EE16B Course Notes

Anmol Parande

Spring 2019 - Professors Anant Sahai, Kris Pister, and Jaijeet Roychowdhury

Contents

1 Digital Circuits

Definition 1 *A digital circuit interprets a continuous signal (like voltage) as a discrete quantity (such as logic "1" and logic "0")*

The opposite of a digital circuit is an analog circuit.

Definition 2 *An analog circuit operates on continuous signals*

In a digital circuit, small changes to inputs don't lead to large changes in output. This makes them more robust to noise.

1.1 Transistors

Definition 3 *A transistor is an electronic device that, in it most simple form, acts like a voltage controlled switch*

Definition 4 *CMOS (Complimentary metal oxide semiconductor) transistor circuits are made of PMOS and NMOS transistors to make digitial circuits.*

CMOS circuits are useful because they use less power than if each transistor type was used alone because current only flows when either the PMOS or NMOS is on, but not when both are off.
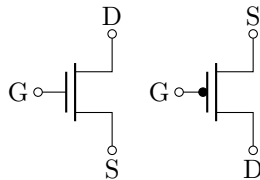
Figure 1: A NMOS transistor (left) and a PMOS transistor (right)

For both PMOS and NMOS transistors, they have a gate voltage (V_G), a Drain voltage (V_{DD}), and a Source voltage (V_{SS}). They also have a threshold voltage V_{th} which determines when the transistor opens or closed.

An NMOS transistor will close when $V_{GS} \geq V_{th}$. The Drain is defined as the terminal which has the higher voltage. The Source is the other terminal.

The PMOS transistor is the opposite. It closes when $|V_{GS}| \geq |V_{th}|$. In contrast to the NMOS, the Source of a PMOS is the terminal with the higher

voltage, and the drain is the other one. This allows us to most simply model the transistors as switches.
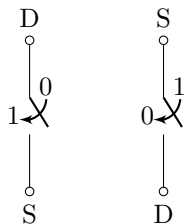
Figure 2: A NMOS Switch Model (left), PMOS Switch Model (right)

Here, the 0 and 1 represent logical 0 and logical 1. In this case, $V_G > V_{th}$ is logical 1 and $V_G < V_{th}$ is logical 0. Logical 1 can be thought of as "high voltage" whereas logical 0 can be thought of as "low voltage".

This simple model is incomplete, however, because transistors dissipate energy (i.e they have resistance). A slightly more accurate model will include resistors.
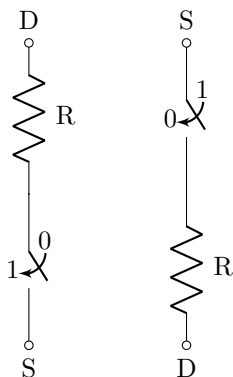
Figure 3: A NMOS Resistor-Switch Model (left), PMOS Resistor-Switch Model (right)

There is another caveat to transistors that even our resistor-switch model doesn't account for, and that is the time it takes for the transistor to actually close. To account for this caveat, we can introduce a capacitor into our model.
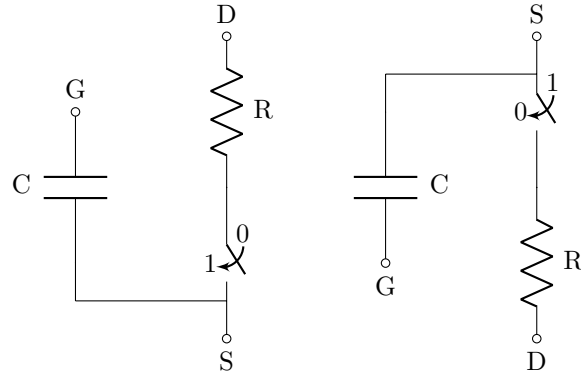
Figure 4: A NMOS Model (left), PMOS Model (right)

1.2 Logic Circuits

Now that we have a good model for transistors, we can start to use them to build circuits which perform logical operations. The simplest logic gate is an inverter

Definition 5 *The inverter ($Y = \bar{A}$) returns the opposite of the input value.*

A	Y
1	0
0	1
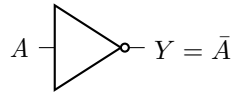
An inverter is made by putting a PMOS and NMOS transistor in series.
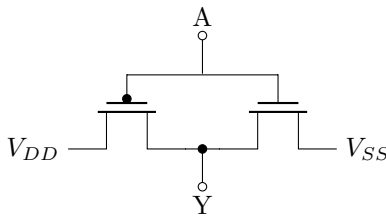
Figure 5: CMOS Inverter

If $A = 1$, then the PMOS will be open and the NMOS will be closed, so $Y = V_{SS}$ which is logical 0. Likewise, if $A = 0$, then the PMOS will be closed and the NMOS will be open, so $Y = V_{DD}$ which is logical 1.

Definition 6 *The NOR gate (Y = $\overline{A + B}$) returns the opposite of the or oper-ation.*

A	B	Y
0	0	1
1	0	0
0	1	0
1	1	0

$$A$$
$$B$$
$$Y = \overline{A + B}$$
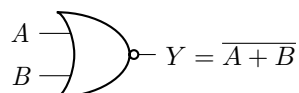
With transistors, the NOR gate looks as follows.
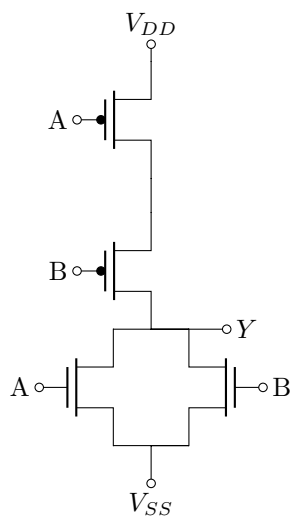
Figure 6: CMOS NOR

To verify this works, try and see how plugging in different combinations for A and B change the output.

Definition 7 *The NAND gate (Y = $\overline{A \cdot B}$) returns the opposite of the AND operation.*

A	B	Y
0	0	1
1	0	1
0	1	1
1	1	0

$$Y = \overline{A \cdot B}$$
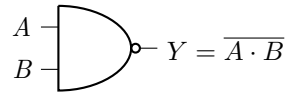
With transistors, this looks like:
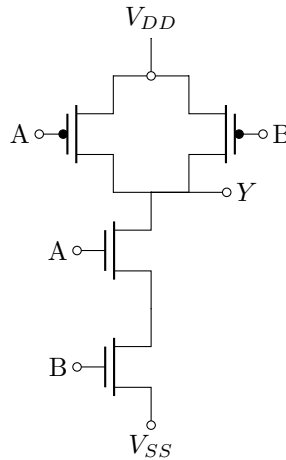
Figure 7: CMOS NOR

In general, CMOS circuits can be thought of as a "pull down" networks consisting of PMOS transitors and "push up" networks consisting of NMOS transistors. This is because the PMOS transistors serve to "pull" V_{DD} down the output where as the NMOS transistors "push" V_{SS} up to the output.

An interesting fact is that NAND, NOR, and NOT gates are universal. This means any logical operator can be constructed out of them.

2 Transient Analysis

Most often in circuits, the behavior of the circuit is varying with time. This happens when the IV relationships of elements have a time dependence (i.e for a capacitor, $I = \frac{dV_c}{dt}$). A time dependent circuit is said to have reached steady state when the currents and voltages have "stabilized."

2.1 RC Circuits

The simplest time dependent circuit is the RC circuit which contains a charged capacitor, a resistor, and a switch.
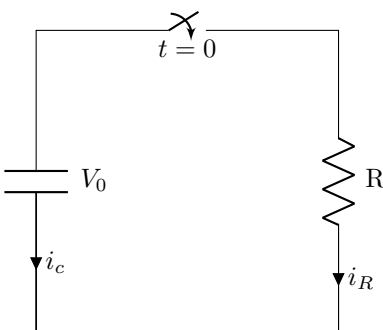
Figure 8: A capacitor discharging

At $t = 0$, the switch closes and current starts flowing in the circuit. Lets try and find the voltage across the capacitor V_C over time.

$$i_c = -i_R$$

$$C\frac{dV_c}{dt} = -\frac{V_c}{R}$$

$$\frac{dV_c}{dt} = -\frac{V_c}{RC}$$

Trying to solve the equation using nodal analysis gives us a first order differential equation. The form of this equation looks incredibly similar to the matrix equation $A\vec{v} = \lambda\vec{v}$. In this case, $\lambda = -\frac{1}{RC}$. This means we can try and solve this equation by asking ourselves "What are the eigenvectors of differentiation"

Remember that $\frac{d}{dt}ce^{\lambda t} = \lambda * ce^{at}$, and this is exactly the form we need. That means that we can think of $ce^{\lambda t}$ as an eigenvector of differentiation. Now we know $V_C = ce^{-\frac{t}{RC}}$ for some constant c. To find c, remember that $V_C(t = 0) = V_0$, so solving the equation $ce^{-\frac{0}{RC}} = V_0$ gives $c = V_0$. Thus $V_C(t) = V_0e^{-\frac{t}{RC}}$.

The value RC is called the time constant of the circuit. At $t = RC$, the voltage on the capacitor is about $V_0e^{-1} \approx 0.67V_0$. To better understand the circuit, lets look at the graph of V_C.

8
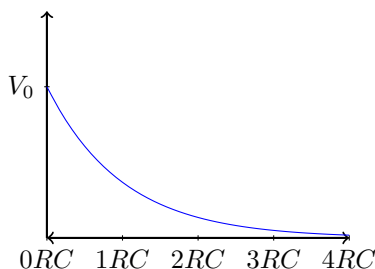
Notice how the voltage decays to zero after a long time. Now let's consider how a capacitor charges up.
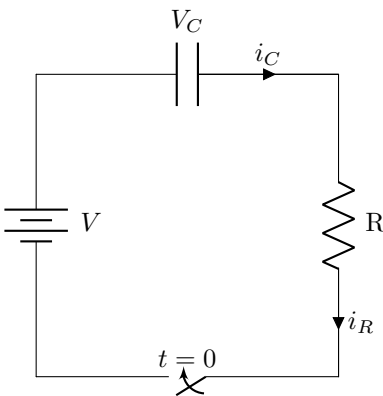
Figure 9: A capacitor charging

At time $t = 0$, the switch closes and currents start flowing to charge the capacitor. As we did for the discharging capacitor, lets apply nodal analysis.

$$V - V_C - i_C R = 0$$

$$RC\frac{dV_c}{dt} = V - V_C$$

$$\frac{dV_c}{dt} = \frac{1}{RC}(V - V_C)$$

This equation looks different from the earlier one, but we can solve it in the same way if we just change variables. Let $z = V - V_C$.

$$\frac{dz}{dt} = -\frac{dV_C}{dt}$$

$$\therefore -\frac{dz}{dt} = \frac{1}{RC}z$$

$$\frac{dz}{dt} = -\frac{1}{RC}z$$

This is the same differential equation we saw before, so $z = ce^{-\frac{t}{RC}}$. At $t = 0$, the capacitor is uncharged, so $V_C = 0$, thus $z = V$ at $t = 0$. This means $c = V$. Now we can change variables back to get V_C

$$Ve^{-\frac{t}{RC}} = V - V_C$$

$$V_C = V(1 - e^{-\frac{t}{RC}})$$

Looking at a graph of this tells us how the capacitor charges up.
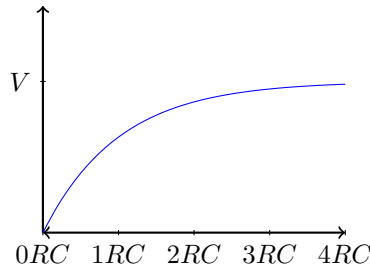
Over time, the voltage on the capacitor takes on the voltage of the battery. The time it takes for capacitors to charge and discharge explain why there are limits to how fast computers can work. Remember that part of the transistor model contains a capacitor, so the time it takes for the capacitor to charge up limits how fast the transistor can switch between logic 0 and logic 1.

2.2   RL Circuits

Another circuit element which is commonly used in circuits is the inductor. The inductor tries to oppose changes in voltage. Its IV relationship is $V = L\frac{dI}{dt}$ where L is a constant called the inductance.

The analysis of inductors is the same as capacitors, but the behaviors are flipped.
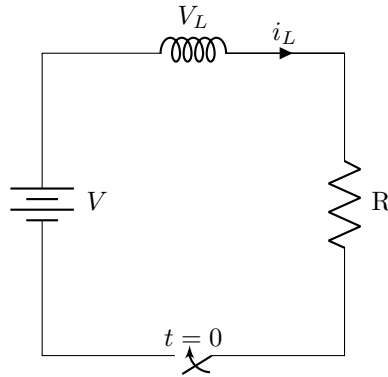
Figure 10:

$$V - V_L - i_L R = 0$$

$$V - L\frac{dI_L}{dt} - i_L R = 0$$

$$\frac{dI_L}{dt} = \frac{V_s - i_L R}{L}$$

Solving this equation yields $I_L = I_0(1 - e^{-\frac{Rt}{L}})$ where I_0 is the maximum current. Notice how this is the same behavior that a capacitor exhibits but for the

current instead.

Another easy comparison that can be made between inductors and capacitors is the energy stored in them. The energy in a capacitor $U = \frac{1}{2}CV^2$, whereas the energy in an inductor is $\frac{1}{2}LI^2$.

3 Frequency Analysis

Solving differential equations is incredibly difficult, and for complex circuits, it is difficult to account for all the variables which are changing and to actually solve the differential equations. Moreover, if we stop using DC inputs and instead use AC inputs, the time domain analysis becomes even more complicated.

To deal with this issue and to analyze more complex circuits, it becomes easier to analyze the "frequency response" of a circuit. In other words, how does it respond to inputs of different frequencies.

3.1 Phasor Representation

Let's start by analyzing how a simple resistor circuit responds to different frequency voltages.
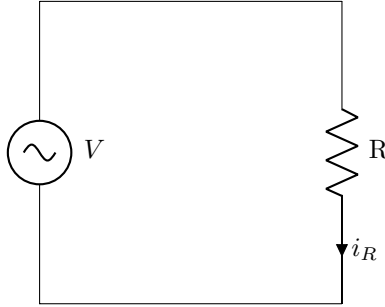
Our input to this circuit is $V(t) = V_0 cos(\omega t + \theta)$. Using Euler's formula, we can represent this as $V(t) = \frac{V_0}{2}e^{j\theta}(e^{j\omega t} + e^{-j\omega t})$. Notice how $e^{-j\omega t}$ is the complex conjugate of $e^{j\omega t}$, which means if we know the angular frequency ω, all of the "information" in the input is described by the term $\frac{V_0}{2}e^{j\theta}$ (i.e we have both the amplitude and the phase of the signal encoded into a single complex number). This encoding is known as a phasor, and lets call it $\tilde{V} = \frac{V_0}{2}e^{j\theta}$.

Definition 8 *A phasor encodes both the amplitude and phase of a sinusoidal input to a circuit.*

Notice that we can do normal circuit analysis with phasors. For example, the current in this circuit is $I = V/R = \frac{V_0}{R}V_0 cos(\omega t + \theta)$. It's phasor representation is $\tilde{I} = \frac{V_0}{2R}e^{j\theta}$. This is exactly what we would get if we applied Ohm's law on the phasor $\tilde{V}$ (i.e, $\tilde{V} = \tilde{I}R$)

3.2   Impedance

In the time domain, the relationship between the voltage and current is the resistance. In the frequency domain, we define a more general concept called Impedance.

Definition 9 *The Impedance Z of a circuit element is the ratio between the voltage phasor $\tilde{V}$ and the current phasor $\tilde{I}$.*

$$Z = \frac{\tilde{V}}{\tilde{I}}$$

3.2.1   Impedance of a Resistor

Recall that for the simple resistor circuit above, $\tilde{V} = \frac{V_0}{2}e^{j\theta}$ and $\tilde{I} = \frac{V_0}{2R}e^{j\theta}$. Thus the impedance $Z = \frac{\tilde{V}}{\tilde{I}} = R$. So for a resistor, the impedance is equivalent to the resistance ($Z = R$).

3.2.2   Impedance of a Capacitor
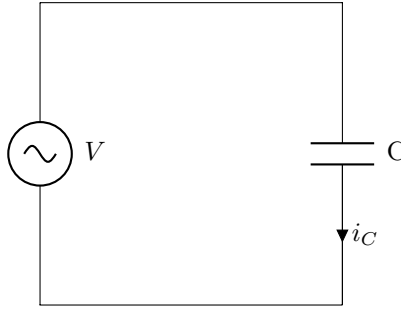
Now lets find the impedance of a capacitor. $V = V_0 cos(\omega t + \theta)$, so $\tilde{V} = \frac{V_0}{2}e^{j\theta}$ hasn't changed.

$$i_c = C\frac{dV}{dt} = -C\omega V_0 sin(\omega t + \theta)$$

Since $sin(\omega t + \theta) = \frac{1}{2j}(e^{j\omega t} - e^{j\omega t})$

$$\tilde{I} = \frac{-C\omega V_0}{2j}e^{j\theta} = -C\omega V_0 j e^{j\theta}$$

Now we can compute the impedance

$$Z = \frac{\tilde{V}}{\tilde{I}} = \frac{1}{j\omega C}$$

This is the impedance of the capacitor.

3.2.3 Impedance of an Inductor
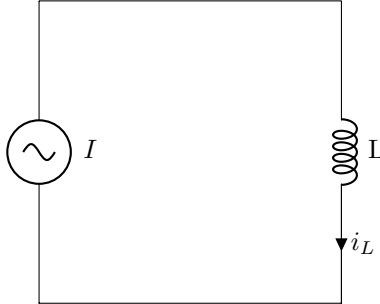
Let's do the same analysis for the inductor. Here our input signal is $I = I_0 cos(\omega t + \theta)$. That means $\tilde{I} = \frac{I_0}{2} e^{j\theta}$.

$$V = L\frac{dI}{dt} = -LI_0\omega sin(\omega t + \theta)$$

$$\tilde{V} = -\frac{LI_0\omega}{2j} e^{j\theta} = -\frac{LI_0\omega j}{2} e^{j\theta}$$

$$Z = \frac{\tilde{V}}{\tilde{I}} = j\omega L$$

This is the impedance of the inductor.

3.3 Phasor Analysis

Now that we know the impedances of different circuit components, we can start to analyze circuits with them. Consider this RC circuit
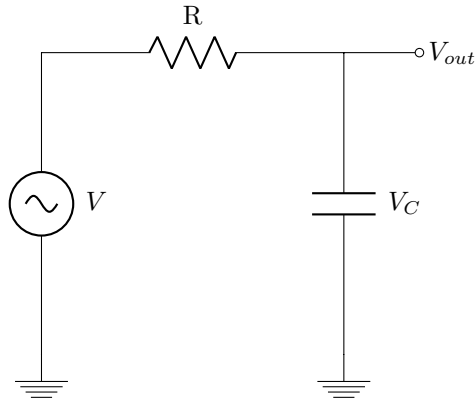
If we wanted to find V_{out}, we could write out the differential equations and solve, but with phasors, everything becomes easier. First we convert everything to the phasor domain. Sinusoidal inputs become DC inputs, and resistors, capacitors, and inductors become impedances.
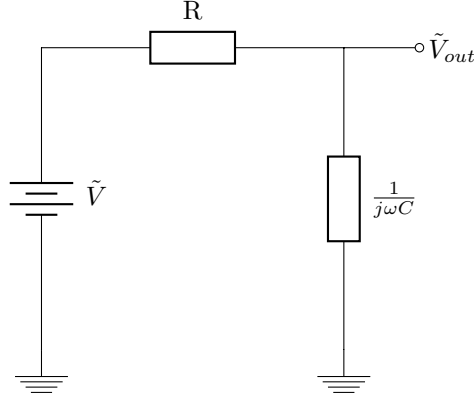
This is just a voltage divider, so

$$\tilde{V}_{out} = \frac{\frac{1}{j\omega C}}{R + \frac{1}{j\omega C}}\tilde{V} = \frac{1}{1 + j\omega RC}\tilde{V}$$

Lets define $H(\omega) = \frac{1}{1+j\omega RC}$, so $\tilde{V}_{out} = H(\omega)\tilde{V}$ Now if our original signal was $V = cos(\omega t)$, then $\tilde{V} = \frac{1}{2}$, so $\tilde{V}_{out} = \frac{1}{2}H(\omega)$. To convert this back into the time domain, we know the phasor is encoding some sinusoidal signal, so

$$V_{out} = \tilde{V}_{out}e^{j\omega t} + \tilde{V}_{out}e^{-j\omega t} = \frac{1}{2}|H(\omega)|cos(\omega t + \angle H(\omega))$$

Thus we have recovered our output signal using no calculus.

3.4 Transfer Functions

Notice how in the previous example, the relationship between our "output" $\tilde{V}_{out}$ and our input $\tilde{V}$ was described by a function $H(\omega)$. This function is called the transfer function.

Definition 10 *The Transfer Function of a circuit is defined as $H(\omega) = \frac{\tilde{X}}{\tilde{V}}$ where $\tilde{X}$ is the phasor output and $\tilde{V}$ is the phasor input*

The transfer function gives us a convenient way to analyze how a circuit will affect an input signal depending on its angular frequency, effectively treating the circuit like a black box. An important note is that transfer functions do not just apply to voltages, they can also apply to currents, but for the most part, we will deal with voltages.

All transfer functions can be expressed in the form $H(\omega) = \frac{Z(\omega)}{P(\omega)}$. $Z(\omega)$ and $P(\omega)$ are complex polynomials. If we factor $P(\omega)$ and $Z(\omega)$, then we see that all transfer functions can be expressed as follows.

$$H(\omega) = \frac{(j\omega)^{N_{z0}}\prod_{z=0}^{n}(j\omega)^z\alpha_z}{(j\omega)^{N_{p0}}\prod_{p=0}^{m}(j\omega)^p\beta_p} = K\frac{(j\omega)^{N_{z0}}\prod_{i=0}^{n}(1+\frac{j\omega}{\omega_{zi}})}{(j\omega)^{N_{p0}}\prod_{k=0}^{m}(1+\frac{j\omega}{\omega_{pk}})}$$

The constants ω_z are the zeros of the complex polynomial $Z(\omega)$, and the constants ω_p are the series of the complex polynomial $P(\omega)$. ω_z are called the zeros of the transfer function, and ω_p are called the poles. N_{z0} and N_{p0} are the number of zeros and poles respecively at $\omega = 0$.

To see how this makes analysis easier, lets look at the transfer function we saw earlier $H(\omega) = \frac{1}{1+j\omega RC}$. We can rewrite $H(\omega) = \frac{1}{1+\frac{\omega}{\omega_p}}$, where $\omega_p = \frac{1}{RC}$. Notice how when $\omega >> \omega_p$, $H(\omega) \approx 0$ and when $\omega << \omega_p$, $H(\omega) \approx 1$. This tells us that for frequencies higher than the pole, the input will be attenuated, but for frequencies less than the pole, the signal will be mostly unaffected.

3.5 Bode Plots

For more complex transfer functions, we need a easier way of analyzing them than taking limits. This is where Bode Plots come in.

Definition 11 *A Bode Plot is a plot of $|H(\omega)|$ and $\angle H(\omega)$ on a log-log scale*

These visually reveal the behavior of circuits. The log-log scale not only allows us to determine the behavior of Bode plots over a large range of frequencies, but they also let us easily figure out what the plot looks like because it converts the transfer function into piecewise linear components.

To see why, lets write our transfer function in polar form.

$$H(\omega) = K\frac{(j\omega)^{N_{z0}} \prod_{i=0}^{n}\left(1 + \frac{j\omega}{\omega_{zi}}\right)}{(j\omega)^{N_{p0}} \prod_{k=0}^{m}\left(1 + \frac{j\omega}{\omega_{pk}}\right)} = Ke^{j\frac{\pi}{2}(N_{z0}-N_{p0})}\frac{\prod_{i=0}^{n} r_{zi}}{\prod_{k=0}^{m} r_{pk}}e^{j(\sum_{i=0}^{n} z_i - \sum_{k=0}^{m} p_k)}$$

Each r is the magnitude of a factor $1 + \frac{j\omega}{\omega_n}$ where ω_n is either a root or a pole, and the z_i, p_k are the phases of each factor. By writing $H(\omega)$ this way, it is clear that

$$|H(\omega)| = K\frac{\prod_{i=0}^{n} r_{zi}}{\prod_{k=0}^{m} r_{pk}}$$

If we take the log of this, we get

$$log(|H(\omega)|) = log(K) + \sum_{i=0}^{n} log(r_{zi}) - \sum_{k=0}^{m} log(r_{pk})$$

Likewise, the exponential form of $H(\omega)$ tells us that

$$\angle H(\omega) = \frac{\pi}{2}(N_{z0} - N_{p0}) + \left(\sum_{i=0}^{n} z_i - \sum_{k=0}^{m} p_k\right)$$

Next, we should verify if we can approximate these equations as linear on a log-log scale. Take the example transfer function $H(\omega) = \frac{1}{1+\frac{j\omega}{\omega_p}} = \frac{1}{r_p}e^{-j\theta_p}$.

$$
\begin{array}{llll}
\text{if } \omega = \omega_p & H(\omega) = \frac{1}{1+j} & r_p = \sqrt{2} & \theta_p = \frac{\pi}{4} \\
\text{if } \omega = 10\omega_p & H(\omega) = \frac{1}{1+10j} & r_p \approx 10 & \theta_p \approx \frac{\pi}{2} \\
\text{if } \omega = 0.1\omega_p & H(\omega) = \frac{1}{1+0.1j} & r_p \approx 1 & \theta_p \approx 0
\end{array}
$$

Thus we can see at decades away from the poles and zeros, the magnitudes and the phases will have less of an effect. Let's try constructing the Bode Plot for this transfer function.
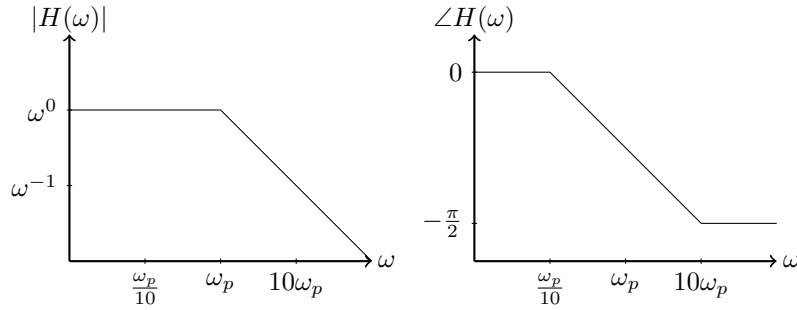
For the magnitude plot, since there are no poles or zeros at $\omega = 0$, we draw a straight line until the pole kicks in at $\omega = \omega_p$ at which point the slope of the line will be -1. For the phase plot, we apply the same logic, except the pole kicks in at $\frac{\omega_p}{10}$ (to see why, look above to see how at $\omega = \omega_p$, the phase is $-\frac{\pi}{4}$). We can apply this same logic for more complicated transfer functions too. Lets take

$$
H(\omega) = 10^9 \frac{(1 + \frac{j\omega}{10^9})}{(j\omega)(1 + \frac{j\omega}{10^7})}
$$

Notice we have a zero at 10^9, poles at $1, 10^7$, and 9 zeros at $\omega = 0$. With this information, we can see the plots will look like this:

16
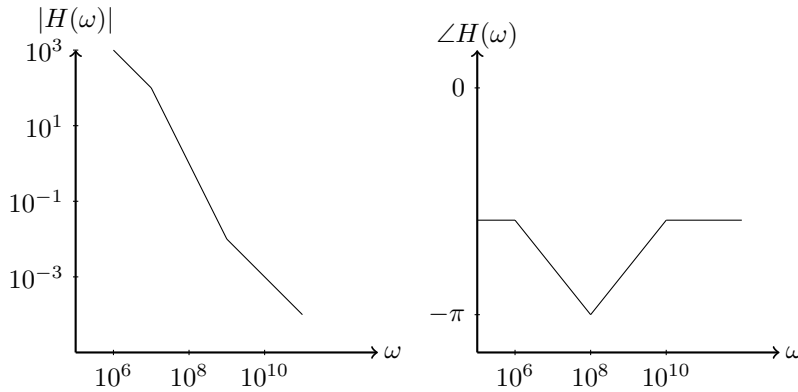
The pole at 0 kicks in immediately, causing the decreasing magnitude and starting the phase at $\frac{-\pi}{2}$. The second pole at 10^7 will kick in next, followed by the zero at 10^9.

3.6 Filters

One type of circuit which Bode plots make very easy to analyze are filters.

3.6.1 Low Pass Filters

Definition 12 *A low pass filter is one which allows frequencies below a particular cutoff frequency to pass through while higher ones are attenuated.*

Two common low pass filters are the RC filter and the LR filter.
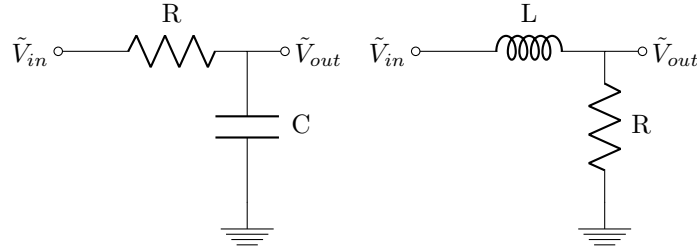
Figure 11: Low Pass Filters

These both have the transfer function $H(\omega) = \frac{1}{1 + \frac{j\omega}{\omega_p}}$ where $\omega_p = \frac{1}{RC}$ or $\omega_p = \frac{R}{L}$ depending on which filter it is. In this case, ω_p is called the cutoff frequency.

3.6.2 High Pass Filters

Definition 13 *A high pass filter is on which allows frequencies higher than a particular cutoff frequency to pass through while lower ones are attenuated.*

Two common high pass filters are the RL and CR filters.
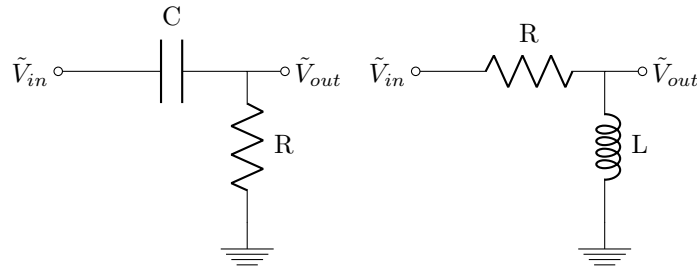
Figure 12: High Pass Filters

These both have the transfer function $H(\omega) = \frac{j\omega}{1 + \frac{\omega}{\omega_p}}$ where $\omega_p = \frac{1}{RC}$ or $\omega_p = \frac{R}{L}$ depending on which filter it is. Once again, ω_p is called the cutoff frequency.

3.6.3 LC Tank

Definition 14 *The LC tank is an inductor and a capacitor in series. When the input is at the resonating frequency, these act like an infinite impedance.*
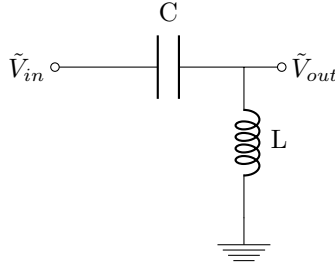
Figure 13: LC Tank

The transfer function $H(\omega) = \frac{j\omega L}{j\omega L + \frac{1}{j\omega C}}$. If $\omega = \omega_n = \sqrt{\frac{1}{LC}}$, then this transfer function gets infinitely large effectively acting like an infinite impedance. ω_n is called the resonant frequency.

4 Discrete Time Models

Although circuits and other systems function in continuous time, computers can only act in discrete time. This means the signals they send out to try and control the system are piecewise-continuous, and they can only sense the system at specified intervals δ. At a discrete time-step i, the computer will measure $\vec{x}_d[i]$ and applies the control $\vec{u}[i]$ over the time-interval $[i, i+1]$. Keep in mind that $t = \delta i$ where t is the continuous time.

Definition 15 *The discrete-time state space model is*

$$\vec{x}_d[i+1] = A_d \vec{x}_d[i] + B_d \vec{u}[i] + \vec{w}[i]$$

where $\vec{w}[i]$ is some disturbance.

4.1 Converting between continuous time and discrete time

To see how to convert between a continuous time model to a discrete time model, consider the following system.

$$\frac{dx}{dt} = \lambda x + u, u = e^{st}$$

18

Solving will give $x = \alpha e^{\lambda t} + \beta e^{st}$ Plugging this into the original equation

$$\alpha \lambda e^{st} + \beta s e^{st} = \lambda(\alpha e^{\lambda t} + \beta e^{st}) + e^{st}$$
$$\text{if } \lambda \neq s$$
$$\beta s = \beta \lambda + 1 \implies \beta = \tfrac{1}{s - \lambda}$$

Since the computer only outputs constant signals, $s = 0$, so $x = \alpha e^{\lambda t} - \tfrac{u}{\lambda}$ If $x(0) = x_0$

$$x(0) = \alpha - \tfrac{u}{\lambda} = x_0$$
$$x = (x_0 + \tfrac{u}{\lambda} e^{\lambda t}) - \tfrac{u}{\lambda}$$
$$x = x_0 e^{\lambda t} + u \left(\tfrac{e^{\lambda t} - 1}{\lambda} \right)$$

Notice that at $x(\delta) = x_0 e^{\lambda \delta} + u \left(\tfrac{e^{\lambda \delta} - 1}{\lambda} \right)$ This means that $x[2\delta]$ can be written as $x[2\delta] = x(\delta) e^{\lambda \delta} + u \left(\tfrac{e^{\lambda \delta} - 1}{\lambda} \right)$ because we simply take $x(\delta)$ to be our initial condition. This allows us to write the discrete time model as

$$x[i + 1] = e^{\lambda \delta} x[i] + \left(\frac{e^{\lambda \delta} - 1}{\lambda} \right) u[i]$$

since the initial condition for the next time-step is simply the value at the previous time-step. For a vector continuous system, A_d and B_d can be found in the same way by simply solving the differential equation for each state variable and converting those to discrete time models and combining the results into matrix-vector form.

4.2   System Identification

What would happen if we didn't know what A and B were though? It turns out that we can find it by gathering data about the system and then using least squares. In the scalar case:

$$x[i + 1] = ax[i] + bu[t] + w[i]$$
$$x[1] = ax[0] + bu[0]$$
$$x[2] = ax[1] + bu[1]$$
$$\vdots$$
$$x[m] = ax[m - 1] + bu[m - 1]$$

This means with m observations, we can construct a matrix D and a vector $\vec{s}$ such that we have the least squares problem

$$D\vec{p} = \vec{s}, \text{ where } \vec{p} = \begin{bmatrix} a \\ b \end{bmatrix}$$

The same thing can be done in the vector case. We just create multiple parallel systems by multiplying out the rows. This is called system identification.

5 Control Theory

5.1 Controllability

Definition 16 *A system is controllable if $\forall x^*, \forall x[0], \exists u[0], u[1], ..., u[k-1]$ such that $x[k] = x^*$ given that the system starts at $x[0]$*

In words, this means that there exists a sequence of control actions that can eventually get the system anywhere we want it to be from any given initial state.

Theorem 1 *If a state is n-dimensional, then a system*

$$\vec{x}_d[i+1] = A\vec{x}[i] + B\vec{u}[i] + \vec{w}[i]$$

is controllable in k time-steps if $span(B, AB, ..., A^{k-1}B) = \mathbb{R}^n$

Proof 1 *Without any noise, $\vec{x}_d[i+1] = A\vec{x}[i] + B\vec{u}[i]$*

$$\vec{x}[1] = A\vec{x}[0] + B\vec{u}[0]$$
$$\vec{x}[2] = A\vec{x}[1] + B\vec{u}[1] = A^2\vec{x}[0] + AB\vec{u}[0] + B\vec{u}[i]$$
$$\vec{x}[3] = A\vec{x}[2] + B\vec{u}[2] = A^3\vec{x}[0] + A^2B\vec{u}[0] + AB\vec{u}[i] + B\vec{u}[2]$$
$$\vdots$$
$$\vec{x}[k] = A^k\vec{x}[0] + \sum_{i=0}^{k-1} A^{k-i-1}B\vec{u}[i]$$

Thus the current state is merely a linear combination of vectors in the column spaces of A^k and $B...A^{k-1}$, so if this span is equal to $\mathbb{R}^n$, we can reach every state.

Definition 17 *The controllability matrix is a the matrix whose column space is all the reachable states.*

$$\mathcal{C} = \left[\ A^{k-1}B\ \middle|\ A^{k-2}B\ \middle|\ ...\ \middle|\ AB\ \middle|\ B\ \right]$$

However, if we want to check if a system is controllable in 100 time-steps for a 100 state system, it will be painful to calculate the column space of the controllability matrix.

It turns out that we can stop checking when $dim(col(B, AB, ..., A^{p-1})) = dim(col(B, AB, ..., A^p)), (p \leq k)$ (i.e the dimension of the column space stops growing) This means we can build the controllability matrix up piece-by-piece and stop early rather than computing the full column space only to fail. Consider the following proof for why this works.

Proof 2 *If $\vec{b}$ is column vector of the matrix B, then $A^p\vec{b} = \sum_{i=0}^{p-1} \alpha_i A^i\vec{b}$ since we assume the dimension of the column space has stopped growing, so $A^p\vec{b}$ is a linear combination of the vectors preceding it. We want to show $\exists \beta_i$ such that*

$$A^{p+1} = \sum_{i=0}^{p-1} \beta_i A^i \vec{b}$$

$$A^{p+1}\vec{b} = A(A^p\vec{b}) =$$

$$A\sum_{i=0}^{p-1} \alpha_i A^i \vec{b} = \sum_{i=0}^{p-1} \alpha_i A^{i+1}\vec{b} =$$

$$\alpha_{p-1}A^p\vec{b} + \sum_{i=0}^{p-2} \alpha_i A^i \vec{b} =$$

$$\alpha_{p-1}\sum_{i=0}^{p-1} \alpha_i A^i \vec{b} + \sum_{i=0}^{p-2} \alpha_i A^i \vec{b}$$

By writing A^{p+1} as a linear combination of $B, AB, ..., A^{p-1}$, we have shown β_i exists, so by induction, all future vectors $p + m \leq k$ will be linear combinations of these same vectors, so the dimension will never grow again and we can be justified in stopping early.

With the controllability matrix, if we want to figure out the controls needed to reach a particular state in $p < k$ steps, all that needs to be done is solve

$$\mathcal{C}_p\vec{u} = \vec{x}^* \text{ where } \vec{u} = \begin{bmatrix} \vec{u}[0] \\ \vec{u}[1] \\ \vdots \\ \vec{u}[p] \end{bmatrix}, \mathcal{C}_p = \begin{bmatrix} A^{p-1}B & | & A^{p-2}B & | & ... & | & AB & | & B \end{bmatrix}$$

Important: this assumes that $\vec{x}[0] = 0$. For an arbitrary $\vec{x}[0]$, we would solve $C_p\vec{u} = \vec{x}^* - \vec{x}[0]$. Look back to where the controllability matrix was derived to see why.

5.2 Observability

Another question we can ask about different systems is whether or not we can figure out exactly where we are.

Definition 18 *A system is observable if from a future state $\vec{x}[n]$, the original state $\vec{x}[0]$ is recoverable.*

Often times, we are limited in the measurements we can take, so we might not be able to "see" the entire state vector $\vec{x}[t]$. Instead, we would only be able to see the output $\vec{y}[t] = C\vec{x}[t]$. To start with, consider what would happen if we applied no inputs to the system.

$$\vec{y}[0] = C\vec{x}[0]$$

$$\vec{y}[1] = C\vec{x}[1] = CA\vec{x}[0]$$

$$\vdots$$

$$\vec{y}[n] = CA^n\vec{x}[0]$$

Definition 19

$$\mathcal{O} = \begin{bmatrix} C \\ CA \\ \vdots \\ CA^i \end{bmatrix}, \ \vec{y} = \begin{bmatrix} \vec{y}[0] \\ \vec{y}[1] \\ \vdots \\ \vec{y}[n] \end{bmatrix}$$

is the observability matrix for a system.

As long as $\mathcal{O}$ is full rank, if we want to recover the state $\vec{x}[0]$, we could apply least squares.

$$\vec{x}[0] = (\mathcal{O}^T \mathcal{O})^{-1} \mathcal{O}^T \vec{y}$$

This brings us the following theorem.

Theorem 2 *A system is observable if $\mathcal{O}$ has linearly independent columns.*

Even if a system has inputs, this kind of analysis still holds. Define

$$\vec{x}_f[i] = A\vec{x}[i], \ \vec{y}_f[i] = C\vec{x}_f[0]$$

This is what the system would do without any inputs. If we unroll the recursion in the system, we see

$$\vec{x}[i] = \vec{x}_f[i] + \sum_{k=0}^{i-1} A^{i-k-1} B\vec{u}[k]$$

$$\vec{y}[i] = \vec{y}_f[i] + \sum_{k=0}^{i-1} CA^{i-k-1} B\vec{u}[k]$$

Since we know the inputs we applied, and $\vec{x}_f[i] = A^i \vec{x}[i]$, we can easily find $\vec{x}_f[i], \ \vec{y}_f[i]$ and then apply least squares to them to compute the initial state.

5.3   Stability

The question of stability is whether or not disturbances will over time control a system if left alone. Specificically, it is asking if the disturbances are bounded, will the states be bounded too?

Definition 20 *Bounded Input, Bounded Output (BIBO) stability means if*

$$|\vec{w}[i]| < \epsilon, \ then \ \exists \delta > 0 \ such \ that \ |\vec{x}[i]| < \delta$$

To see where this definition comes into play, consider the following scalar systems:

1. $x[i+1] = u[i] + w[i]$

2. $x[i+1] = \frac{1}{2}x[i] + u[i] + w[i]$ ("Weak" dependence on previous state)

3. $x[i+1] = 2x[i] + u[i] + w[i]$ ("Strong" dependence on previous state)

System 1: In this system, there is no dependence on the previous state. Thus any disturbances are limited to affecting only the next state and no others. This makes the system stable because disturbances will not add up. To see this mathematically, pretend $u[i] = 0$ since we want to see how the system evolves without doing anything.

$$|x[i+1]| = |w[i]| < \epsilon, \quad \therefore |x[i]| < \epsilon$$

Thus we have BIBO stability.

System 2: Unlike the previous system, this system has a "weak" dependence on previous states. We consider it "weak" because only half of the previous state influences the next state. Intuitively, this system must be BIBO stability then because although disturbances will accumulate, they will be attenuated through an infinite geometric sum. Once again, we will pretend $\vec{u}[i] = 0$ to check this mathematically.

$$|x[i+1]| = \frac{1}{2}x[i] + w[i] < \frac{1}{2}x[i] + \epsilon$$

Unrolling the recursive relationship, we see

$$|x[i+1]| < \epsilon + \frac{1}{2}\epsilon + \frac{1}{4}\epsilon + ... + \left(\frac{1}{2}\right)^i \epsilon$$

$$\lim_{i->\infty} |x[i+1]| < \frac{\epsilon}{1 - \frac{1}{2}} = 2\epsilon$$

Thus the system is BIBO stability since $|x[i]| < 2\epsilon$

System 3: This system has a "strong" dependence on previous states since the contribution of previous states is doubled each times. Intuitively, this means the system must be unstable because small disturbances will be explode over time. To see this mathematically, once again consider what happens if we apply no inputs.

$$|x[i+1]| = 2x[i] + w[i] < 2x[i] + \epsilon$$

Unrolling the recursion,

$$|x[i+1]| < \epsilon + 2\epsilon + 4\epsilon + ... + 2^i\epsilon$$

$$\lim_{i->\infty} |x[i+1]| < \infty$$

Thus $|x[i]|$ has no bound, so the system is not stable.

Note that in the scalar case, as long as $|\lambda| < 1$ for the system $x[i+1] = \lambda x[i] + bu[i] + w[i]$, the system will be stable. Note that this is $\lambda < 1$, not $\lambda \leq 1$. That is because when $\lambda = 1$, we can see the errors will add up.

Definition 21 *A discrete system is marginally stable if $\lambda = 1$*

We call it marginally stable because if there are no disturbances, then it would be stable.

Theorem 3 *A vector discrete system $\vec{x}[t+1] = A\vec{x}[t] + B\vec{u}[t] + \vec{w}[t]$ will be stable if the eigenvalues of A are less than 1.*

Proof 3 *If $\exists\, n\, \vec{v}_i$ such that $A\vec{v}_i = \lambda_i \vec{v}_i$, then changing coordinates to the eigenbasis gives*

$$\tilde{A} = \begin{bmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_n \end{bmatrix}$$

This means $\vec{\tilde{x}}[t+1] = \tilde{A}\vec{\tilde{x}}[t] + V^{-1}B\vec{u}[t] + V^{-1}\vec{w}[t]$ where V is the matrix whose columns are A's eigenvectors.

This is merely n different scalar cases, but we still have to make sure that $|\vec{w}[t]| < \epsilon \implies |V^{-1}\vec{w}[t]| < \kappa$ where κ is some bound.

Lets say $m = V_{ij}^{-1}$ is the largest entry in V^{-1}. In the worst case all $\vec{w}_i[t]$ and all $V_{ij} = m$. Then $\forall i,\, \left[V^{-1}\vec{w}[t]\right]_i \leq nm\epsilon$ where n is the dimension of V, which means $|V^{-1}\vec{w}[t]| < \kappa$.

Thus we can say that $\tilde{x}[t]$ is BIBO implies that $\vec{x}[t]$ is BIBO.

It is important to realize that the condition $|\lambda| < 1$ holds only for discrete-time systems. If instead our system is continuous (i.e $\frac{d}{dt}\vec{x}(t) = A\vec{x}(t) + B\vec{u}(t)$), then the system will only be stable if $Re\{\lambda\} < 0$. To see why, consider the following simple linear system with initial condition x_0.

$$\frac{d}{dt}x(t) = \lambda x(t)$$
$$\therefore x(t) = x_0 e^{\lambda t}$$

Clearly if $\lambda > 0$, this system will grow infinitely large. The same analysis applies to vector cases.

Theorem 4

1. A continuous system is stable if the real part of its eigenvalues are less than 0 (i.e $Re\{\lambda\} < 0$)

2. A discrete system is stable if $|\lambda| < 1$

5.3.1 Stability for a general A matrix

Notice that checking stability based on the eigenvalues requires the matrix A to be diagonalizable becase we need to check all of its eigenvalues. This leaves

open the question of what happens when A is not diagonalizable, meaning there are not n distinct eigenvalues. In order to resolve this problem, we are going to need a new matrix form called Schur Upper Triangular Form.

Definition 22 *The Schur Upper Triangular form of a matrix A is*

$$\begin{bmatrix} \lambda_1 & * & * & ... \\ 0 & \lambda_2 & * & ... \\ 0 & 0 & \ddots & \\ \vdots & \vdots & & \end{bmatrix}$$

It is an upper triangular matrix with eigenvalues on the diagonal. The eigenvalues may equal = 0.

Schur form allows us to easily find the characteristic polynomial of the matrix and solve the problem of stability. To see why it helps with stability, consider the following proof.

Proof 4
$$\tilde{x}[t+1] = \tilde{A}\tilde{x}[t] + \tilde{w}[t]$$

If $\tilde{A}$ is in Schur form, then the nth component is

$$\tilde{x}_n[t+1] = \lambda_n \tilde{x}_n[t] + \tilde{w}_n[t]$$

so if $|\lambda_n| < 1$, then $\tilde{x}_n$ is BIBO stable. Likewise, for the n-1th component

$$\tilde{x}_{n-1}[t+1] = \lambda_{n-1}\tilde{x}_{n-1}[t] + s\tilde{x}_n[t] + \tilde{w}_{n-1}[t]$$

s is one of the $*$ entries in the upper part of Schur form. It doesn't matter what it is because $\tilde{x}_n[t]$ is BIBO, so $s\tilde{x}_n[t]$ is also BIBO.

As a result $\tilde{x}_{n-1}[t]$ will be BIBO if $|\lambda_{n-1}| < 1$ and $|\lambda_n| < 1$. Since the component $\tilde{x}_{n-i}[t+1]$ depends on $\tilde{x}_n[t], \tilde{x}_{n-1}[t], ...,$ as long as $\tilde{x}_n[t], \tilde{x}_{n-1}[t], ...$ are stable, $\tilde{x}_{n-i}[t+1]$ will be stable too.

Thus as long as all $|\lambda_i| < 1$, then Schur form tells us whether or not the system is stable.

Now that we know how it works, we need to know how to find a basis U such that $\tilde{A} = U^{-1}AU$. As a bonus, it would be helpful if U is orthonormal. To make things easier later, we'll define $M_n = A$. Since all matrices have at least one eigenpair, we can choose

$$M\vec{v}_1 = \lambda_1 \vec{v}_1, ||\vec{v}_1|| = 1$$

Using Gram-Schmidt, we can now pick $n-1$ vectors $\vec{r}_i$ such that

$$U_1 = \vec{v}_1, \vec{r}_1, ..., \vec{r}_{n-1}$$

is an orthonormal basis. Note, the vectors $\vec{r_i}$ form a $n \times (n-1)$ matrix R_{n-1}

Lets look at M_n in the U_1 basis.

$$U_1^{-1} M_n U_1 = \begin{bmatrix} \vec{v_1}^T \\ R_{n-1}^T \end{bmatrix} M_n \begin{bmatrix} \vec{v_1} & R_{n-1} \end{bmatrix} = \begin{bmatrix} \vec{v_1}^T M_n \vec{v_1} & \vec{v_1}^T M_n R_{n-1} \\ R_{n-1}^T M_n \vec{v_1}^T & R_{n-1}^T M_n R_{n-1} \end{bmatrix} = \left[\begin{array}{c|c} \lambda_1 & * \\ \hline 0 & R_{n-1}^T M R_{n-1} \end{array} \right]$$

We can see that we are getting closer to where we want to be, so lets continue with the matrix M_{n-1}. We can find a vector of size $n-1$ such that

$$M_{n-1} \vec{v_2} = \lambda_2 \vec{v_2}, \|\vec{v_2}\| = 1$$

Extending this to an orthonormal basis U_2, we get a matrix R_{n-2}

$$\begin{bmatrix} \vec{v_2}^T \\ R_{n-2}^T \end{bmatrix} M_{n-1} \begin{bmatrix} \vec{v_2} & R_{n-2} \end{bmatrix} = \begin{bmatrix} \vec{v_2}^T M_{n-1} \vec{v_2} & \vec{v_2}^T M_{n-1} R_{n-2} \\ R_{n-1}^T M_{n-1} \vec{v_1}^T & R_{n-1}^T M_{n-1} R_{n-1} \end{bmatrix} = \left[\begin{array}{c|c} \lambda_2 & * \\ \hline 0 & R_{n-2}^T M_{n-1} R_{n-2} \end{array} \right]$$

We can keep repeating this by saying $M_{n-2} = R_{n-2}^T M_{n-1} R_{n-2}$, but we need to know if our λ_2 has any relation to the eigenvalues of A. Lets say

$$U_2 = \vec{v_1}, \vec{x}, rest$$

is a basis where $\vec{x}$ has some dependence on $\vec{v_2}$. There aren't many options for what $\vec{x}$ and the rest of the basis vectors could be, so let's try.

$$\vec{x} = R_{n-1} \vec{v_2}, \ rest = R_{n-1} R_{n-2}$$

because this will get us the appropriate sized vectors. Notice this basis will also be orthonormal because $\vec{v_1}^T R_{n-1} = 0$ and $\vec{v_2} R_{n-1}^T R_{n-1} R_{n-2} = \vec{v_2}^T R_{n-2} = 0$

Lets see how M_n looks in the basis U_2 to see if we are getting closer to Schur form.

$$U_2^{-1} M_n U_2 = \begin{bmatrix} \vec{v_1}^T \\ (R_{n-1}\vec{v_2})^T \\ (R_{n-1}R_{n-2})^T \end{bmatrix} M_n \begin{bmatrix} \vec{v_1}^T & R_{n-1}\vec{v_2} & R_{n-1}R_{n-2} \end{bmatrix} =$$

$$\begin{bmatrix} \vec{v_1}^T M_n \vec{v_1} & \vec{v_1}^T M_n R_{n-1}\vec{v_2} & \vec{v_1}^T M_n R_{n-1}R_{n-2} \\ (R_{n-1}\vec{v_2})^T M_n \vec{v_1}^T & (R_{n-1}\vec{v_2})^T M_n R_{n-1}\vec{v_2} & (R_{n-1}\vec{v_2})^T M_n R_{n-1}R_{n-2} \\ (R_{n-1}R_{n-2})^T M_n \vec{v_1} & (R_{n-1}R_{n-2})^T M_n R_{n-1}\vec{v_2} & (R_{n-1}R_{n-2})^T M_n R_{n-1}R_{n-2} \end{bmatrix} =$$

$$\left[\begin{array}{c|c|c} \lambda_1 & * & * \\ \hline 0 & \lambda_2 & * \\ \hline 0 & 0 & (R_{n-1}R_{n-2})^T M_n R_{n-1}R_{n-2} \\ \vdots & \vdots & \end{array} \right]$$

26

The λ_2 showed up because

$$\vec{v}_2^T R_{n-1}^T M_n R_{n-1} \vec{v}_2) = \vec{v}_2 M_{n-1} \vec{v}_2 = \lambda_2$$

This tells us we can continue with this process to get to Schur form. To get to Schur Form, all we need to do is

Let $M_n = A$

Loop:

Find $\lambda_i, \vec{v}_i$ such that $M_{n+1-i}\vec{v}_i = \lambda_i \vec{v}_i$, $||\vec{v}_i|| = 1$

Find R_{n-i} such that $R_{n-i}^T \vec{v}_i = 0$ and $R_{n-i}^T R_{n-i} = I_{n-1}$

$M_{n-i} = R_{n-i}^T M_{n+1-i} R_{n-i}$

End Loop

$$U = \vec{v_1}, \ R_{n-1}\vec{v}_2, \ R_{n-1}R_{n-2}\vec{v}_2, \ ..., \ (\textstyle\prod_{j=1}^{k} R_{n-j})\vec{v}_{k+1}$$

Thus $\tilde{A} = U^T A U$ will give the Schur Form of the matrix A. Notice that $(U^T A U)^T = U^T A^T U$. This forms the basis for **Spectral Theorem**.

Theorem 5 (Spectral Theorem) *Every real symmetric matrix has real eigenvalues can be orthogonally diagonalized*

Two consequences of spectral theorem are that if the symmetric matrix A is real, then

1. All eigenvalues are real

2. All eigenvectors are real

5.4   Closed Loop Control

Lets say we have a system which is modeled by $\vec{x}[t+1] = A\vec{x}[t] + B\vec{u}[t] + \vec{w}[t]$ We must choose our inputs $\vec{u}[t]$ based on only $\vec{x}[t]$ since we can't observe the disturbances. How should we choose our inputs to control the system? If we try $\vec{u}[t] = k\vec{x}[t]$, then

$$\vec{x}[t+1] = A\vec{x}[t] + Bk\vec{x}[t] + \vec{w}[t]$$
$$\vec{x}[t+1] = (A + Bk)\vec{x}[t] + \vec{w}[t]$$

Which means that if we choose K appropriately, we can potentially change the behavior of the system. Consider the following simple scalar case example

$$x[t+1] = 3x[t] + u[t] + w[t]$$
$$u[t] = kx[t]$$
$$x[t+1] = (3+k)x[t] + w[t]$$

27

Notice that even if the system is initially unstable, we can set our k to make it stable since the domain of $\lambda = 3 + k$ is ∞

What if we have a system $\vec{x}[t+1] = A\vec{x}[t] + \vec{b}u[t] + \vec{w}[t]$ where $A, \vec{b}$ are controllable, but we only get a scalar input $u[t]$. Can we find a set of feedback gains $\vec{f}^T = [f_0, f_1, ..., f_{k-1}]$ such that $u[t] = -\vec{f}^T\vec{x}[t]$ makes $A - \vec{b}\vec{f}^T$ have the eigenvalues we want?

We know how to easily solve scalar cases like $z[t + 1] = az[t] + u[t]$, but it is hard to get this from a vector case because we only have a simple input, so we will be limited in how we can set our eigenvalues this way. Since this won't work, consider the folowing recursive system.

$$z[t + 1] = a_{k-1}z[t] + a_{k-2}z[t - 1] + ... + a_0z[t - (k - 1)] + u[t]$$

This is definetly controllable since our input $u[t]$ could potentially be any value in $\mathbb{R}$. Moreover, if we choose $u[t] = -\sum_{i=0}^{k-1} f_i z[t - (k - 1) + i]$, we can create any recurrence relation we want.

For example, if we wanted $z[t+1] = d_{k-1}z[t] + d_{k-2}z[t-1] + ... + d_0z[t - (k-1)]$, then $f_i = a_i - d_i$ Writing this in vector form, we get

$$\vec{z}[t + 1] = A_z\vec{z}[t] + \vec{b}u[t]$$

$$\vec{z}[t] = \begin{bmatrix} z[t - (k - 1)] \\ \vdots \\ z[t - 2] \\ z[t - 1] \\ z[t] \end{bmatrix}$$

Notice that the vector $\vec{z}[t]$ here is different from the scalar $z[t]$ which it is made up of. Each "state" here is a scalar quantity, but to represent the recursive relationship properly, we collect all of the previous states in a vector to define the next state.

Given our definition of $\vec{z}[t]$ (which are the scalar values of previous states), we can fill in values for A_z and $\vec{b}$

$$A_z = \begin{bmatrix} 0 & 1 & 0 & 0 & ... & 0 \\ 0 & 0 & 1 & 0 & ... & 0 \\ 0 & 0 & 0 & 1 & ... & 0 \\ \vdots & & & & \ddots & \vdots \\ a_0 & a_1 & a_2 & ... & & a_{k-1} \end{bmatrix}, \vec{b} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}$$

To understand where these come from, thing of the A_z matrix "scrolling down" the state vector $\vec{z}[t]$ to get how it influences $\vec{z}[t + 1]$. All the rows except the

28

last are 0's and 1's since the previous states can't change, but the last row determines the next state.

Notice that all entries in the b-vector are 0 except the last since we can't affect the past with present control. We can assume $a_0 \neq 0$ since we defined the recurrence to have length k. Breaking A_z into its block components gives us some more insight to its structure.

$$A_z = \left[\begin{array}{c|c} 0 & I_{k-1} \\ \hline & \vec{a}^T \end{array} \right]$$

We can check through multiplication that this system will be controllable.

Definition 23 *A system is in Controllable Canonical Form if*

$$A_z = \left[\begin{array}{c|c} 0 & I_{k-1} \\ \hline & \vec{a}^T \end{array} \right], \ \vec{b}_z = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}$$

Let's try finding the eigenvalues of A_z. If A_z is large, it'll be hard to calculate the characteristic polynomial, so instead lets guess their form and figure them out that way.

$$\vec{v}_\lambda = \begin{bmatrix} 1 \\ * \\ * \\ \vdots \end{bmatrix}$$

Where * is any element. We know $A_z \vec{v}_\lambda = \lambda v_\lambda$, and since A_z scrolls the vector down, we can see that the second * must be λ. Applying this same logic to the second *, it must be λ^2. Continuing this, we get

$$\vec{v}_\lambda = \begin{bmatrix} 1 \\ \lambda \\ \lambda^2 \\ \vdots \\ \lambda^{k-1} \end{bmatrix}$$

If we wanted to find the characteristic polynomial, we could look at the last entry in $A_z \vec{v}_\lambda$

$$\sum_{i=0}^{k-1} a_i \lambda_i = \lambda * \lambda^{k-1} = \lambda_k$$

Rearranging: $\lambda_k - \sum_{i=0}^{k-1} a_i \lambda_i = 0$

Notice this is the characteristic polynomial of A. That means if we want to simply find what the CCF form of the system looks like, rather than computing

the transformation $\mathbb{HG}^{-1}$, we can instead use the fact that $a_0, a_1, ...a_{k-1}$ are the coefficients of the characteristic polynomial.

Along the same train of thought, if we want to design feedback controls $\vec{u}[t] = -\vec{f}^T \vec{z}[t]$ to make the system stable, all we need to do is set $f_i = a_i - d_i$ where d_i are the desired coefficients of the desired characteristic polynomial. To see why, refer back to the scalar case or try multiplying out $(A_z - \vec{b}_z \vec{f}^T)$. Thus CCF is useful because it makes it extremely easy to place the eigenvalues for any system.

Now that we know about this matrix form, the question becomes: can we transform any controllable system with matrix A into controllable canonical form? (i.e can we make $\vec{x}[t+1] = A\vec{x}[t] + \vec{b}u[t]$ into $\vec{z}[t+1] = A_z\vec{z}[t] + \vec{b}_z u[t]$)

We know that $\vec{b}, A\vec{b}, A^2\vec{b}, ...A^{k-1}\vec{b}$ are linearly independent since our system is controllable. Let this define basis $\mathbb{G}$. If we write the matrix A in the basis of G, we get $\tilde{A} = G^{-1}AG$. The naming choice is very appropriate here because this matrix multiplication will be painful by hand, so lets look for another way to get $\tilde{A}$.

Remember that any linear transformation is defined by what it does to the basis vectors. We know

$$\tilde{b} = G^{-1}\vec{b} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \end{bmatrix}$$

since $\vec{b}$ is the first basis vector of $\mathbb{G}$. Thus,

$$\tilde{A}\tilde{b} = G^{-1}A\vec{b} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ \vdots \end{bmatrix}, \ \tilde{A}^2\tilde{b} = G^{-1}A^2\vec{b} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ \vdots \end{bmatrix} ...$$

This gives us k columns of $\tilde{A}$. For the last column must be a linear combination of all the basis vectors, so $\tilde{A}^k\tilde{b} = \sum_{i=0}^{k-1} \alpha_i \tilde{A}^i\tilde{b}$ This means

$$\tilde{A} = \begin{bmatrix} 0 & 0 & ... & \alpha_0 \\ 1 & 0 & ... & \alpha_1 \\ 0 & 1 & ... & \alpha_2 \\ \vdots & & \ddots & \vdots \\ 0 & ... & 1 & \alpha_{k-1} \end{bmatrix}$$

which is A_z^T. This means that our basis $\mathbb{G}$ is essentially the controllability matrix $\mathcal{C}$ except with the rows reordered.

30

Now we have a basis $\mathbb{G}$ with $(\tilde{A}, \tilde{b})$, but we want a basis $\mathbb{H}$ with $(\tilde{A}^T, \vec{b}_z)$ We want $\tilde{A}^T, \vec{b}_z$ to be controllable, which means $\vec{b}_z, \tilde{A}^T \vec{b}_z, ..., (\tilde{A}^T)^{k-1} \vec{b}_z$ are linearly independent. Let's construct a matrix H which will take us from a coordinates in basis $\mathbb{H}$ to coordinates in the basis $\mathbb{G}$.

$$\mathbb{H} = \left[\ \vec{b}_z \ \middle| \ \tilde{A}^T \vec{b}_z \ \middle| \ ... \ \middle| \ (\tilde{A}^T)^{k-1} \vec{b}_z \ \right]$$

Once we have the matrix H, to get $(\tilde{A}^T, \vec{b}_z)$ from $(A, \vec{b})$, we can apply the change of coordinates $G^{-1}H$ so $\vec{x}_z[t] = HG^{-1}\vec{x}[t]$ and put our entire system into controllable canonical form.

5.5   Linearization

Up until now, we have been assuming our systems are linear. To understand what this means, we must first understand what linearity means. First, let's define what we know as a function.

Definition 24 *A function is a simple mapping from a domain to a range (i.e y(t)=L(u(t))). To evaluate y(t), we only need to know u(t), making the function "memory-less"*

Definition 25 *A function $f(x)$ is linear iff*

- $\forall \alpha, \forall x \in \mathbb{R}, f(\alpha x) = \alpha f(x)$ *(Scaling)*

- $\forall u, \forall v \in \mathbb{R}, f(u + v) = f(u) + f(v)$ *(Superposition)*

Notice that by this definition, any linear function must necessarily have $f(0) = 0$ since otherwise with $\alpha = 1, x = 0, f(1 * 0) = f(0) = 1 * f(0) \neq 0$

These conditions can be simplified down into a single expression

$$f(\alpha u + v) = \alpha f(u) + f(v), \forall u, \forall v, \forall \alpha \in \mathbb{R}$$

If we can find a single setting of (u, v, α) for which this equation does not hold, then our function is not linear.

If we have a scalar nonlinear function we can linearize it in a simple way. Remember that the Taylor series expansion of a function around a certain point is

$$f(x) = \frac{f(c)}{0!} + \frac{f'(c)}{1!}(x - c) + \frac{f''(c)}{2!}(x - c)^2 + ... = \sum_{i=0}^{\infty} \frac{f^{(i)}}{i!}(x - c)^i$$

where c is some expansion point we choose. Lets say we choose some point x^* and decide to linearize around that point. The tangent line to the function is

$$f_L(x) = f'(x^*)(x - x^*) + f(x^*)$$

However, $f(0) \neq 0$, so this tangent line is still not linear. To linearize it, we can change coordinates.

$$m = \frac{df}{dx}|_x$$

$$f(x^* + \delta x) = f(x^*) + m\delta x$$

$$\delta y = f(x^* + \delta x) - f(x^*)$$

$$f(x^* + \delta x) = f(x^*) + \delta y = f(x^*) + m\delta x$$

$$\delta y = m\delta x$$

Thus the linearized expression is in terms of the "error" δy, a.k.a how far it deviates from what it was at the point of linearization. While this makes sense for functions, how does it apply to systems. Systems are more than simple functions, they are functionals.

Definition 26 *A functional $y(t) = L\{u(t)\}$ is a function which requires knowing the entire u in order to evaluate it.*

A good example of a functional is

$$L\{u(t)\} = \int_{\tau-1}^{\tau} u(t)d\tau$$

The definition of linearity for functionals is virtually the same as for a normal function.

Definition 27 *A system $L\{\}$ is linear iff*

- $\forall \alpha \in \mathbb{R}, \forall u(t), L\{\alpha u(t)\} = \alpha L\{u(t)\}$ *(Scaling)*

- $\forall u(t), \forall v(t), L\{u(t) + v(t)\} = L\{u(t)\} + L\{v(t)\}$ *(Superposition)*

With this definition, we can now formally define how to linearize a non-linear system. Take the system $\frac{d}{dt}x(t) = f(x(t)) + bu(t)$ where f is a nonlinear function and b is a scalar multiplier. To linearize the system:

1. Choose a constant with time input $u(t) = u^*$

2. Find a solution $x(t) = x^*$ to the system

$$\frac{dx^*}{dt} = f(x^*) + bu^* \implies f(x^*) = -bu^*$$

3. Linearize $f(x)$ around x^*

$$f(x) \approx f(x^*) + \frac{df}{dx}|_{x^*}(x - x^*)$$

4. Use the linearization of f to make the differential equation linear

$$\frac{dx}{dt} = f(x^*) + \frac{df}{dx}|_{x^*}(x(t) - x^*) + bu(t)$$

Let $\delta x = x(t) - x^*, \delta u = u(t) - u^*$

$$\frac{d}{dt}(\delta x + x^*) = f(x^*) + \frac{df}{dx}|_{x^*}\delta x + b(u^* + \delta u)$$

$$\frac{d}{dt}\delta x = \underline{f(x^*) + bu^*}^{\nearrow 0} + \frac{df}{dx}|_{x^*}\delta x + b\delta u(t)$$

$$\frac{d}{dt}\delta x = \frac{df}{dx}|_{x^*}\delta x + b\delta u(t)$$

Note that this only works assuming that δx is small, so when linearizing our systems, we must check that this property is true otherwise our linearization may be invalid.

To see why, consider the system $\frac{d}{dt} = -sin(x)$. Try linearizing about $x^* = 0$.

$$\frac{d}{dt}\delta x = -\delta x$$

$$\therefore \delta x = \delta x(0)e^{-t}$$

Since δx is the "error" of our approximation, we can see that

$$\lim_{\delta x \to \infty} \delta x = 0$$

This means the linearization is stable, and our approximation is legitimate. However, if we linearize about $x^* = \pi$

$$\frac{d}{dt}\delta x = \delta x$$

$$\therefore \delta x = \delta x(0)e^t$$

$$\lim_{\delta x \to \infty} \delta x = \infty$$

Thus this linearization is unstable and will diverge over a long period of time, so it is not a good linearization.

These concepts of linearization extend beyond simple 1D functions. For example, the linearization of a 2D function is

$$f(x^* + \delta x, u^* + \delta u) \approx f(x^*, u^*) + \frac{\partial f}{\partial x}|_{x^*,u^*}\delta x + \frac{\partial f}{\partial u}|_{x^*,u^*}\delta u$$

We can further generalize the idea of linearization to vector functions.

$$f(\vec{x}) = \begin{bmatrix} f_1(x_1, x_2) \\ f_2(x_1, x_2) \end{bmatrix} \text{ where } \vec{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

If we linearize each of the functions inside the vector, we get

$$f(\vec{x^*} + \delta\vec{x^*}) = \left[\begin{array}{c} f_1(x_1^* + \delta x_1, x_2^* + \delta x_2) \\ f_2(x_1^* + \delta x_1, x_2^* + \delta x_2) \end{array} \right]$$

$$f_1(x_1^* + \delta x_1, x_2^* + \delta x_2) \approx f_1(x_1^*, x_2^*) + \frac{\partial f_1}{\partial x_1}|_{x_1^*, x_2^*}\delta x_1 + \frac{\partial f_1}{\partial x_2}|_{x_1^*, x_2^*}\delta x_2$$

$$f_2(x_1^* + \delta x_1, x_2^* + \delta x_2) \approx f_2(x_1^*, x_2^*) + \frac{\partial f_2}{\partial x_1}|_{x_1^*, x_2^*}\delta x_1 + \frac{\partial f_1}{\partial x_2}|_{x_1^*, x_2^*}\delta x_2$$

$$\therefore f(\vec{x^*} + \delta\vec{x^*}) = \left[\begin{array}{c} f_1(x_1^* + \delta x_1, x_2^* + \delta x_2) \\ f_2(x_1^* + \delta x_1, x_2^* + \delta x_2) \end{array} \right] = \left[\begin{array}{c} f_1(x_1^*, x_2^*) \\ f_2(x_1^*, x_2^*) \end{array} \right] + \left[\begin{array}{cc} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_1}{\partial x_2} \end{array} \right]|_{x_1^*, x_2^*} \left[\begin{array}{c} \delta x_1 \\ \delta x_2 \end{array} \right]$$

The matrix with partial derivatives is known as the **Jacobian** and is denoted either ∇f or J Thus a succinct form the linearization of a vector system is

$$f(\vec{x^*} + \delta\vec{x}) = f(\vec{x^*}) + J_x f(\delta\vec{x})$$

For multiargument functions,

$$f(\vec{x^*} + \delta\vec{x}, \vec{u^*} + \delta\vec{u}) = f(\vec{x^*}, \vec{u^*}) + J_x\delta\vec{x} + J_u\delta\vec{u}$$

5.6 Planning

With all of these tools, we can now tackle the problem of planning. With closed loop control, we know we can keep a system near a specific state even if the system is unstable by itself due to disturbances. We can use this fact to keep our system close to a specified "open loop" trajectory by redefining our equilibrium position is along a specific trajectory.

5.6.1 Open Loop Control

Lets say we have a controllable system $\frac{d}{dt}\vec{x}[t] = A\vec{x}[t] + \vec{b}u[t] + \vec{w}[t]$, a desired trajectory $\vec{x}_n[t]$, and a set of inputs that will take us along that trajectory $u_n[t]$. We can define $\vec{v}[t] = \vec{x}[t] - \vec{x}_n[t]$ and $u_v[t] = u[t] - u_n[t]$ as the trajectory deviation and input deviation respectively. Writing this equation in terms of the deviations gives us

$$\frac{d}{dt}\vec{x}[t] = A\vec{x}[t] + \vec{b}u[t] + \vec{w}[t]$$

$$\frac{d}{dt}(\vec{v}[t] + \vec{x}_n[t]) = A(\vec{v}[t] + \vec{x}_n[t]) + \vec{b}(u_v[t] + u_n[t]) + \vec{w}[t]$$

$$\frac{d}{dt}\vec{v}[t] + \frac{d}{dt}\vec{x}_n[t] = (A\vec{v}[t] + \vec{b}u_v[t]) + (A\vec{x}_n[t] + u_n[t]) + \vec{w}[t]$$

$$\frac{d}{dt}\vec{v}[t] = A\vec{v}[t] + \vec{b}u_v[t] + \vec{w}[t]$$

Since our original system was controllable, this system must be controllable as well. In fact, it will have all the same characteristics as the original system

(controllability, stability, observability) since they share the same A and $\vec{b}$.

This tells us that if the original system was not stable, then we will not be able to follow the trajectory $\vec{x}[n]$ since our deviation from the trajectory $\vec{v}[t]$ can blow up from disturbances. Lets apply feedback control to make sure the deviation does not grow expnentially large.

$$u_v[t] = \begin{bmatrix} f_0 & f_1 & f_2... \end{bmatrix} \vec{v}[t]$$

This will let us control the deviation from the desired trajectory. This lets us compute the controls we need to follow the trajectory by inserting this result back into the formula for $u[t]$

$$u[t] = u_n[t] + u_v[t] = u_n[t] + \begin{bmatrix} f_0 & f_1 & f_2 & ... \end{bmatrix} \vec{v}[t]$$

$$u[t] = u_n[t] + \begin{bmatrix} f_0 & f_1 & f_2 & ... \end{bmatrix} (\vec{x}[t] - \vec{x}_n[t])$$

5.6.2 Minimum Energy Control

For any generic system, we know that if the system is controllable, we can use controls to take it to any state we want. For example, if the state is 10-dimensional, then it will take a maximum of 10 time-steps to move anywhere. But what should we do if we want to get to a system in 100 time-steps instead?

One approach is to move to the desired state x^* in 10 time-steps and then do nothing for the remaining 90 time-steps, but this is a bad solution because the dynamics of the environment may prevent our system from staying in one place at a time (imagine a jumping robot that needs to reach a height at a specific time. Continuously jumping up and down waste's a lot of energy)

Another idea is to visit random states for 90 time-steps and spend the next 10 moving towards the goal, but once again, this wastes energy. Ideally, we should take a "natural" path which uses the dynamics of the environment as much as possible.

A good example of this is a satellite such as Voyager. It could continuously use fuel to get where it needs to go, but it is much more energy efficient for it to use the gravity of planets to gain speed instead. This concept is known as minimum energy control.

Definition 28 *The minimum energy control problem is to find* $\operatorname{argmin} \|\vec{u}\|$ *such that* $C\vec{u} = \vec{x}^*$ *where C is the controllability matrix for the system, $\vec{x}^*$ is a desired state, and $\vec{u}$ is a vector of inputs applied at each time-step (assuming our inputs are scalar).*

Like most problems, this one is difficult to solve in its original state. Since the controllability matrix is of dimension $n \times k$ with $k > n$ (n is the dimension of the state and k is how many time-steps we want to reach our goal in), the

matrix must have a null space. Moving around inside this null space is a waste of energy because it doesn't get us close to x^*. This tells us two things about how we might solve this problem.

1. We should change coordinates to an orthonormal basis because it might make the problem easier to solve without changing the length of the vectors.

2. We should try an isolate the null space when we construct a basis because we don't want to move in those directions.

Mathematically, this means we want

$$V = \begin{bmatrix} \vec{v_1} & \vec{v_2} & ... & \vec{v_k} \end{bmatrix} \text{ such that } V^T V = I \text{ and } C\vec{v_i} = 0 \text{ for } i > n$$

In the new basis, the problem becomes

$$\operatorname{argmin} ||\tilde{\vec{u}}|| \text{ such that } CV\tilde{\vec{u}} = \vec{x}^*$$

Notice that the product $CV\tilde{\vec{u}}$ is a linear combination of only the first n basis vectors.

$$C \begin{bmatrix} | & | & & | \\ \vec{v_1} & \vec{v_2} & ... & \vec{v_k} \\ | & | & & | \end{bmatrix} \begin{bmatrix} \tilde{u}_1 \\ \vdots \\ \tilde{u}_n \end{bmatrix} = C(\tilde{u}_1\vec{v_1} + ... + \tilde{u}_n\vec{v_n})$$

This is because $C\vec{v_i}\tilde{u}_i = 0$ for $i > n$ based on the properties we want out of V. Clearly it would be very useful to have this basis V because then we could let $\tilde{u}_i = 0$ for $i > n$ to minimize $||\tilde{\vec{u}}||$.

At this point, we should try constructing V. We know that any symmetric matrix can be orthogonally diagonalized because of spectral theorem. Lets try finding a symmetric matrix Q such that $Null(Q) = Null(C)$ because then the eigenvectors of Q which form a basis for $Null(Q)$ will also form an orthogonal basis for $Null(C)$. Notice that $C^T C$ has this property.

Proof 5 $\vec{v} \in Null(C) \implies C\vec{v} = \vec{0}$, then $C^T C\vec{v} \implies \vec{v} \in Null(C^T C)$
$\vec{v} \in Null(C^T C) \implies C^T C\vec{v} = \vec{0}$, $C\vec{v} \cdot C\vec{v} = \vec{v}^T C^T C\vec{v} = \vec{0} \implies \vec{v} \in Null(C)$
Since any vector in $Null(C)$ belongs to $Null(C^T C)$ and vice versa, the two null spaces are the same.

Let's say V is the orthonormal basis of eigenvectors of Q which we know exists from the spectral theorem. If we order our eigenvectors such that their corresponding eigenvalues are related by $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_k \geq 0$, then we get

$$\Lambda = V^T(C^T C)V = \begin{bmatrix} \lambda_1 & 0 & ... & 0 \\ 0 & \lambda_2 & & \vdots \\ \vdots & & \ddots & \\ 0 & ... & & \lambda_k \end{bmatrix}$$

where only the first n diagonal entries are not zero

An interesting fact about these eigenvalues is that $\lambda_i = ||C\vec{v}_i||^2$

Proof 6

$$C^T C \vec{v}_i = \lambda_i \vec{v}_i$$
$$\vec{v}_i^T C^T C \vec{v}_i = ||C\vec{v}_i||^2$$
$$\vec{v}_i^T C^T C \vec{v}_i = \vec{v}_i^T Q \vec{v}_i = \lambda \vec{v}_i^T \vec{v}_i = \lambda_i$$
$$\therefore \lambda_i = ||C\vec{v}_i||^2$$

For convenience (and for reasons which will be clear later), lets defined $\sigma_i = \sqrt{\lambda_i} = ||C\vec{v}_i||$ to remove the exponent. A curious fact about the eigenvectors is that the matrix C does not change their orthogonality.

Proof 7 $(C\vec{v}_i^T)(C\vec{v}_j) = \vec{v}_i^T C^T C \vec{v}_j = \vec{v}_i^T \lambda_j \vec{v}_j = 0$

This means the vectors $C\vec{v}_i$ for $i \leq n$ form an orthonormal basis for $\mathbb{R}^n$ (we disregard $C\vec{v}_i$) for $i > n$ since these are merely zero vectors). Lets normalize this basis so

$$\vec{w}_i = \frac{C\vec{v}_i}{||C\vec{v}_i||} = \frac{C\vec{v}}{\sigma_i}$$

Now that we have extracted everything useful out of our basis V, let's return to the original problem of argmin $\tilde{u}$.

$$CV\tilde{u} = \sum_{i=1}^{n} C\vec{v}_i \tilde{u}_i =$$

$$\sum_{i=1}^{n} \sigma_i \tilde{u}_i \vec{w}_i = \vec{x}^*$$

By inspection, we can see that if we let $\tilde{u}_i = \frac{\vec{w}_i^T x_i^*}{\sigma_i}$, then the sum will work out. This means the following $\tilde{u}$ will solve our problem.

$$\tilde{u} = \begin{bmatrix} \frac{\vec{w}_1^T x_1^*}{\sigma_1} \\ \vdots \\ \frac{\vec{w}_n^T x_n^*}{\sigma_n} \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

And from these we can find the actual controls by $\vec{u}_{opt} = V\tilde{u}$.

6 SVD and PCA

The analysis which we did above reveals an incredibly interesting fact.

$$\tilde{u}_i = \vec{v}_i^T \vec{u}$$
$$C\vec{u} = \sum_{i=1}^{n} C\vec{v}_i \tilde{u}_i =$$

$$\sum_{i=1}^{n} \left(\sigma_i \vec{w}_i \vec{v}_i^T u_i \right) =$$

$$\sum_{i=1}^{n} \left(\sigma_i \vec{w}_i \vec{v}_i^T \right)\vec{u}$$

$$\therefore C = \sum_{i=1}^{n} \left(\sigma_i \vec{w}_i \vec{v}_i^T \right) = W\Sigma V^T$$

This shows that the matrix $C \in \mathbb{R}^{m \times n}$ can be factored into a product of three matrices. $W \in \mathbb{R}^{m \times m}$ has orthonormal columns, $\Sigma \in \mathbb{R}^{m \times n}$ has σ_i on the diagonal, and $V^T \in \mathbb{R}^{n \times n}$ also has orthonormal columns. Together, these matrices form the **Singular Value Decomposition** of C. The values σ_i on the diagonal are called the **singular values**. Note that C was a generic matrix, which means **any** matrix can be factored in this way.

Definition 29 *Any matrix $A \in \mathbb{R}^{m \times n}$ can be factored such that $A = U\Sigma V^T$ where $U \in \mathbb{R}^{m \times m}$, $\Sigma \in \mathbb{R}^{m \times n}$, $V \in \mathbb{R}^{n \times n}$ with U, V having orthonormal columns and Σ is diagonal.*

6.1 Computing the SVD

The procedure for computing the SVD of a matrix procedes exactly what was done above to find minimum energy control. First, we find the eigenpairs of $A^T A$ and these become our V matrix and $\lambda_i = \sigma_i^2$. To find U, we let $\vec{u}_i = \frac{A\vec{v}_i}{\sigma_i}$ for all $\sigma \neq 0$ and then use Gram Schmidt to find the rest of the basis. However, if the matrix A is $m \times n$, if $A^T A$ will be $n \times n$. If $n > m$, then this means we have to compute more eigenvectors. Instead, it would be less work to find the eigenvectors of AA^T and make those the vectors in U. In general, the SVD can be found as follows.

If $m > n$

1. $\vec{v}_i$ are the eigenvectors of $A^T A$, $\sigma_i = \sqrt{\lambda_i}$

2. $\vec{u}_i = \frac{A\vec{v}_i}{\sigma_i} \forall \sigma_i \neq 0$

3. Extend the basis of vectors $\vec{u}_1, \vec{u}_2, ..., \vec{u}_n$ into an orthonormal basis for $\mathbb{R}^m$ using Gram Schmidt.

If $m < n$

1. $\vec{u}_i$ are the eigenvectors of AA^T, $\sigma_i = \sqrt{\lambda_i}$

2. $\vec{v}_i = \frac{A^T \vec{u}_i}{\sigma_i} \forall \sigma_i \neq 0$

3. Extend the basis of vectors $\vec{v}_1, \vec{v}_2, ..., \vec{v}_m$ into an orthonormal basis for $\mathbb{R}^n$ using Gram Schmidt.

Extending a basis can be a lot of work, so there also exists the compact form of the SVD.

Definition 30 *Any matrix $A \in \mathbb{R}^{m \times n}$ can be factored such that $A = U\Sigma V^T$ where $U \in \mathbb{R}^{m \times k}$, $\Sigma \in \mathbb{R}^{k \times k}$, $V \in \mathbb{R}^{k \times n}$ with U, V having orthonormal columns and Σ is square. k is the number of nonzero singular values. This is known as the compact form of the SVD.*

The compact form is found by following the same procedure of finding the SVD but not extending any bases.

6.2 Outer Product Form of SVD

In the SVD of a matrix A, $A = \sum_{i=1}^{n} (\sigma_i \vec{w}_i \vec{v}_i^T)$, the term $\vec{w}\vec{v}^T$ is a special type of product known as an **outer product**. If $\vec{w} \in \mathbb{R}^n$ and $\vec{v} \in \mathbb{R}^m$, then $\vec{w}\vec{v}^T$ produces a $n \times m$ matrix of rank 1. All matrix multiplications can be expressed as a sum of outer products.

$$\begin{bmatrix} | & | & & | \\ \vec{x}_1 & \vec{x}_2 & ... & \vec{x}_m \\ | & | & & | \end{bmatrix} \begin{bmatrix} - & \vec{y}_1^T & - \\ - & \vec{y}_2^T & - \\ & \vdots & \\ - & \vec{y}_m^T & - \end{bmatrix} = \sum_{i=1}^{m} \vec{x}_i \vec{y}_i^T$$

This is why $A = \sum_{i=1}^{n} (\sigma_i \vec{w}_i \vec{v}_i^T)$ is known as the **outer product form** of the SVD. The outer product form shows us that every matrix can be expressed as a sum of outer products a.k.a a sum of rank 1 matrices. The outer products which correspond to higher singular values have more "weight" in making up the original matrix.

6.3 Approximating a matrix

Another outcome of outer products is that theoretically, if we have a large matrix (such as an image), we can retain most of its information but remember fewer numbers by throwing away the outer products which correspond to lower singular values. i.e for a matrix A which is $p \times q$

$$A = \sum_{i=1}^{m} \sigma_i \vec{u}_i \vec{v}_i^T \approx \sum_{i=1}^{k} \sigma_i \vec{u}_i \vec{v}_i^T$$

Where m is the smaller dimension of the matrix and $k < m$. The error of this approximation can also be found easily.

$$A - \hat{A} = \sum_{i=k+1}^{m} \sigma_i \vec{u}_i \vec{v}_i^T$$

These are merely the thrown away terms. We can reduce this to a number by taking the Frobenius Norm of the error matrix.

Definition 31 *The Froebenius norm of a matrix A, written $||A||_F$, is the square root of the sum of the squares of each element.*

$$||A||_F^2 = \sum_{i=1}^{n}\sum_{j=1}^{m}|a_{ij}|^2$$

With this definition, it is easy to see that the error of our approximation depends only on the approximations.

$$||A - \hat{A}||_F^2 = \sum_{i=k+1}^{m}\sigma_i^2$$

The following theorem contains useful features of the Frobenius Norm and can be proved fairly easily.

Theorem 6

1. $||A||_F^2 = Tr\{A^T A\}$

2. For an orthonormal matrix U, $||A||_F = ||UA|_F = ||AU||_F$

6.3.1   The Moore-Penrose Pseudoinverse

SVD gives us insight into how we might find a solution to $A\vec{x} = \vec{y}$ when $A \in \mathbb{R}^{m \times n}$ where $n > m$. If $Rank(A) = m$, then there will be infinitely many solutions. We can easily construct one solution by "undoing" the SVD.

$$A\vec{x} = \vec{y}$$
$$U\Sigma V^T \vec{x} = \vec{y}$$
$$\Sigma V^T \vec{x} = U^T \vec{y}$$

$$V^T \vec{x} = \begin{bmatrix} \frac{1}{\sigma_1} & 0 & \dots & 0 \\ 0 & \frac{1}{\sigma_2} & \dots & 0 \\ 0 & 0 & \ddots & \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix} U^T \vec{y}$$

$$\vec{x} = V \begin{bmatrix} \frac{1}{\sigma_1} & 0 & \dots & 0 \\ 0 & \frac{1}{\sigma_2} & \dots & 0 \\ 0 & 0 & \ddots & \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix} U^T \vec{y}$$

This "undoing" is known as the Moore-Penrose Pseudoinverse.

Definition 32 *The Moore-Penrose Psuedoinverse is $A^\dagger = V\tilde{\Sigma}U^T$ where*

$$\tilde{\Sigma} = \begin{bmatrix} \frac{1}{\sigma_1} & 0 & ... & 0 \\ 0 & \frac{1}{\sigma_2} & ... & 0 \\ 0 & 0 & \ddots & \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix}$$

$\tilde{\Sigma}$ can be thought of as $(\Sigma^T)^{-1}$ where divisions by 0 are ignored.

Notice that the Moore-Penrose Psuedo-inverse gives us the exact solution to the minimum energy control problem which we saw earlier.

6.4   PCA

PCA is a technique which is closely related to the SVD. It applies to matrices which represent data. Lets say we have a matrix $A \in \mathbb{R}^{n \times m}$ in which each row represents a data vector. To make analysis more accurate, it is nice to have all of our data centered around zero. To do this, find the mean of each column $\mu_1, ..., \mu_m$ and subtract out the column average from each data point.

$$\tilde{A} = \begin{bmatrix} x_{11} - \mu_1 & x_{12} - \mu_2 & ... & x_{1m} - \mu_m \\ x_{21} - \mu_1 & x_{22} - \mu_2 & ... & x_{2m} - \mu_m \\ & & \vdots & \\ x_{n1} - \mu_1 & x_{n2} - \mu_2 & ... & x_{nm} - \mu_m \end{bmatrix}$$

Definition 33

$$S \triangleq \frac{1}{n}\tilde{A}^T\tilde{A}$$

is the covariance matrix of A. n is the number of datapoints, and $\tilde{A}$ is the mean-centered A matrix.

$$S = \begin{bmatrix} s_{11} & ... & s_{1m} \\ \vdots & \ddots & \vdots \\ s_{m1} & ... & s_{mm} \end{bmatrix}$$

Theorem 7 *The covariance matrix S has the following properties.*

- $S^T = S$ *(i.e it is symmetric)*

- $S_{ii} \geq 0, S_{ii} \in \mathbb{R}$

- $|s_{ij}| \leq |s_{ii}s_{jj}|$

Notice that the diagonals contain the statistical variance of each "feature" of the data (each feature corresponds to a different column). (Remember that the variance $\sigma_x^2 = \frac{1}{n}\sum_{i=1}^{n} x_i - \mu$ for data points $x_1, ..., x_n$). To simplify notation, lets rename each diagonal entry to be $s_i^2 = s_{ii}$. With the covariance matrix, we can now find the correlation matrix, another statistical matrix which tells us the linear correlation between data.

Definition 34

$$R \triangleq \begin{bmatrix} 1 & \frac{s_{12}}{s_1 s_2} & ... & \frac{s_{1m}}{s_1 s_m} \\ \frac{s_{21}}{s_2 s_1} & \ddots & & \\ \vdots & & \ddots & \vdots \\ \frac{s_{m1}}{s_m s_1} & ... & ... & 1 \end{bmatrix}$$

The R matrix contains the correlations between each feature. All of the terms on its diagonal are 1, and $R^T = R$. Each $-1 \leq r_{ij} \leq 1$. A correlation of 1 means complete positive linear correlation while correlation of -1 means complete negative linear correlation. A correlation of 0 means no correlation at all. The correlation matrix is nice to do analysis on because it tells us how each feature is correlated. However, it is painful to calculate and if any of the diagonal entries of the covariance matrix is 0, then the R matrix will be undefined. PCA is a way to get around these issues.

PCA finds principal components of the data.

Definition 35 *The principal components of a matrix A are the eigenvectors of the covariance matrix S i.e, since S is symmetric, it can be othorgonally diagonalized into $S = P\Lambda P^T$. The principal components are merely the vectors in P.*

Principal components are the "axes" along which the data varies the most. Principal component analysis involves projecting the mean centered data $\tilde{A}$ onto the principle components ($\hat{A} = \tilde{A}P$)). Note that $\hat{A}$ will also be mean centered since

$$\begin{bmatrix} 1 & 1 & 1 & 1 & ... \end{bmatrix}\frac{}{n}\tilde{A}P = 0$$

$$\therefore \frac{\begin{bmatrix} 1 & 1 & 1 & 1 & ... \end{bmatrix}}{n}\hat{A} = \frac{\begin{bmatrix} 1 & 1 & 1 & 1 & ... \end{bmatrix}}{n}\tilde{A}P = 0$$

The reason why the eigenvectors of S are the axes along which the data varies the most is because if we compute $\hat{S}$, the covariance of the projected data, we get a diagonal matrix.

$$\hat{S} = \frac{1}{n}\hat{A}^T\hat{A} = \frac{1}{n}P^T\tilde{A}^T\tilde{A}P = P^TSP = \Lambda$$

The fact that covariance matrix is 0 means thats all of the variances between different features are 0, effectively meaning that we have "decoupled" the features. This gives us a good basis to analyze our data for tasks like classification

because features are not interacting with each other. Notice that the matrix P is the same as the matrix V in the SVD of $\tilde{A}$ since

$$S = \frac{1}{n}\tilde{A}^T\tilde{A} = \frac{1}{n}V\Sigma^T U^T U\Sigma V^T = \frac{1}{n}V\Sigma^T\Sigma V^T$$

Thus $\Lambda = \frac{\Sigma^T\Sigma}{n}$. This gives us an easy way to do PCA since we can just do SVD.

7 Signal Processing

7.1 Interpolation

Definition 36 *Interpolation is the process of reconstructing a signal from a series of samples.*

For example, if we have an audio signal and record the waveform's value at specific intervals, then interpolation allows us to approximate what the orginal waveform looked like using only the samples.

7.1.1 Interpolation with basis functions

One way which we can do interpolation is with basis functions.

Definition 37 *A basis function $\phi(t)$ is a function where*

$$\phi(t) = \left\{ \begin{array}{cc} 1, & if\ t = 0 \\ 0, & if\ t = k\Delta,\ k \neq 0 \end{array} \right\}$$

where Δ is the time interval at which samples were taken.
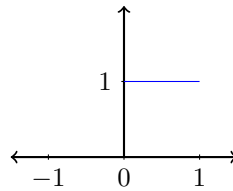
If we have samples $y_1, y_2, ..., y_n$, then the reconstructed function $f(x)$ is

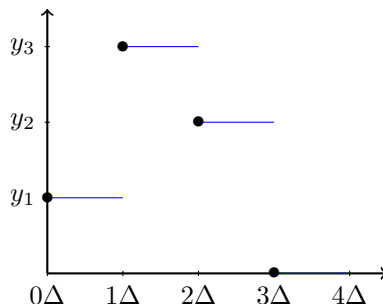$$f(x) = \sum_{i=1}^{n} y_i\phi(t - \Delta)$$

By interpolating with basis functions, our resulting point hits every sample properly (which is accomplished by shifting ϕ by Δ for each datapoint).

Definition 38 *Zero-Order-Hold interpolation is when*

$$\phi(t) = \left\{ \begin{array}{cc} 1, & if\ t = 0 \\ 0, & else \end{array} \right\}$$

When used in interpolation, Zero-Order-Hold looks like the following.

Notice that Zero-Order hold interpolation is not continuous, nor is it differentiable. In a sense, it is the simplest form of interpolation. Another really simply interpolation is piecewise-linear interpolation.

Definition 39 *The Piecewise linear basis function is*

$$\phi(t) = \left\{ \begin{array}{c} x + 1, \ \ if \ -\Delta < x <= 0 \\ -x + 1, \ \ if \ 0 <= x < \Delta \\ 0, \ \ else \end{array} \right\}$$
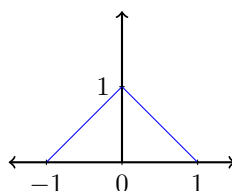
Piece-wise linear interpolation is equivalent to connecting each sample.
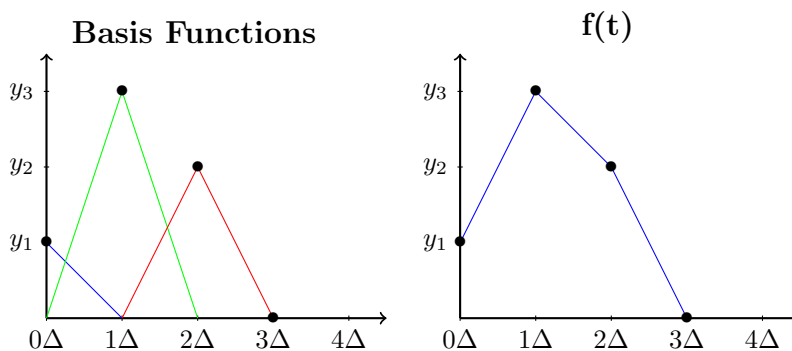
**Basis Functions**   **f(t)**

Piece-wise linear interpolation gives us a function which is continuous but not differentiable.

44

Definition 40 *Sinc interpolation has the basis function*

$$\phi(t) = \left\{ \begin{array}{l} 1, \ if \ t = 0 \\ \frac{sin(\pi t)}{\pi t}, t \neq 0 \end{array} \right\}$$
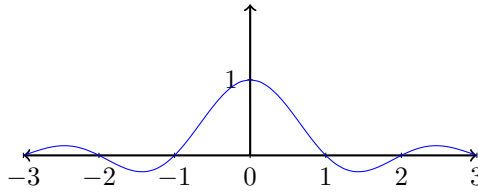
Notice how the amplitude is attenuated further from zero. The sinc function is both differentiable and continuous. Sinc interpolation will look something like this.
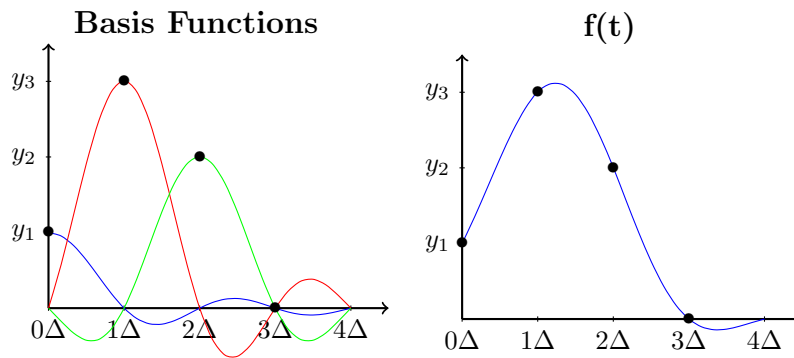
7.1.2   Polynomial Interpolation

Another form of interpolation is polynomial interpolation where the datapoints are fit to a polynomial. If we have n datapoints, then our polynomial is $y(t) = \sum_{i=0}^{n} a_i t^i$ To fit this polynomial to our datapoints (x_k, y_k), solve the system

$$\begin{bmatrix} 1 & x_0 & x_0^2 & ... & x_0^n \\ 1 & x_1 & x_1^2 & ... & x_1^n \\ & & \vdots & & \\ 1 & x_n & x_n^2 & ... & x_n^n \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

The matrix on the left is called the Vandermonde matrix, and it will always be invertible as long as there are no duplicate (x_k, y_k). While it might seem like polynomials are the best form of interpolation, they suffer heavily from over-fitting, so while they may perform well within samples, outside of the sample range, they degenerate very rapidly.

The process of fitting a polynomial to datapoints is computationally expensive because it requires taking an inverse. Instead, lets try and find so called Lagrange Polynomials $L_0...L_{n-1}$ to use as our basis functions. We know

$$L_i(x_j) = \left\{ \begin{array}{c} 1, \text{ if } j = i \\ 0, \text{ else} \end{array} \right\}$$

Since $L_i(x_j) = 0, \forall j \neq i$, we know that $(x - x_j)$ must be factors of the Lagrange polynomial. Since $L_i(x_j) = 1, i = j$, we know there must be a normalizing factor $x_i - x_j$. With these two restrictions, we can say the Lagrange polynomials for a set of data are

$$L_i(x) = \prod_{j=0; j \neq i}^{j=n-1} \frac{x - x_j}{x_i - x_j}$$

This makes the problem of fitting a polynomial equivalent to how we did piecewise linear, zero-order hold, and sinc interpolation. Namely,

$$f(x) = \sum_{i=0}^{n-1} y_i L_i(x)$$

7.2 Discrete Fourier Transform

One technique which is freqently used in signal processing is the Discrete Fourier Transform (DFT). An instance in which DFTs are particularly useful is in filtering. Building good filters out of hardware can be expensive, so instead, if we can filter signals digitally using the DFT, we can reduce the cost. Before exploring the DFT itself, we need to establish some details about complex numbers.

7.2.1 Complex Vectors and Matrices

Definition 41

$$||\vec{a}|| = \sqrt{\sum_{i=0}^{n-1} ||a_i||^2}, \ where \ \vec{a} \in \mathbb{C}$$

The norm of a complex vector is virtually identical to the norm of a real vector except in the complex case, we need to take the magnitude of each entry so we get a real number.

Definition 42 *The inner product of two complex vectors $< \vec{x}, \vec{y} >$*

$$< \vec{x}, \vec{y} > = \vec{y^*}\vec{x}$$

$\vec{y}^*$ is known as the Hermitian of the vector $\vec{y}$ and is equal to $\bar{\vec{y}}^T$ (the conjugate transpose). This inner prodoct reduces into the dot product when both $\vec{x}, \vec{y} \in \mathbb{R}$. Notice however, that $< \vec{x}, \vec{y} > \neq < \vec{y}, \vec{x} >$. Instead, $< \vec{x}, \vec{y} > = \overline{< \vec{y}, \vec{x} >}$(i.e, the conjugate of the commuted dot product).

7.2.2 Roots of Unity

When we allow ourselves to use complex numbers, the nth root of 1 becomes an interesting question. Since every complex number can be represented in polar form as $x = Me^{j\theta}$, $x^n = M^n e^{j\theta n}$. Graphically, this can be seen as a rotation around the unit circle.
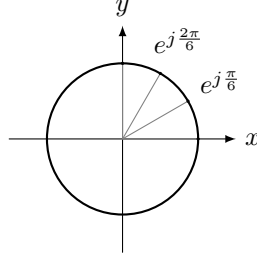
With this visual, it is easy to see that if $\theta = \frac{2\pi}{N}$, then $x^N = 1$ if $M = 1$.

Definition 43 *The nth roots of unity are $x = e^{\frac{2\pi k}{N}}$ for $k \in [0, N-1]$ We say the Nth root of unity is $\omega_N = e^{\frac{2\pi}{N}}$*

These roots of unity will play a large part in forming the DFT.

7.2.3 The DFT Matrix

Definition 44 *The following matrix is known as the DFT matrix.*

$$
F_N = \begin{bmatrix}
1 & 1 & \dots & \dots & 1 \\
1 & \omega_N^{-1*1} & \omega_N^{-1*2} & \dots & \omega_N^{-1*(N-1)} \\
\vdots & & \ddots & & \vdots \\
1 & \omega_N^{-(N-1)*1} & \dots & \dots & \omega_N^{-(N-1)(N-1)}
\end{bmatrix}
$$

The ijth entry in the DFT $F_{N(ij)} = \omega_N^{-ij}$. To see where the DFT matrix came from, check out `http://bit.ly/2XIIjrc`. Notice that the DFT matrix is a Vandermonde matrix. We'll call the kth column of the DFT $\vec{u}_k$

7.2.4 Properties of the DFT

1. The DFT is invertible.

2. $F_n = F_N^T$ (The DFT is symmetric)

3. $||\vec{u}_k|| = \sqrt{N}$ (The norm of each column is $\sqrt{N}$)

4. $< \vec{u}_i, \vec{u}_k > = 0$ if $k \neq i$ (The columns of F_N are orthogonal)

5. $\vec{u}_k = \bar{\vec{u}}_{N-k}, k \neq 0$

While the first three properties are trivial to show, the second two require a bit of proof.

Proof 8 *To see why the columns of the DFT are orthogonal, consider* $\vec{u}_k, \vec{u}_i, i \neq k$

$$\vec{u}_k^* = \begin{bmatrix} \omega_N^0 & \omega_N^k & ... & \omega_N^{(N-1)*k} \end{bmatrix} \quad \vec{u}_i = \begin{bmatrix} \omega_N^0 \\ \omega_N^{-i} \\ \vdots \\ \omega_N^{-(N-1)*i} \end{bmatrix}$$

$$< \vec{u}_k, \vec{u}_i > = \vec{u}_i^* \vec{u}_k = \sum_{l=0}^{N-1} \omega_N^{il} \omega_N^{-lk} =$$

$$\sum_{l=0}^{N-1} \omega_N l(i-k)$$

This is a finite geometric series with common ratio $\omega_N^{(i-lk)}$. *Therefore*

$$\sum_{l=0}^{N-1} \omega_N l(i-k) = \frac{1 - \omega_N^{N(i-k))}}{1 - \omega_N^{i-k}} = 0$$

Thus since the inner product between nonequal columns is 0, the columns are orthogonal.

Proof 9 *To see why* $\vec{u}_k = \bar{\vec{u}}_{N-k}, k \neq 0$, *consider that the entries of* $\vec{u}_{N-k}$ *are* $\omega_N^{-i(N-k)}$

$$\omega_N^{-i(N-k)} = \omega_N^{-iN} \omega_N^{ik} = \omega_N ik$$

$$\bar{\omega_N^{ik}} = \omega_N^{-ik}$$

ω_N^{-ik} *are the entries of* $\vec{u}_k$, *therefore the two columns must be complex conjugates.*

7.2.5 The Inverse DFT

Notice that $F_N^* F_N = N I_N$. This means that we can define $F_N^{-1} = \frac{1}{N} F_N^*$ to be the inverse DFT.

7.2.6 Using the DFT for frequency decomposition

If we pick a fundamental frequency f_0, then the so called harmonic frequencies are $0, f_0, 2f_0, ..., Mf_0$ where $M \in \mathbb{R}$. Any sinusoid which has one of these frequencies will be of the form $x_l(t) = A_l cos(2\pi(lf_0) + \theta_l)$ where $l \in [0, M]$.

Lets define the period to be $T = \frac{1}{f_0}$ since all of our harmonic frequencies will repeat over this period. Take one of these sinusoids $x_l(t)$ and sample it at N equally spaced points with $N = 2M + 1$. This gives us samples $x(k\Delta), \Delta = \frac{T}{N}, k \in [0, N-1]$. Take a specific sample $x(k\Delta)$

$$x(k\Delta) = A_l cos(2\pi(lf_0)k\Delta + \theta_l) = A_l cos(2\pi(lf_0)k\frac{T}{N} + \theta_l) = A_l cos(2\pi l \frac{k}{N} + \theta_l)$$

In complex form, this is

$$x(k\Delta) = \frac{A_l}{2} e^{\frac{j2\pi lk}{N}} e^{j\theta_l} + \frac{A_l}{2} e^{\frac{-j2\pi lk}{N}} e^{-j\theta_l}$$

Notice $e^{\frac{j2\pi lk}{N}} = \omega_N^k$. Lets define $c_l = \frac{A_l}{2} e^{j\theta_l}$ Then

$$x(k\Delta) = c_l \omega_N^{lk} + \bar{c}_l \bar{\omega}_N^{lk}$$

If we construct a vector out of our samples,

$$
\begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-1} \end{bmatrix} = c_l \begin{bmatrix} \omega_N^{l*0} \\ \omega_N^{l*1} \\ \vdots \\ \omega_N^{l*(N-1)} \end{bmatrix} + \bar{c}_l \begin{bmatrix} \omega_N^{-l*0} \\ \omega_N^{-l*1} \\ \vdots \\ \omega_N^{-l*(N-1)} \end{bmatrix}
$$

Notice that these vectors can be expressed in terms of columns of the DFT matrix.

$$\vec{x}_{lf_0} = c_l \bar{\vec{u}}_l + \bar{c}_l \bar{\vec{u}}_{N-l}$$

If we apply the DFT matrix to our samples now,

$$F_N \vec{x}_{lf_0} = c_l F_N \bar{\vec{u}}_l + \bar{c}_l F_N \bar{\vec{u}}_{N-l}$$

Since $F_N = F_N^T$,

$$
F_N \bar{\vec{u}}_l = \begin{bmatrix} \vec{u}_0 \bar{\vec{u}}_l \\ \vdots \\ \vec{u}_l \bar{\vec{u}}_l \\ \vdots \\ \vec{u}_{N-1} \bar{\vec{u}}_l \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ N \\ \vdots \\ 0 \end{bmatrix}
$$

This is is just a multiple of the lth unit vector $\vec{e}_l$. Thus

$$\vec{X}_{lf_0} = F_N \vec{x}_{lf_0} = N c_l \vec{e}_l + N \bar{c}_l \vec{e}_{N-l}$$

$$
\vec{X}_{lf_0} = \begin{bmatrix} 0 \\ \vdots \\ N c_l \\ \vdots \\ N \bar{c}_l \\ \vdots \\ 0 \end{bmatrix}
$$

This means that if our signal is one of our harmonic frequencies, we can identify which frequency based on the entries in the DFT basis. Accordingly, if our

signal is a sum of our harmnoic frequencies, then

$$x(t) = \sum_{i=0}^{M} A_i cos(2\pi(if_0) + \theta_i)$$

$$\vec{x} = \sum_{i=0}^{M} \vec{x}_i$$

$$\vec{X} = \sum_{i=0}^{M} \vec{X}_i$$

This means we can decompose whatever signal we want into the harmonic frequencies.

7.3 Fourier Series

The premise upon which the Fourier Series is built is that any periodic waveform can be written as a sum of harmonically related frequencies.

Definition 45 *A function $f(t)$ is periodic iff $f(t+T) = x(t) \; \forall t, T \in \mathbb{R}$*

A function which is composed of a periodic function will also be periodic.

Theorem 8 *If $y(t) = f(x(t))$, and $x(t)$ is T-periodic, then $y(t)$ is also T-periodic*

7.3.1 Building the Fourier Series

Sine and Cosine are both periodic function, so it seems intuitive, even if strange, that we can write any function as a sum of them. To begin developing the Fourier Series, we will assume this to be true without going into any proof.

To begin, let $x(t)$ be a T-periodic funtion. Using our assumption, we can write

$$x(t) = B_0 + B_1 cos(2\pi(f_0)t + \theta_1) + B_2 cos(2\pi(2f_0)t + \theta_2) + ...$$

Since $cos(x) = \frac{e^{jx} + e^{-jx}}{2}$, we can convert this to complex form.

$$x(t) = A_0 + A_1 e^{j2\pi f_0 t} + A_2 e^{j2\pi 2f_0 t} + ... + \bar{A}_1 e^{-j2\pi f_0 t} + \bar{A}_2 e^{-j2\pi 2f_0 t} + ...$$

This gives us following infinite sum.

$$x(t) = \sum_{-\infty}^{\infty} A_i e^{j2\pi i f_0 t}$$

This is the Fourier series of our periodic signal.

7.3.2 Finding A_i

We now need an easy way to find the coefficients A_i in our summation. To get the kth coefficient, multiply both sides of the equation by $e^{-j2\pi k f_0 t}$

$$e^{-j2\pi k f_0 t}x(t) = \sum_{-\infty}^{\infty} A_i e^{j2\pi i f_0 t}e^{-j2\pi k f_0 t} = \sum_{-\infty}^{\infty} A_i e^{j2\pi (i-k) f_0 t}$$

Now we can integrate both sides of the equation over our period T.

$$\int_0^T e^{-j2\pi k f_0 t}x(t)dt = \sum_{-\infty}^{\infty} A_i \int_0^T e^{j2\pi (i-k) f_0 t}dt$$

If $i = k$,

$$\int_0^T e^{j2\pi (i-k) f_0 t}dt = \int_0^T e^{j2\pi (k-k) f_0 t}dt = T$$

if $i \neq k$

$$\int_0^T e^{j2\pi (i-k) f_0 t}dt = \frac{1}{j2\pi (i-k) f_0}\left[e^{j2\pi (i-k) f_0 T} - 1\right] = 1 - 1 = 0$$

This means

$$e^{-j2\pi k f_0 t}x(t) = \sum_{-\infty}^{\infty} A_i e^{j2\pi (i-k) f_0 t} = TA_k$$

Therefore

$$A_k = \frac{1}{T}\int_0^T e^{-j2\pi k f_0 t}x(t)dt$$

7.3.3 Truncated Fourier Series

Since the Fourier Series requires an infinite number of terms, it's more useful to be able to truncate it. If we change the bounds of the sum from $[-\infty, \infty]$ to $[-M, M]$ for $M \in \mathbb{R}$, then we get the truncated Fourier Series.

$$\hat{x}(t) = \sum_{-M}^{M} A_i e^{j2\pi (i f_0)t}$$

We can do this because if we want a particular maximum error ϵ, we can choose our M such that $|\hat{x}(t) - x(t)| < \epsilon$. Also, physical systems generally don't have incredibly high frequencies, so we can safely truncate the series because their A_k will be small.