as:

- Off ↔ reverse bias
- On ↔ forwards bias

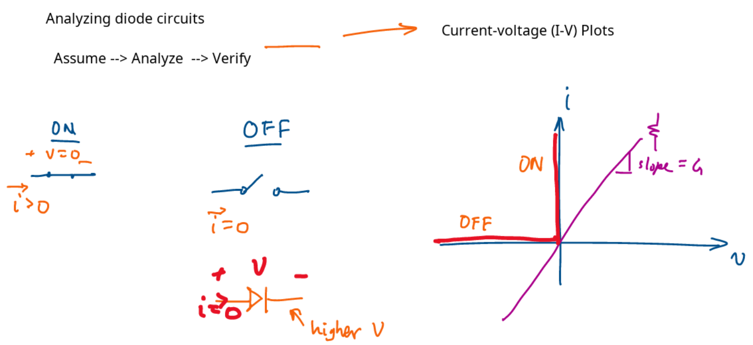
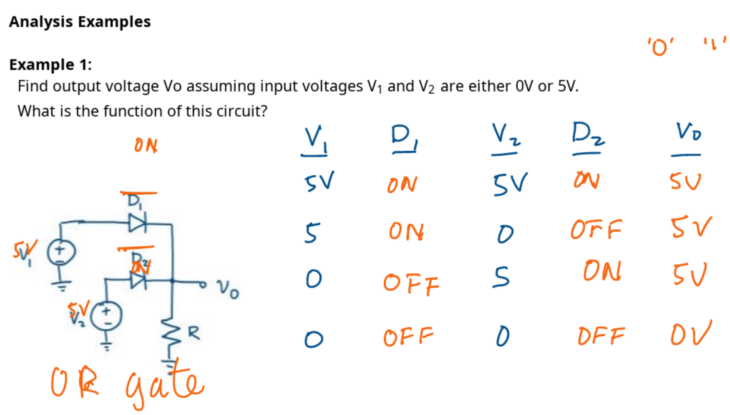
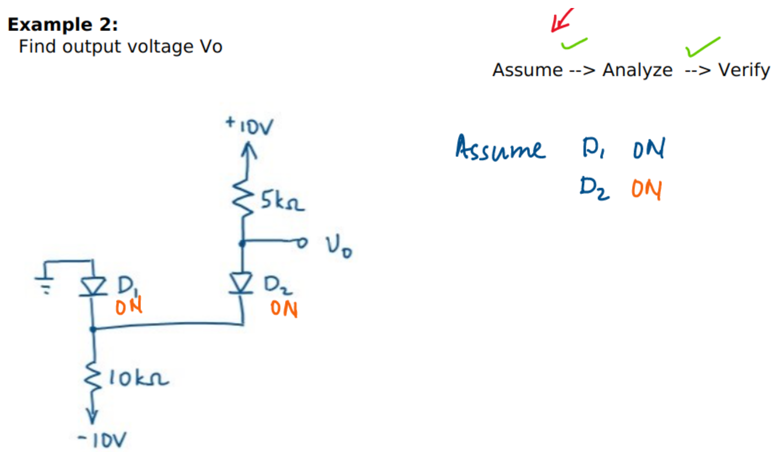
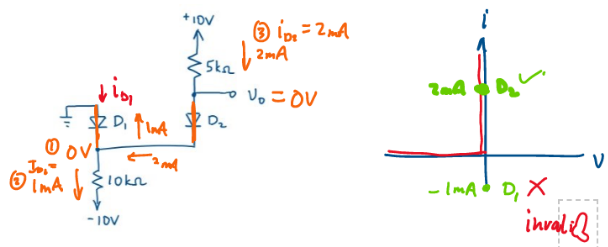


Figure 6. General steps for analyzing non-linear circuits. Note plotting out expected response

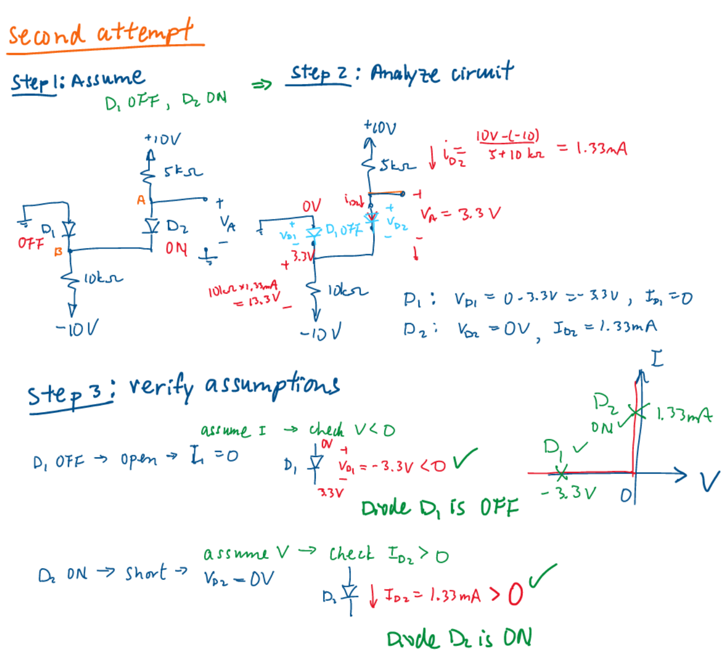


An example of how this is used in circuit design is to manage two power sources. Consider an Arduino that could be powered by an AC adapter or by a computer's USB port. This circuit would choose the higher voltage source and prevent back-flow into the other power source due to any potential power differentials. It is also effectively an OR gate





In this example the initial assumption was incorrect.
Let's try another analysis with D_1 off and D_2 on:



If we were to do this brute force we'd have to consider 4 cases, so it's important to build up some sort of intuition for the circuit.

SUBSECTION 10.2
Lecture 3

Today we're going to look at the characteristics of real diodes.

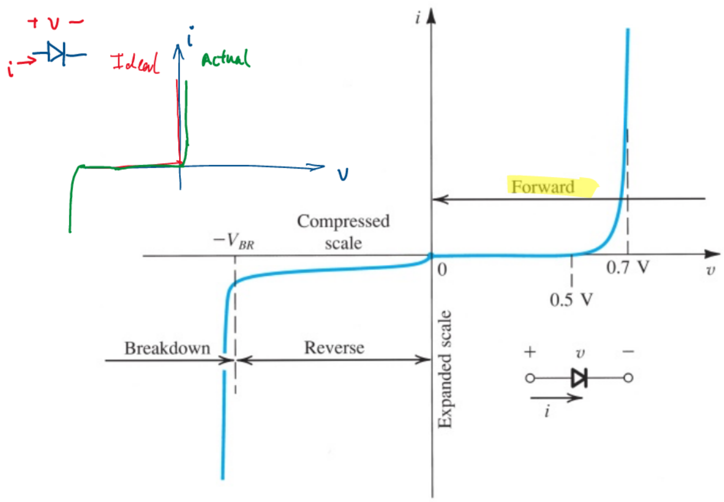


Figure 4.8 The silicon diode i–v relationship with some scales expanded and others compressed in order to reveal details.

Real diodes have a little bit of leakage current and also encounter a breakdown point where they're no longer able to block the current.

Theorem 6 **Forward Bias**

$$i = I_s(e^{\frac{V}{V_T}-1}) \tag{10.1}$$

Where:

$$V_T = \frac{kT}{q} \quad [V] \tag{10.2}$$

Most of the time we can assume that the circuit is at room temperature and that $v_T = 25mV$. Note that this value explodes when $V > V_T$ which is the breakdown point. When encountering a reverse bias $V_s < 0$, the -1 term comes in and causes $i \approx I_s$
The scale current is just a general constant which varies in range from $10^{-9} to 10^{-15} A$ and scales with temperature, doubling with every approximately 5^oC increase in temperature.
Note: the ideal diode equation can be rearranged to find an expression for voltages

$$V = V_T \ln{(\frac{i}{I_s})} = \ln{(10)} V_T \log_{10}{(\frac{i}{I_s})} \tag{10.3}$$

These expressions turns out to be quite reliable for reasonable diodes to reasonable voltages.

k is Boltzmann's constant, T is temperature in Kelvins, q is the charge of an electron.
I_s is the scale current which is usually $\approx 1pA$, which doesn't change much until the breakdown point.

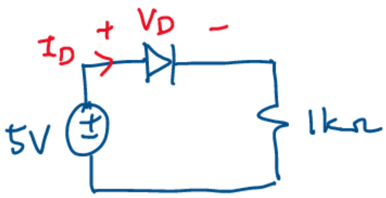
Using the ideal diode equation we can find the relationship between voltages and currents as they pass through the diode.

$$\frac{i_2}{i_1} = \frac{I_s e^{\frac{V_2}{V_T}}}{I_s e^{\frac{V_1}{V_T}}} = e^{\frac{V_2 - V_1}{V_T}} \tag{10.4}$$

$$V_2 - V_1 = V_T \ln{(\frac{i_2}{i_1})} \xrightarrow{\text{room temperature}} 60mV \log_{10} \frac{i_2}{i_1} \tag{10.5}$$

Example:

Calculate the diode voltage and current in the circuit below.
Assume that the diode voltage is 0.7V at 1mA and V_T = 25mV.



Example

Recall (10.1). Plugging in the given values gives us the scale current.

$$1mA = I_s e^{\frac{0.7V}{25mV}}, I_s = 6.9 \cdot 10^{-16} A \Rightarrow I_o = I_s e^{v_o/v_T} \tag{10.6}$$

Ohm's law can then be applied at the resistor

$$V_r = IR = I_o R = 5V - V_D \Rightarrow 5 - V_D = I_o R \tag{10.7}$$

V_D is the voltage across the diode

So we have two equations and two unknowns (since we know $v_T = 25mV$ but v_o was used at first just to find I_s) Solving for the unknowns gives us:

- $V_o = 0.736V$
- $I_D = 4.264mA$

Section 11

Lecture 4: Forward conducting diodes

The exponential model accurately describes the diode outside of the breakdown region, though it's nonlinear behaviour makes it difficult to use.

For $V_{DD} > 0.5V$

V_{DD} denotes a DC power supply

$$I_D = I_S e^{V_D/V_T} \tag{11.1}$$

Where

- I_S is the diode parameter
- V_T is the thermal voltage

Another equation may be produced via Kirchhoff's law

$$I_D = \frac{V_{DD} - V_D}{R} \tag{11.2}$$

The unknown quantities I_D and V_D may be solved for via graphical analysis or iteration.

Example

This simple circuit is used to demonstrate the exponential model of the diode.

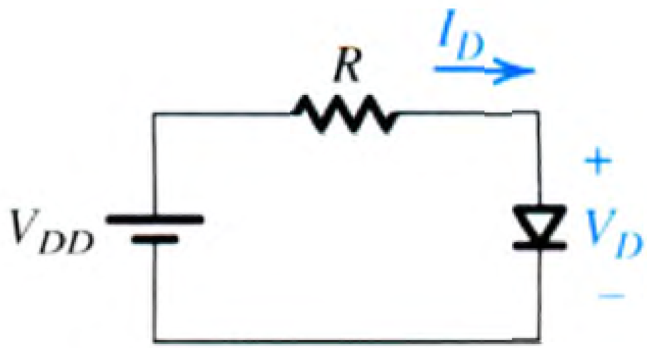
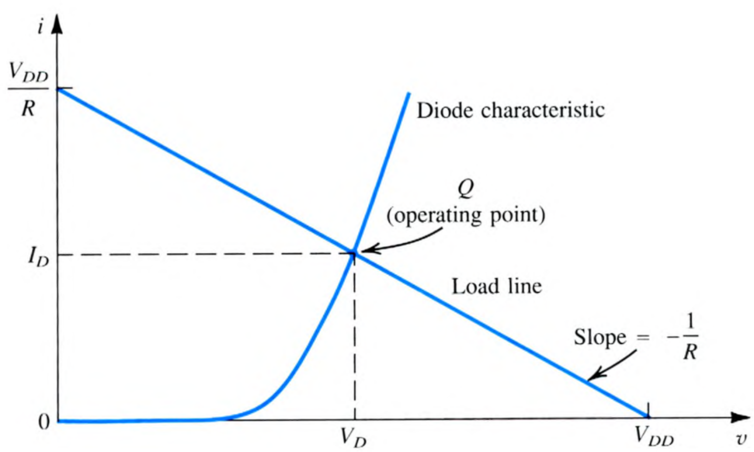


Figure 7. Simple example circuit with diode

Plots of the diode characteristics and Kirchhoff's relation are plotted, the intersection of which gives the solution.



An iterative procedure may also be applied to solve for the unknowns, the procedure for which will be illustrated through an example

Example Find I_D, V_D for the circuit in the previous example (Fig. 7). $V_{DD} = 5V, R = 1k\Omega$, and at $V_D 0.7V, I_D = 1mA$

1. Assume $V_D = 0.7V$, then use (11.2) to find I_D.

$$I_D = \frac{5V - 0.7V}{1k\Omega} = 4.3mA \tag{11.3}$$

2. Use the diode equation (11.1) to get a better estimate for V_D.

$$V_2 - V_1 = 2.3V_T \log \frac{I_2}{I_1} \Rightarrow V_2 = V_1 + 0.06 \log \frac{I_2}{I_1} \tag{11.4}$$

substituting $V_1 = 0.7V, I_1 = 1mA, I_2 = 4.3mA$,

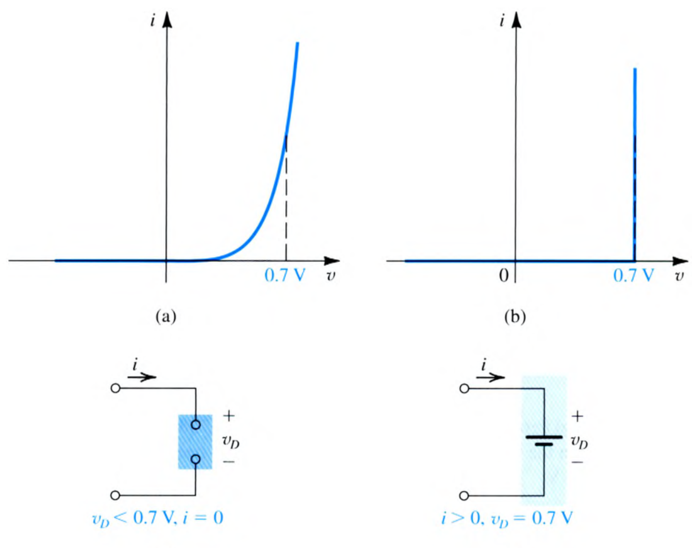
$$V_2 = 0.738V \Rightarrow I_D = 4.3mA, V_D = 0.738V \tag{11.5}$$

- This states that for a decade[7] change in current the diode voltage drop changes by $2.3(V_T \approx 60mV)$ which is negligibly small for $v < 0.5V$. The voltage at which the this behaviour becomes significant is called the **cut-in voltage**

3. Repeat steps 1 and 2 with the new values until the values more or less become stable

[7] *Factor of 10*

This iterative model is powerful and yields accurate results, but can be computationally expensive especially when calculating by hand. To address this we employ other models such as the *constant-voltage-drop* model which approximates the exponential characteristics via a piecewise linear model. The reason why this is possible is because forward conducting diodes exhibit a voltage drop that varies in a relatively narrow range.



Using the constant voltage drop model in our analysis looks the same as before, but with V_D directly taking on the value of $0.7V$ (as per the prior example) instead of being solved for with the diode equation.

In applications that involve voltages greater than the voltage drop (i.e. usually $\approx 0.6 - 0.8V$) we can neglect the diode voltage drop altogether while calculating the diode current.

$$\begin{aligned} V_D &= 0V \\ I_D &= \frac{5-0}{1} = 5mA \end{aligned} \tag{11.6}$$

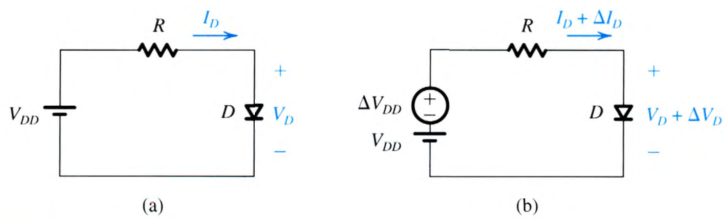
This is generally good enough for a first estimate, though the previous model isn't that much more work and gives more accurate results. The primary use of this model is to determine which diodes are on or off in a multi-diode circuit

11.0.1 Small-Signal Model

The small signal method is an alternative model used to describe the nonlinear diode's characteristics with greater accuracy than piecewise linear models.

Consider a small ΔV_{DD} applied to the diode, which would cause a small $\Delta I_D, \Delta V_D$. We want to find a quick way of determining the values of these incremental changes.

Similar methods will be applied to transistors in later chapters



Skipping a bunch of math[8] the results are as follows:

[8] It is 11:17pm and I have two more lectures to catch up to today

Definition 19

Small signal approximation

$$i_D(t) \approx I_D(1 + \frac{v_d}{V_T}) \quad (11.7)$$

This is valid for when variations in diode voltage $|v_d| \lessapprox 5mV$
The quantity relating i_d to v_d is the diode small-signal resistance

$$r_d = \frac{V_T}{I_D} \quad (11.8)$$

The steps for calculating the small signal model are as follows:

Small signal analysis can be performed separately from the dc bias analysis because of the linearization of diode characteristics in the small-signal approximation

1. Perform a dc analysis using the exponential, constant-voltage-drop, or piecewise-linear model.
2. Linearise the circuit. For a forward-based diode, find r_d by substitution I_D into (11.8). The small-signal equivalent circuit is found by eliminated all independent dc sources[9] and replacing the diode with its small-signal resistance r_d
3. Solve the linearised circuit. In particular we would want to find $\Delta I_D, \Delta V_D$ and check to see if it is consistent with our approximation, i.e. that $\Delta V_D \lessapprox 5mV$

[9] since we already accounted for them in step 1

The reason why we linearise these non-linear systems, we, as engineers, try to linearise them because it is convenient to be able to use superposition, phasors, Fourier series, Laplace transforms, and so forth.

PART

V

ECE358: Foundations of Computing

Taught by Prof. Shurui Zhou

Section 12

Admin and Preliminary

Subsection 12.1

Lecture 1

Topics covered will include:

- Graphs, trees
- Bunch of sorts

- Fancy search trees; red-black, splay, etc
- DP, Greedy
- Min span tree, single source shortest paths
- Maximum flow
- NP Completeness, theory of computation
- Blockchains??
- Θ

Solutions will be posted on the window of SF2001. Walk there and take a picture.

12.1.1 Mark Breakdown

Table 4. Mark Breakdown

Homework x 5	25
Midterm (Open book)	35
Final (Open book)	40

Section 13

Complexities

Subsection 13.1

Lecture 2

This lecture we talked about big O notation. For notes on this refer to my tutorial notes for ESC180, ESC190: https://github.com/ihasdapie/teaching/

Definition 20 Big O notation (upper bound)
$g(n)$ is an asymptotic upper bound for $f(n)$ if:

$$O(g(n)) = \{f(n) : \exists \quad c, n_0 \quad s.t. \quad 0 \leq f(n) \leq c \cdot g(n), \forall n \geq n_o\} \tag{13.1}$$

Proof **What is the big-O of $n!$?**

$$n! \leq n \cdot n \cdot n \cdot n \ldots n = n^n \Rightarrow n! \in O(n^n) \tag{13.2}$$

□

Definition 21 Big Ω notation (lower bound)
$h(n)$ is an asymptotic lower bound for $f(n)$ if:

$$\Omega(h(n)) = \{f(n) : \exists \quad c, n_0 > 0 \quad s.t. \quad 0 \leq c \cdot h(n) \leq f(n), \forall n \geq n_o\} \tag{13.3}$$

Proof **Find Θ for $f(n) \sum_i^n i$.**
For this we will employ a technique for the proof where we take the right half of the function, i.e. from $\frac{n}{2} \ldots n$ and then find the bound

$$
\begin{aligned}
f(n) &= 1+2+3\ldots+n \\
&\geq \lceil\frac{n}{2}\rceil + (\left\lceil\frac{n}{2}\right\rceil+1) + (\left\lceil\frac{n}{2}\right\rceil+2)+\ldots n \quad n/2 \text{ times} \\
&\geq \left\lceil\frac{n}{2}\right\rceil + \left\lceil\frac{n}{2}\right\rceil + \left\lceil\frac{n}{2}\right\rceil + \ldots \left\lceil\frac{n}{2}\right\rceil \\
&\geq \frac{n}{2}\cdot\frac{n}{2} \\
&= \frac{n^2}{4}
\end{aligned}
\tag{13.4}
$$

And therefore for $c = \frac{1}{4}$ and $n = 1$, $f(n) \in \Theta(n^2)$

□

Definition 22 Big Θ notation (asymptotically tight bound)

$$\Theta(g(n)) = \{f(n) : \exists \quad c_1 c_2, n_0 \quad s.t. \quad 0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n), \forall n \geq n_o\} \tag{13.5}$$

PROOF Prove that

$$\sum_{j=1}^{n} i^k = \Theta(n^{k+1}) \tag{13.6}$$

First, prove $O(f(n)) = O(n^{k+1})$

$$
\begin{aligned}
f(n) = \sum_{j=1}^{n} i^k &= 1^k + 2^k + \ldots n^k \\
&\leq n^k + n^k + \ldots n^k \\
&= n^{k+1}
\end{aligned}
\tag{13.7}
$$

Next, prove $\Omega(f(n)) = \Omega(n^{k+1})$

$$
\begin{aligned}
f(n) = \sum_{j=1}^{n} i^k &= 1^k + 2^k + \ldots n^k \\
&= n^k + (n_1)^k + \ldots 2^k + 1^k = \sum_{i=1}^{n}(n-i+1)^k \\
&\geq \frac{n}{2}^k * n \geq \frac{n^{k+1}}{2^k} = \Omega(n^{k+1})
\end{aligned}
\tag{13.8}
$$

Therefore $f(n) = \Theta(n^{k+1})$ □

Note that we may not always find both a tight upper and lower bound so not all functions have a tight asymptotic bound.

Theorem 7 **Properties of asymptotes:**
Note: $\wedge$ means AND
Transitivity [10]

$$(f(n) = \Theta(g(n)) \wedge g(n) = \Theta(h(n))) \Rightarrow f(n) = \Theta(h(n)) \tag{13.9}$$

[10] *The following applies to* O, Θ, o, ω

Reflexivity[11]

$$f(n) = \Theta(f(n)) \tag{13.10}$$

Symmetry

$$f(n) = \Theta(g(n)) \iff g(n) = \Theta(f(n)) \tag{13.11}$$

Transpose Symmetry

$$\begin{aligned} f(n) = O(g(n)) &\iff g(n) = \Omega(f(n)) \\ f(n) = o(g(n)) &\iff g(n) = \omega(f(n)) \end{aligned} \tag{13.12}$$

[11] *The following applies to* O, Θ

Runtime complexity bounds can sometimes be used to compare functions. For example, $f(n) = O(g(n))$ is like $a \leq b$

- $O \approx \leq$
- $\Omega \approx \geq$
- $\Theta \approx \approx$
- $o \approx <$; an upper bound that is **not** asymptotically tight
- $\omega >$ a lower bound that is **not** asymptotically tight

Note that there is no trichotomy; unlike real numbers where we can just do $a < b$, etc, we may not always be able to compare functions.

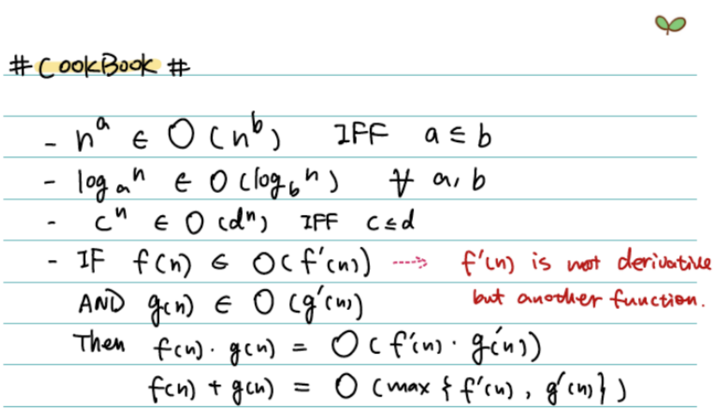


Figure 8. Complexity Cookbook

Subsection 13.2

Lecture 3: Logs & Sums

13.2.1 Functional Iteration

$f^{(i)}(n)$ denotes a function iteratively applied i times to value n.

For example, a function may be defined as:

$$f^{(i)}(n) = \begin{cases} f(n) & \text{if } i = 0 \\ f(f^{(i-1)}(n)) & \text{if } i > 0 \end{cases} \tag{13.14}$$

Recall:

$$a = b^c \Leftrightarrow log_b a = c \tag{13.13}$$

Given (13.14) we see that

1. $f(n) = 2n$
2. $f^{(2)}(n) = f(2n) = 2^2 n$
3. $f^{(3}(n) = f(f^{(2)}(n)) = 2^3 n \vdots$
4. $f^{(i)}(n) = 2^i n$

As an exercise we may look at an iterated logarithm function, 'log star'

$$lg^*(n) = \min\{i \geq 0 : lg^{(i)} n \leq 1\} \tag{13.15}$$

This describes the number of times we can iterate $log(n)$ until it gets to 1 or smaller.

- $log^* 2 = 1$
- $log^* 4 = 2 = log^* 2^2 = 1 + log^* 2 = 2 \vdots$
- for practical reasons log^* doesn't really get bigger than 5. This is one of the slowest growing functions around.

Summations & Series

PROOF Proof for a finite geometric sum:

$$\begin{aligned} \sum_{k=0}^{n} x^k &= S \\ S &= 1 + x + x^2 \dots x^n \\ xS &= x + x^2 + x^3 \dots x^{n+1} \\ S &= \frac{1 - x^{n+1}}{1 - x} \end{aligned} \tag{13.16}$$

□

$$\sum_{i=1}^{\infty} x^i = \frac{1}{1-x} \quad \text{if } |x| < 1 \tag{13.17}$$

$$\sum_{k=0}^{\infty} kx^k = \frac{x}{(1-x)^2} \quad \text{if } |x| < 1 \tag{13.18}$$

PROOF Begin by differentiating both sides over x

$$\sum_{k=0}^{\infty} x^k = \frac{1}{(1-x)} \quad \text{if } |x| < 1 \tag{13.19}$$

$$\sum_{k=0}^{\infty} kx^{k-1} = \frac{1}{(1-x)^2} \tag{13.20}$$

And then multiply both sides by x, therefore (13.18) follows. □

Telescoping Series

$$\sum_{k=1}^{n} a_k - a_{k-1} = a_n - a_0 \tag{13.21}$$

Proof Write it out and cancel out terms

$$(a_1 - a_0) + (a_2 - a_1) \dots (a_n - a_{n-1}) = a_n - a_0 \tag{13.22}$$

Therefore the sum telescopes □

Another telescoping series may be proved similarly:

$$\sum_{k=1}^{n-1} \frac{1}{k(k+1)} \xrightarrow{math} \sum_{k=1}^{n-1} (\frac{1}{k} - \frac{1}{k+1}) = (1 - \frac{1}{2}) + \dots (\frac{1}{n-1} - \frac{1}{n}) = a_o - a_n \tag{13.23}$$

Subsection 13.3

Lecture 4: Induction & Contradiction

13.3.1 Induction

The general steps for proving a statement by induction are:

1. Basis
2. Hypothesis
3. Inductive step

I.e. if the basis holds for some i, i.e. $0, 1, 2, 3, 12, \dots$ AND if we assume that the hypothesis holds for an arbitrary number k, then we just need to prove that the inductive step follows, or that $P(n+1)$ holds.

Example Prove that $P(n) = 1 + 2 + 3 + \dots n = \frac{n(n+1)}{2}$

Proof

1. Basis: $P(0) = 0 = \frac{0(0+1)}{2}$
2. Hypothesis (assume that it is true): $P(k) = \frac{k(k+1)}{2}$
3. Inductive step (need to prove $P(k+1) = \frac{(n+1)(n+2)}{2}$): $P(k+1) = \underbrace{1 + 2 + \dots + n}_{\frac{n(n+1)}{2}} + (n+1) = \dots = \frac{n^2+3n+2}{2} = \frac{(n+1)(n+2)}{2}$

□

Example Show that for any finite set S, the power set 2^S has $2^{|S|}$ elements (that is, there are $2^{|S|}$ distinct subsets of S)

The power set of a set S is the set of all subsets of S

PROOF

1. Basis:
$$n = 0, |S| = 0, |2^S| = 1 = 2^0 \quad (13.24)$$
$$n = 1, |S| = 1, |2^S| = 2 = 2^1 \quad (13.25)$$

2. Hypothesis: Assume that 2^S has 2^n elements when $|S| = n$

3. Inductive step: need to prove that when $|S| = n+1, |2^S| = 2^{n+1}$

   Let $B = S \not\{a\}$ for some $a \in S$. Now there are two types of subsets of S; those that include a and those who do not include a

   For subsets that do *not* include a , $|2^B| = 2^{|B|} = 2^n$, by the hypothesis.

   For subsets that do include a, these sets are of size $2^B \cup \{a\}, which is 2^n$.

   Therefore the total number of subsets of S is $2^n + 2^n = 2^{n+1}$, as desired.

□

The same kind of argument can be applied to problems such as the Towers of Hanoi and the tiling problem.

13.3.2 Contradiction

1. Assume the theorem is false
2. Show that the assumption is false (leads to a contradiction)
   - Therefore the theorem is true

Example Prove that $\sqrt{2}$ is irrational

PROOF Assume that $\sqrt{2}$ is rational.
Therefore we can write $\sqrt{2}$ as
$$\sqrt{2} = \frac{a}{b} \quad (13.26)$$
Where a, b **have no common factors**.
We can square both sides
$$2 = \frac{a^2}{b^2} \to a^2 = 2b^2 \quad (13.27)$$
Therefore a^2 is even.
Let $a = 2c$
$$2^2c^2 = 2b^2 \to b^2 = 2c^2 \quad (13.28)$$
Therefore b is even as well.
This results in a contradiction since we assumed that a, b have no common factors, but our analysis shows that both would have to be even (and share a common factor of 2). □

PART VI

MAT389: Complex Analysis

Taught by Prof. Sigil

SECTION 14

Complex Numbers

Subsection 14.1

Lecture 1

Consider a 2-vector $\vec{x} = (x, y) \in \mathcal{R}$. As complex numbers correspond to two-vectors

$$\vec{x} = (x, y) \leftrightarrow z = x + iy, i^2 = -1 \tag{14.1}$$

z is, therefore, a complex variable. What are the benefits of a complex number like z?

This prof lectures at the speed of sound and talks *into* the board. Couldn't quite follow during this lecture, hopefully I get better about it in the following ones.

Definition 23

Imaginary and Complex Numbers
i is an imaginary number such that

$$i^2 = -1 \tag{14.2}$$

A complex number has the form:

$$z = x + iy \tag{14.3}$$

Definition 24

There are a number of operations we can perform on complex numbers.
Addition

$$z + z' = (x + x') + i(y + y') \tag{14.4}$$

Multiplication

$$zz' = (x + iy)(x' + iy') = (xx' - yy') + i(xy' + x'y) \tag{14.5}$$

Proof

Proof of (14.5):

$$\begin{aligned} zz' &= (x + iy)(x' + iy') \\ &= x + ixy' + iyx' + i^2yy' \\ &= xx' - yy' + i(xy' + yx') \end{aligned} \tag{14.6}$$

□

Magnitude

$$|z| = \sqrt{x^2 + y^2} \tag{14.7}$$

Conjugate
The complex conjugate has the properties:

- $\overline{z}z = |z|^2$
- $\overline{(z + z')} = \overline{z} + \overline{z}'$
- $\overline{z \cdot z'} = \overline{z} \cdot \overline{z}'$

We can define a new operation

$$\forall \text{complex} z, \exists \quad \text{complementary number } w \text{ such that } zw = wz = 1 \tag{14.8}$$

Denote

$$w = \frac{1}{z} = z^{-1} \tag{14.9}$$

Proof

Proof of (14.9): Find w s.t. $zw = 1$

$$\begin{aligned} zw &= 1 \\ w\overline{z}z &= \overline{z}z = |z|^2 > 0 \\ |z|^2 w &= \overline{z} \\ w &= \frac{\overline{z}}{|z|^2} \to Z^{-1} = \frac{\overline{z}}{|z|^2} \end{aligned} \tag{14.10}$$

□

Furthermore, there are operators that we can define on complex numbers.

Definition 25

Real and Imaginary Operators

Given $z = x + iy$, we can define the real and imaginary operators

$$x = Re\left\{z\right\} \tag{14.11}$$

$$y = Im\left\{z\right\} \tag{14.12}$$

Example

$$Im\left\{(1 = \sqrt{2}i)^{-}1\right\} \tag{14.13}$$

By (14.9), we have

$$Im\left\{z^{-1}\right\} = \frac{-Im\left\{z\right\}}{|z|^2} \tag{14.14}$$

And

$$Re\left\{z^{-1}\right\} = \frac{-Re\left\{z\right\}}{|z|^2} \tag{14.15}$$

Using these, for example, we find that the $Im = \frac{-\sqrt{2}}{3}$
We can get the real component in a similar way.

Here is an enumeration of absolute value properties for complex numbers:

$$|z \cdot w| = |z||w| \tag{14.16}$$

$$|z + w| \leq |z| + |w| \tag{14.17}$$

$$|\overline{z}| = |z| \tag{14.18}$$

$$|z + w|^2 = (\overline{x} + \overline{w})(z + w) = |z|^2 + |w|^2 + \overline{z}w + \overline{w}z \tag{14.19}$$

Proof

Note that $\overline{z}w + \overline{w}z = 2Re\left\{z\overline{w}\right\}$, by (14.19)
And so

$$|z + w|^2 \leq |z|^2 + |w|^2 + 2|z||w| = (|z| + |w|)^2 \tag{14.20}$$

□

Subsection 14.2
Lecture 2

Whereas a two-vector $\vec{x} \in \mathbb{Z}$, complex numbers exist in the complex plane, $z \in \mathbb{C}$

Theorem 8 **Polar Decomposition**
Complex numbers can be expressed in polar form as well

$$z = r(cos\theta + isin\theta) \tag{14.21}$$

Where

$$r = |z| \qquad x = rcos\theta \qquad y = rsin\theta \tag{14.22}$$

This has a number of useful properties

$$z \cdot z' = |z||z'|(\cos(\theta + \theta') + i\sin(\theta + \theta')) \tag{14.23}$$

$$\frac{z}{z'} = \frac{|z|}{|z'|}(\cos(\theta - \theta') + i\sin(\theta - \theta')) \tag{14.24}$$

Proof. Proof for (14.23):

$$\begin{aligned} z \cdot z' &= |z|(\cos(\theta + i\sin\theta)) \times |z'|(\cos\theta' + i\sin\theta') \\ &= |z||z'|(\cos\theta\cos\theta' + i\cos\theta\sin\theta' + i\sin\theta\cos\theta' - \sin\theta\sin\theta') \\ &= |z||z'|\left[\cos\theta\cos\theta' - \sin\theta\sin\theta' + i(\cos\theta\sin\theta' + \sin\theta\cos\theta')\right] \end{aligned} \tag{14.25}$$

And the proof follows □

Lemma 2 A corollary exists

$$z^2 = |z|^2(\cos 2\theta + i\sin 2\theta) \tag{14.26}$$

Theorem 9 **Moivre's Theorem**

$$z^n = |z|^n(\cos(n\theta) + i\sin(n\theta)) \tag{14.27}$$

So we may define z to be the n^{th} root of w which implies that

Lemma 3 Every complex number has a n^{th} root $\forall n$

Proof.

$$\text{Let } z = |w|^{\frac{1}{n}}(\cos\frac{\theta}{n} + i\sin\frac{\theta}{n}) \tag{14.28}$$

Then

$$w = |w|(\cos\theta + i\sin\theta), \text{ then } z^n = w \tag{14.29}$$

□

Intuition: define z to be $\frac{1}{n}$ and then take the n^{th} power of both sides to show that $z^n = w$

This leads us to the conclusion that representations of complex numbers are not unique[12].

[12] *They are part of a cyclic group*

Proof. If every z can be written as $z = r(\cos\theta + i\sin\theta)$, then it holds for $\theta + 2\pi n \forall n \in \mathbb{Z}$ since $\sin\theta = \sin(\theta + 2\pi n)$ and $\cos\theta = \cos(\theta + 2\pi n)$. □

14.2.1 Functions on complex planes

Definition 26 Given a domain $\mathbb{D} \in \mathbb{C}$, a function f is a rule such that

$$z \in \mathbb{D} \xrightarrow{f} w \in \mathbb{D} \leftrightarrow w = f(z) \tag{14.30}$$

Definition 27 We may define $\mathbb{D}$ to be the domain of f
Likewise, range is defined as

$$Ran\{f\} = \{w \in \mathbb{C} : \exists z \in D : f(z) = w\} \tag{14.31}$$

Example

$$f(z) = \frac{1}{z+i} \tag{14.32}$$

What is the maximum domain of f?

$$Dom\{f\} = \{z \in \mathbb{C} : |z| < -i\} \tag{14.33}$$

What is the range of f?

$$\frac{1}{z+i} = w \tag{14.34}$$

For which values of w can we solve this equation?

$$z = -i + \frac{1}{w} \tag{14.35}$$

So the range of the function is

$$Ran\{f\} = \{w \in \mathbb{C} : |w| \neq 0\} \tag{14.36}$$

Example

$$f(z) = z^2 + 1 \tag{14.37}$$

It is fairly clear that $Dom\{f\} = f \in \mathbb{C}$
The range can be found by solving for z in

$$z^2 + 1 = w \tag{14.38}$$

And so

$$Ran\{f\} = \{w \in \mathbb{C}\} \tag{14.39}$$

14.2.2 Exponential Functions

Definition 28 Given $z = x + iy$,

$$e^z = e^x(\cos y + i\sin y) = e^{Re\{z\}}(cos(Im\{z\}) + i\sin(Im\{z\})) \quad (14.40)$$

1. $e^{z+w} = e^z e^w$
2. $|e^z| = e^{Re\{z\}} \neq 0$
3. $e^{z+i2\pi n} = e^z$

Proof (1) follows from the product rule for complex numbers

(2) follows by definition

(3) follows by definition (recall: writing z w.r.t. sin, cos) □

More properties:

arg, or argument is the angle from the real axis to that on the complex plane. Usually denoted by θ

- $Dom\{e^z\} = \mathbb{C}$
- $Ran\{e^z\} = \{\mathbb{C} \setminus \{0\}\}$
- $e^z = w \qquad \text{if } w \neq 0$

[13]

[13] Note: ‘ \ ’ denotes set exclusion

$$\begin{aligned} z &= ln|w| + i\arg w \\ e^z &= e^{ln|w| + i\arg w} \\ &= e^{ln|w|}e^{i\arg w} \\ &= |w|\cos(\arg w) + i\sin(\arg w) \\ &= w \end{aligned} \quad (14.41)$$

Remark **Polar representation**

$$w = |w|e^{i\arg w} \quad (14.42)$$

Example **Find polar coordinates of $z = i + 1$**

$$r = |w| \quad \theta = \arg w \quad (14.43)$$

$$\begin{aligned} |z| &= \sqrt{1+i} = \sqrt{2} \\ \cos\theta &= \frac{1}{\sqrt{2}} \to \theta = \frac{\pi}{4} \\ z &= \sqrt{2}e^{i\pi/4} \end{aligned} \quad (14.44)$$

Example Find

$$(1+i)^{\frac{1}{3}} \quad (14.45)$$

Solution: $z = \sqrt{2}e^{\frac{i\pi}{4}} \to z^{1/3} = 2^{\frac{1}{6}}e^{i\pi/12}$

Definition 29 **Trig functions for complex numbers**

$$\cos z = \frac{1}{2}\left(e^{iz} + e^{-iz}\right) \quad (14.46)$$

Proof

$$\cos x = \frac{1}{2}(e^{ix} + e^{-ix}) = \frac{1}{2}\left(\cos x + i\sin x + \underbrace{\cos(-x)}_{\text{odd; }=\cos(x)} + \underbrace{i\sin(-x)}_{\text{even; }=-\sin(x)}\right) = \cos x \tag{14.47}$$

□

$$\sin z = \frac{1}{2}\left(e^{iz} - e^{-iz}\right) \tag{14.48}$$

And a similar proof follows for $\sin z$.
These have the following properties

$$\sin z|_{ImZ=0} = \sin x \tag{14.49}$$

$$\cos(z + 2\pi n) = \cos z \forall n \in \mathbb{Z} \tag{14.50}$$

$$\sin(z + 2\pi n) = \sin z \forall n \in \mathbb{Z} \tag{14.51}$$

Proof

$$\begin{aligned} \cos z + 2\pi n &= \frac{1}{2}(e^{i(z+2\pi n)} + e^{-i(z+2\pi n)}) \\ &= \frac{1}{2}(e^{iz}e^{i2\pi n} + e^{-iz}e^{-i2\pi n}) \\ &= \frac{1}{2}(e^{iz} + e^{-iz}) \\ &= \cos z \end{aligned} \tag{14.52}$$

□

The domain of $\{\cos z, \sin z\} = \mathbb{C}$
Range?
Solve $\cos z = w$ for z

$$\begin{aligned} \frac{1}{2}(e^{iz} + e^{-iz}) &= w \\ \ldots \times 2e^{iz} &\text{ on both sides} \\ e^{2iz} - 2we^{iz} + 1 &= 0 \\ &\ldots\text{Let } S = e^{iz} \\ S^2 - 2ws + 1 &= 0 \\ S &= w \pm \sqrt{w^2 - 1} \equiv u \end{aligned} \tag{14.53}$$

Now we note that $e^{iz} = u$ can be solved for z for any $u \neq 0$

$$u = 0 \leftrightarrow w = \pm\sqrt{w^2 - 1} \tag{14.54}$$

$$w^2 = w^2 - 1 \text{ impossible for } u \neq 0 \tag{14.55}$$

Therefore:

$$Ran\{\cos z\} = Ran\{\sin z\} = \mathbb{C} \tag{14.56}$$

Remark An intuitive way of interpreting this result is thinking of $\{\sin, \cos\}$ being a function that projects values from the complex domain to the real plane; though $\{\sin, \cos\}$ takes on a limited range of values in the real domain, in the complex domain it spans the entire plane. Think: mental model of a complex number spinning around and having that project onto a real line. More formally, see: the Little Picard Theorem

PART

VII

ECE444: Software Engineering

Taught by Prof. Shurui Zhou

Section 15

Preliminary

Subsection 15.1

Lecture 1, 2

- Software engineering is different from what coding is; design, architecture, documentation, testing, etc v.s. just script kiddie-ing
- Vasa syndrome
- Rockstar engineers are a myth

Section 16

Project Management

Subsection 16.1

Lecture 3

Definition 30 Conway's law states that 'Any organization that designs a system (defined broadly) will produce a design whose structure is a copy of the organization's communication structure'.

The waterfall method is slow and costly and defects can be extremely costly, especially early on in the development lifecycle.

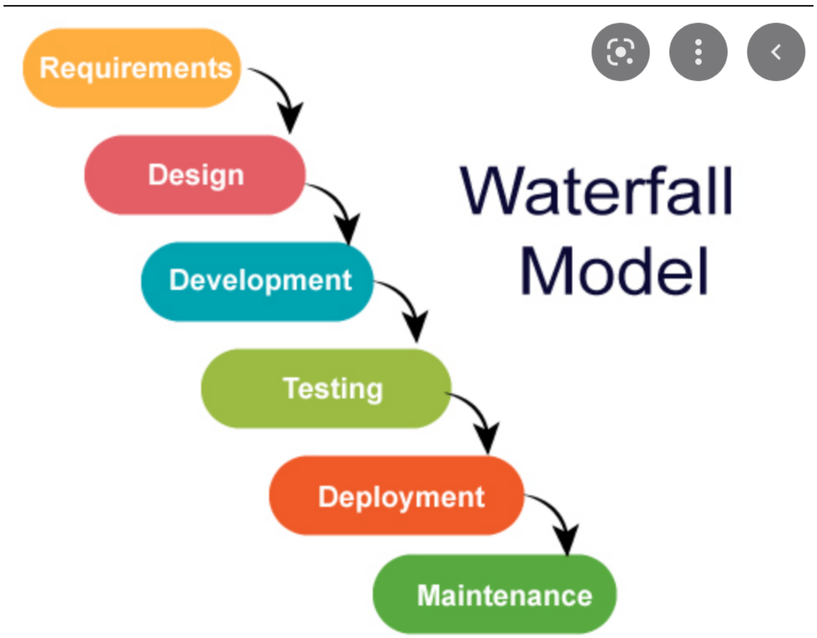


Figure 9. Waterfall method

In order to address this the V model was introduced which increases the amount of testing to reduce the possibility of having to rework everything

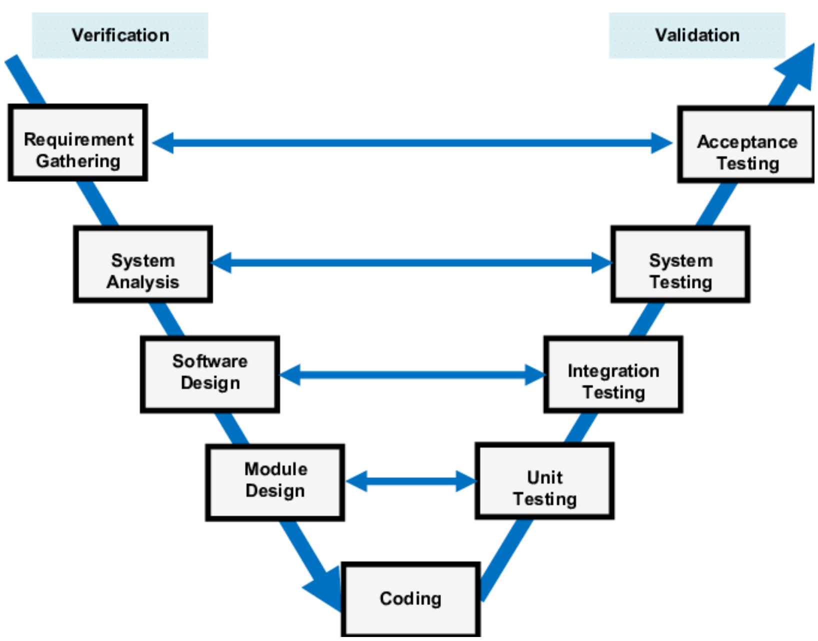


Figure 10. V model

Generally speaking the waterfall model isn't used much anymore due to the reality that software specifications change on a near daily basis.

Recall: aUToronto Spring 2022 integration hell

16.1.1 Agile

Agile is a project management approach which, in most general terms, seeks to respond to change and unpredictability using incremental, iterative work (sprints). This allows for a balance between the need for predictability and the need for flexibility. Some agile methods include:

- Extreme programming: really really fast iteration (think days)
- Scrum: 2-4 week sprints with standups and backlogs; sticky notes for tasks, etc. Think kanban boards. Daily scrum meetings to unblock ASAP. Development lifecycle is therefore a series of sprints.
- On-site customer; frequent interaction with end users to figure out what exactly they need.

Subsection 16.2

I dropped this course

I decided to drop this course because the courseload was a little too much to handle between EngSci ECE, clubs, design teams, work, and trying to have a life.

PART

VIII

ESC301: Seminar

Section 17

Preliminary

Subsection 17.1

Seminar 1