ECE 310

Fall 2025

Review Notes for ECE 310 (Digital Signal Processing)

This is in no way comprehensive.

Aniketh Tarikonda

Contents

1 Midterm 1

1.1 DT vs. CT signals

ECE 210 dealt primarily with CT (continuous time) signals. ECE 310 deals with DT (discrete time) signals.
Signals:

- Continuous-domain (analog) $\rightarrow x(t)$ for $(t \in \mathbb{R})$
- Discrete-domain (digital) $\rightarrow x[n]$ for $(n \in \mathbb{Z})$

Notation is technically important, $x[n]$ refers to a singular sample at n. A signal is represented with $\{x[n]\}_n$ for $n \in \mathbb{Z}$, and a system can be represented as $\{y[n]\} = S\{x[n]\}$ for $n \in \mathbb{Z}$. However, I really cannot be bothered to do this, and most problems don't either.

1.2 LTI/LSI Systems

Linear Time Invariant or Linear Shift Invariant (same thing).

- Linear: Superposition of inputs has a corresponding superposition of outputs. Stuff like scaling is preserved.
- Time/Shift Invariant: Time shift in the input results in an equal time shift in the output.

Most of the fancy theorems in this class rely on systems being LTI.

Determining linearity is relatively straightforward (IMO) but time-invariance isn't (again, IMO). But generally the approach seems to be plugging in a time shift n_o into the input. Then apply the same time shift in the output, and see if the two expressions match.

Trivial Example: $y[n] = x[n^2]$

$x[n]$ transforms $x[n] \rightarrow x[n - n_o]$. Thus, $y[n] = x[n^2 - 2nn_o + n_o^2]$. Applying the same time shift to the output, we get $y[n - n_o] = x[n^2 - n_o]$. These are not the same expression, and thus this system is time-variant.

1.3 The Delta Function

Similar to the delta function from ECE 210. Except its not an infinite spike.

$\delta[n] := \begin{cases} 1 \text{ if } n{=}0 \\ 0 \text{ otherwise} \end{cases}$

We can represent a DT signal as a superposition of scaled and shifted delta functions. For example:

- $x[n] = \sum_{k \in \mathbb{Z}} x[k]\delta[n - k]$
- $y[n] = \sum_{k \in \mathbb{Z}} x[k]S\{\delta[n - k]\}$

1.4 Convolution

Thankfully we don't have to do 3D tiled convolution kernels for this class.

In ECE 210, we use convolution to "apply" an impulse response to an input CT signal. In this class its more or less the same, except its with DT signals.

Discrete convolution is defined as the following:
$y[n] = (x * h)[n] = \sum_{k\in\mathbb{Z}} x[k]h[n-k] = \sum_{k\in\mathbb{Z}} h[k]x[n-k]$

Some useful properties of convolution:
- start point = (start of $x[n]$)(start of $h[n]$)
- end point = (end of $x[n]$)(end of $h[n]$)
- start index = (start index of $x[n]$) + (start index of $h[n]$)
- end index = (end index of $x[n]$) + (end index of $h[n]$)
- Convolution is associative, distributive, commutative (and linear)

1.5 LCCDEs

Stands for Linear Constant Coefficient Difference Equations, and is a popular way to represent LTI/LSI systems. There are two ways of solving LCCDE's: guess-and-check (painful) and Z-transforms (not painful).

General form of LCCDEs:

$$y[n] = \sum_{i=1}^{K} b_i y[n-i] + \sum_{j=0}^{M} c_i x[n-j] \quad \text{for } \{K, M \in \mathbb{Z} \mid 0 \leq K < \infty \text{ and } 1 \leq M < \infty\} \tag{1}$$

I may be a little too ECE 210-pilled, but I guess you can think of the left summation as the Zero-Input terms, and the right summation as the Zero-State terms.

1.5.1 FIR/IIR Systems

FIR (Finite Impulse Response) systems occur when you have LCCDEs with $K = 0$ (no feedback terms). IIR (Infinite Impulse Response) systems have $K > 0$.

1.6 Z-Transforms

Motivation for Z-Transforms: Can we find a class of signals which do not change shape once passed through an LTI/LSI system?

The Z-Transform $X(z)$ of a DT signal $x[n]$ is defined as:

$$X(z) = \sum_{n=-\infty}^{\infty} x[n]z^{-n} \text{ where } z \in \mathbb{C} \tag{2}$$

The ROC (region of convergence) is the range of values in the z-domain in which the Z-transform converge. **Any particular Z-transform may have multiple ROCs which correspond to different inverse Z-transforms.**

A Few Reoccuring Z-Transforms:
- $\alpha^n u[n] \rightarrow \frac{1}{1-\alpha z^{-1}}$ (Right-handed signal, ROC: $|z| > \alpha$)
- $-\alpha^n u[-n-1] \rightarrow \frac{1}{1-\alpha z^{-1}}$ (Left-handed signal, ROC: $|z| < \alpha$)
- $\delta[n-m] \rightarrow z^{-m}$

A Few Reoccuring Z-Transform properties
- Linearity
- Time Shifting: $x[n-k] \rightarrow z^{-k}X(z)$
- Convolution: $x_1[n] * x_2[n] \rightarrow X_1(z)X_2(z)$. Note: this is a reoccuring property of FTs and LTs as well. Pretty sure thats why a lot of convolution algo's use FFT so that convolution becomes an $O(n\log(n))$ operation

Due to the convolution property, we can express LTI systems in the Z-domain as follows:

$$y[n] = (x * h)[n] \rightarrow Y(z) = X(z)H(z) \tag{3}$$

Alternatively, we can write $H(z) = \frac{Y(z)}{X(z)}$ and inverse Z-transform to recover the impulse response $h[n]$. This is probably the best way of determining the impulse response for LCCDEs (for now, at least).

Inverse Z-transforming often devolves into a lot of partial fraction decomposition, so review that.

1.7 Causality

Pretty much the same as introduced in ECE 210.

Output depends solely on current/past inputs $\rightarrow$ **Causal**
Output depends on future inputs $\rightarrow$ **Not Causal**

An anticausal system relies **solely** on future inputs.

A system is causal if impulse response $h[n] = 0$ for $n < 0$.

Both causal and non-causal systems have their merits. Causal systems are used often in real-time systems. Example application of non-causal systems would be in image post-processing, and (generally) signal processing on data which has been stored in some memory unit.

1.8 BIBO Stability

A system is BIBO (Bounded-Input Bounded-Output) stable if for some $\{y[n]\}_n = S\{x[n]\}_n$, whenever $|x[n]| \leq B_{\text{in}} < \infty$, it holds that $|y[n]| \leq B_{\text{out}} < \infty$.

Because of convolution properties, a finite length impulse response is indicative of a BIBO stable system.

What if $h[n]$ is infinite-length? Then it must converge.
$S\{h[n]\}$ is BIBO stable iff $\sum_{k=-\infty}^{\infty} |h[n]| < \infty$.

1.8.1 Z-domain and Stability

In general, a system with transfer function $H(z)$ and associated ROC_H is stable if it contains $|z| = 1$ (the unit circle).

With problems with multiple terms within the transfer function, the ROC is at least the intersection of all terms. For problems that ask to find a bounded input which will result in a bounded output for a non-bounded system, use pole-zero cancellation. For problems that ask to find a bounded input which results in an unbounded output, try to make divergent terms non-zero (delta function does the trick usually).

An LTI system is **marginally stable** if its ROC is open at the unit circle $|z| = 1$.

Define: $x[n] = a^n u[n]$ and $h[n] = b^n u[n]$ where $|a| = |b| = 1$. We can thus expand $a = e^{j\varphi}$ and $b = e^{j\theta}$, respectively. The system output $y[n] = \sum_{k=-\infty}^{\infty} x[n-k]h[k] = \sum_{k=-\infty}^{\infty} e^{j\varphi(n-k)} e^{j\theta(k)} u[n-k]u[k]$. We factor out $\exp(j\varphi n)$ and simplify the summation bounds to rewrite this as:

$y[n] = e^{j\varphi n} \sum_{k=0}^{n} e^{jk(\theta-\varphi)} = e^{j\varphi n} \left(\frac{1-e^{j(\theta-\varphi)(n+1)}}{1-e^{j(\theta-\varphi)}}\right) u[n]$.

When $\varphi = \theta$, the indetermant expression can be written as:

$y[n] = e^{j\varphi n} \sum_{k=0}^{n} (1) = (n+1)e^{j\varphi n} u[n]$.

We can see this expression is unbounded due to the $(n+1)$ term.

When $\varphi \neq \theta$, y[n] oscillates, but remains bounded.

Thus, in a marginally stable system, only inputs that match at least one unit circle pole of the system will produce unbounded outputs. Otherwise, it will be bounded.

2 Midterm 2.

2.1 CTFT and DTFT

Continuous-Time Fourier Transform (CTFT): $\{x(t)\}_{t\in\mathbb{R}} \leftrightarrow \{X_c(\Omega)\}_{\Omega\in\mathbb{R}}$ where:

$$X_c(\Omega) = \int_{-\infty}^{\infty} x(t)e^{-j\Omega t}d\Omega \tag{4}$$

Discrete-Time Fourier Transform (DTFT): $\{x[n]\}_{n\in\mathbb{Z}} \leftrightarrow \{X_d(\omega)\}_{\omega\in\mathbb{R}}$ where:

$$X_d(\omega) = \sum_{-\infty}^{\infty} x[n]e^{-j\Omega n} \tag{5}$$

Important Things to keep in mind about **both** CTFT and DTFT:
- CTFT converts a continuous input into a continuous output. DTFT converts a discrete input into a continuous output.
- DTFT is a special case of the Z-transform, CTFT is a special case of the Laplace transform.
- Both DTFT/CTFT provide a representation of a signal as a linear combination of complex exponentials (which can be thought of as 2D vectors in the complex plane)

DTFT properties
- $X_d(\omega)$ is 2π-periodic
- **If $\{x[n]\}_{n\in\mathbb{Z}}$ is real-valued**, $|X_d(-\omega)| = |X_d(\omega)|$ and $\angle X_d(-\omega) = -\angle X_d(\omega)$ (Magnitude is symmetric, phase is anti-symmetric)
- Convolution in n-domain is multiplication in the ω-domain. Thus, LTI systems have a corresponding frequency response $H_d(\omega)$
- Most CTFT properties have an analogue for the DTFT
- Inverse DTFT: $x(t) = (2\pi)^{-1}\int_{-\pi}^{\pi} X_d(\omega)e^{j\omega n}d\omega$

2.1.1 Frequency Response

For any **stable** LSI system, we have $H_d(\omega) = H(z)|_{z=\exp(j\omega)}$ where $H(z)$ denotes the transfer function.

If we have an input signal $x[n] = \exp(j\omega_0 n)$, and pass it through a LSI system with frequency response $H_d(\omega)$, the output $y[n] = H_d(\omega_0)\exp(j\omega_0 n)$ In other words, signals of the form $A\exp(j\omega_0 n)$ are eigenfunctions of LSI systems.

Similarly, $x[n] = \cos(\omega_0 n + \varphi) \to |H_d(\omega_0)|\cos(\omega_0 n + \varphi + \angle X_d(\omega_0))$

2.2 Sampling and ADC/DAC

Idea: We have a **continuous** signal $x_c(t)$, and every T seconds, we take a sample of it. We then end up with a **discrete** signal $x[n] = x_c(nT)$. Our sampling period is T, and our sampling frequency is $f_s = \frac{1}{T}$.

Question: What is the relationship between $X_c(\Omega)$ and $X_d(\omega)$? In other words, what effect does sampling have in the Fourier domain?

Quick lil' derivation:

By definition, we know $x[n] = x_c(nT) = (2\pi)^{-1} \int_{-\infty}^{\infty} X_c(\Omega) e^{j\Omega(nT)} d\Omega$. Furthermore, $x[n] = (2\pi)^{-1} \int_{-\pi}^{\pi} X_d(\omega) e^{j\omega n} d\omega$.

Thus, $\int_{-\infty}^{\infty} X_c(\Omega) e^{j\Omega(nT)} d\Omega = \int_{-\pi}^{\pi} X_d(\omega) e^{j\omega n} d\omega$. Substituting $\omega = \Omega T$, we get:

$$\left(\frac{1}{T}\right) \int_{-\infty}^{\infty} X_c\left(\frac{\omega}{T}\right) e^{j\omega n} d\omega = \int_{-\pi}^{\pi} X_d(\omega) e^{j\omega n} d\omega \tag{6}$$

Lastly, because X_c is 2π-periodic, we can expand the LHS as the following.[1]

$$\left(\frac{1}{T}\right) \int_{-\infty}^{\infty} X_c\left(\frac{\omega}{T}\right) e^{j\omega n} d\omega = \frac{1}{T} \sum_{k \in \mathbb{Z}} \int_{-\pi+2\pi k}^{\pi+2\pi k} X_c\left(\frac{\omega + 2\pi k}{T}\right) e^{j\omega n} d\omega \tag{7}$$

Using the change of variables $\omega' = \omega - 2\pi k$, we can further simplify the bounds of the integral expression. Matching the integrands of this expanded expression and the DTFT integral, we determine:

$$X_d(\omega) = \frac{1}{T} \sum_{k \in \mathbb{Z}} X_c\left(\frac{\omega + 2\pi k}{T}\right) \tag{8}$$

2.2.1 Aliasing and Nyquist Frequency

Suppose we have a **band-limited** signal whose Fourier trnasform is $X_c(\Omega)$. Thus, $X_c(\Omega) = 0$ for $|\Omega| > B$, where B is the bandwidth (highest-frequency) of the signal.

If we sample this signal, we get that $X_d(\omega) = \frac{1}{T} \sum_{k \in \mathbb{Z}} X_c\left(\frac{\omega + 2\pi k}{T}\right)$. Graphically, this looks like an infinite series of scaled & shifted versions (aliases) of $X_c(\Omega)$, each of which centered at some $2\pi k$. To find where the bandwidth gets "mapped" to after sampling, we use the relation $\omega = \Omega T$ where $\Omega \to B$.

We know B is highest-frequency in our original signal, and has units rad/s, so $B = 2\pi f_B$. We also know $T = \frac{1}{f_s}$. Thus, we can express ω_B (the bandwidth of the signal **after** sampling) as $\omega_B = BT = 2\pi \frac{f_B}{f_s}$

Each of the aliases is centered at $2\pi k$ for $k \in \mathbb{Z}$. They have the potential to overlap if the bandwidth of each alias in the ω-domain exceeds π (think about it graphically). This overlap is called **aliasing**, and it causes us to lose information about the original signal.

[1] Note: Omitted the shift of ω in the complex exponential term because a shift by $2\pi k$ in the exponent corresponds to a 360 degree rotation (thus leaving the complex term unchanged).

Thus, if we don't want aliasing to occur, the bandwidth in the ω-domain must be less than or equal to π.

Equivalenty, $\omega_B = 2\pi\frac{f_B}{f_s} \leq \pi \rightarrow \boldsymbol{f_s \geq 2f_B}$

This condition is called the Nyquist Criterion, and the **Nyquist Frequency** is $2f_B$, which is the lowest sampling frequency in which you will observe no aliasing.

In general there are two ways to resolve aliasing for ADC conversion:
- Increase f_s to at least Nyquist Frequency.
- Use a low-pass filter (LPF) on $X_c(\Omega)$ to artificially reduce f_B.[2]

2.2.2 Ideal ADC and DAC

The process of sampling converts a CT signal to a DT one, and so that process is called Analog-to-Digital Conversion (ADC).

Ideal Digital-to-Analog conversion basically consists of two steps:
- Apply a LPF to remove the higher-frequency aliases, and just retain the frequency spectrum centered at $\omega = 0$.
- Properly scale/resize the resulting frequency spectrum to obtain that of $X_c(\Omega)$

2.3 DFT

The DTFT is continuous in the Fourier domain, wheras the DFT is discretized in the Fourier domain.

Given a length-N signal $\{x[n]\}_{n=0}^{N-1}$, the N-point DFT $\{X[k]\}_{k=0}^{N-1}$ is defined:

$$X[k] = \sum_{n=0}^{N-1} x[n]e^{-j\frac{2\pi nk}{N}} = \sum_{n=0}^{N-1} x[n]W_N^{-kn} = \sum_{n=0}^{N-1} x[n]e^{-j\omega n}|_{\omega=\frac{2\pi k}{N}} \tag{9}$$

Won't come up on the midterm (probably), but the DFT allows for some really cool matrix/vector representations in Fourier Analysis, as it is a linear transformation.

2.3.1 DFT properties

DFT properties are pretty similar to DTFT, albeit with some modifications. Here are a few of the important ones:
- DFT is periodic by N. Thus, $X[k + aN] = X[k]$, $a \in \mathbb{Z}$
- $x[\langle n - m\rangle_N] \rightarrow \exp\big(-j\frac{2\pi mk}{N}\big)X[k] = W_N^{-km}X[k]$
- $x[n] \circledast h[n] = X[k]H[k]$
- $X[\langle k - m\rangle_N] = \exp\big(j\frac{2\pi mn}{N}\big)x[n]$

[2] This comes at a cost of losing information about the original signal. However, often times the information lost by low-passing is much less than the potential information lost by aliasing. In other scenarios however, we can actually just allow aliasing to occur, and then use a digital LPF in the ω-domain to select the range of frequencies we need. There was a homework problem about this.

2.3.2 Zero-Padding

Idea: if we 'artificially' increase the length of the input signal by adding zeros to the end of it, the DTFT is unchanged. However the DFT, which has the same number of samples as the input signal, increases in length. More specifically, it takes more "samples" of the DTFT, yielding a higher resolution of the frequency spectrum of the original signal.

This is important to keep in mind when doing spectral analysis, and we need an increased resolution to better identify peaks.

2.3.3 Windowing

Windowing is a method to attenuate spectral leakage, which is a consequence of the DFT.

Windowing is just multiplication in the time domain. The two windows we use for this class are Rectangular (boxcar) windows and Hamming windows. Rectangular window has a narrower main-lobe frequency response, but larger side-lobes. The Hamming window has a wider main-lobe, but significantly attenuated side-lobes in its frequency response.

2.3.4 Spectral Analysis

- Identify peaks (sinusoidal components) within a finite-length signal.
- Given some finite-length superposition of sinusoidal components can we identify peaks?

We can use the DFT to do this analysis, but we must be careful about spectral leakage. For example, say we have signal $x[n] = A\cos(\omega_0 n)$, defined for $0 \leq n \leq N-1$.

Given ω_0, we know the period $N_0 = \frac{2\pi}{\omega_0}$. Thus, the number of full cycles captured in the n-domain is $\frac{N}{N_0} = \frac{N\omega_0}{2\pi}$. If this value is an integer, then the DFT will capture this sinusoid perfectly (this can be verified by manually doing the DFT, and if done right all of the values of k which don't correspond with peaks should evaluate to 0).

If the number of full cycles is not an integer, we will have a periodic discontinuation at the end. The DFT assumes all signals to be periodic, and so it will introduce new (nonzero) components because of the discontinuation.

Challenges to Spectral Analysis

- If frequency components are to close together, we may not be able to distinguish them -> we need higher resolution
- If amplitudes vary greatly, you could have a sidelobe of one component mask the mainlobe of another, smaller frequency component.

3 Final

3.1 FFT (Fast Fourier Transform)

- Important (obviously for FT), but other indirect applications of FT, like fast convolution.
- Motivation: Naive N-point DFT is $O(N^2)$. Example:

```
for i in range(0, N):
    for j in range(0, N):
        x[i] += x[j] * np.exp(-2 * pi * k * n * 1j/ N)
```

- FFT is $O(N \log N)$, uses divide-and-conquer, first breaking it down to a small unit, and then recombining on the way up to recover the DFT.
- We break the signal down into even and odd index components, keep on doing this repeatedly.
- Quick derivation of FFT recombination:

$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{-kn}$ by definition of DFT. We break this up as:

$$X[k] = \sum_{n=0}^{\frac{N}{2}-1} x[2n] W_N^{-k2n} + \sum_{n=0}^{\frac{N}{2}-1} x[2n+1] W_N^{-k(2n+1)} \tag{10}$$

We can define $a[n] = x[2n]$ and $b[n] = x[2n+1]$. We can also make the simplification $W_N^{-2kn} = \exp\left(j \times 2\pi \times \left(-k \frac{n}{\frac{N}{2}}\right)\right) = W_{\frac{N}{2}}^{-kn}$. Thus, we can simplify the above expression as:

$$\begin{aligned} X[k] &= \sum_{n=0}^{\frac{N}{2}-1} a[n] W_{\frac{N}{2}}^{-kn} + \left(\sum_{n=0}^{\frac{N}{2}-1} b[n] W_{\frac{N}{2}}^{-kn} \right) W_N^{-k} \\ &= A[k] + W_N^{-k} B[k] \end{aligned} \tag{11}$$

Notably, this only covers the first half of $X[k]$, to find the recombination formula for this upper half, we shift everything by $\frac{N}{2}$. However, A, B are $\frac{N}{2}$-periodic, so the equation reduces to: $X_{\text{upper}}[k] = A[k] - W_N^{-k} B[k]$.

- There are many graph representation of FFT, like the radix 2 "Butterfly" diagram.

3.1.1 Circular Convolution

- Direct application of FFT–fast convolution.
- Circular convolution definition: $y[n] = \sum_{n=0}^{N-1} h[m] x[\langle k - m \rangle_N]$
- Circular convolution in time domain corresponds with multiplication in the N-point DFT domain.

- Thus, instead of performing circular convolution, we can compute two FFTs, multiply them, and then take the inverse FFT.
- However, we must resolve the artifacts of N-periodicity in the DFT. Thus, we zero pad **both** our signals to the same length $N + K - 1$.

3.2 Filter Design

- Ideal Filters are not realistic to implement. These ideal filters consist of rect functions in the frequency domain, which implies sinc functions in the time domain.
- However, sinc functions are:
  - non causal
  - infinite length
  - not BIBO-stable
- Solution, truncate the sinc function. Ex: $h_{\text{LPF}}[n] = \frac{\omega_c}{\pi}\,\text{sinc}(\omega_c n), n \in \{0, 1, ..., N-1\}$
- Group delay: $\tau_{\text{gd}} = -\frac{d}{d\omega}\angle H_d(\omega)$.
- Linear phase $\rightarrow$ uniform group delay $\rightarrow$ all frequencies get delayed by the same amount

3.2.1 Linear Phase Filter Types

- Type 1 - even symmetric around some point, odd length. Could be any canonical filter type.
- Type 2 - even symmetric around some point, even length. Could be LP or BP
- Type 3 - odd (anti) symmetric around some point, odd length. Could be BP.
- Type 4 - odd symmetric around some point, even length. Could be BP or HP.

3.2.2 Linear Phase versus Generalized Linear Phase

- Linear Phase: $\angle H_d(\omega) = -\alpha\omega$
- Generalized Linear Phase: $\angle H_d(\omega) = -\alpha\omega + \beta(\omega)$ where $\beta(\omega)$ only adds $\pm\pi$ phase shifts whenever the amplitude becomes negative.

3.3 Upsampling/Downsampling

3.3.1 Downsampling

- You're taking the D'th sample of the original signal (think of it as sampling twice). Your sampling period is then D times that of the original signal.
- Downsampling by a factor of D yields the DTFT:

$$Y_{d(\omega)} = \frac{1}{D}\sum_{k=0}^{D-1} X_{d(\frac{\omega-2\pi k}{D})} \tag{12}$$

- In the DTFT domain, downsampling stretches out values by a factor of D. Thus, we need to be careful about aliasing.
- Ideally, we apply some anti-aliasing filter $\omega_c = \frac{\pi}{D}$ **before** downsampling.

3.3.2 Upsampling

- Take more samples of the original signal by adding $U - 1$ zeros after each sample
- Period of upsampled signal decreases by factor of U
- In the DTFT domain, $Y_{d(\omega)} = X_{d(U\omega)}$. Much cleaner than downsampling, no amplitude changes.
- Upsampling brings aliases closer together, potentially within the $[-\pi, \pi]$ range. Thus, we need an interpolation filter.
- Interpolation filter is basically a LPF with $\omega_c = \frac{\pi}{U}$. this filter gets applied **after** upsampling.

3.3.3 Rate Conversion by Non-Integer values

- You can accomplish this by both upsampling and downsampling (and using their respective filters, of course)
- Upsample first! Think of this as a data loss question, you're gonna lose more data downsampling first and interpolating compared to vice versa.

3.4 Practical ADC/DAC, Zero-Order Holds (ZOH)

- sinc interpolation isn't the most feasible for real-time systems, we need something simpler $\rightarrow$ zero-order hold (interpolation using a 0-order polynomial)
- when doing D/A, the compensation filter will not be ideal, and so we can upsample to increase the allocated bandwidth for the filter, which in terms allows a more accurate filter shape within passband.
- Reconstruction Filter Transition Bandwidth: $\frac{2\pi}{T}(U - 1)$. Larger U means reconstruction filter becomes easier to implement with analog circuitry (and so we wrap back to ECE 210).