EE301L Signals and Systems – Computer Problems, Labs, Projects

April 22, 2021

1 Computer Problem/Lab: Getting Familiar with Python by Replicating `signal.scipy.lfilter`

Write a subroutine for a general ARMA filter with:

- inputs:
  - `b`: an array of MA coefficients
  - `a`: an array of AR coefficients
  - `x`: an array of inputs
- outputs:
  - `y`: array of outputs for the filter.

This should replicate the functionality of `scipy.signal.freqz`. Some useful Python routines for this implementation may be `np.roll` and `np.dot`. Check your results using the following code snippet (i.e., your implementation vs `scipy.signal.freqz` – you should check other cases, but everybody should submit results for this special case.

```
1 ##########################################################
2 ### toy filtering example
3 ##########################################################
4 ### some toy synthetic data (noisy sine wave)
5 nu0 = 0.025
6 n = np.arange(0, int(3/nu0))
7 x = np.sin(2 * np.pi * nu0 * n) + np.random.normal(0, 0.5, len(n))
8 b, a = signal.butter(3, 2 * (1.5 * nu0) )
9 y = filter_and_plot(b, a, x)
```

2 Computer Problem/Lab: Exploring the Frequency Response via a Two-Tap DT Filter

Consider a moving average filter with a single delay element

$$y[n] = b_0 x[n] + b_1 x[n-1] \tag{1}$$

where b_0 and b_1 are real numbers.

1. For the MA filter in (1), show that if $x[n] = \exp(j2\pi\nu_0 n)$, then $y[n] = H(\nu_0)\exp(j2\pi\nu_0 n)$ where $H(\nu_0)$ is a complex gain that depends on the input frequency ν_0. Find $H(\nu_0)$ and determine $|H(\nu)|$ and $\angle H(\nu)$. This notation is to get you to begin to consider $H(\nu)$ as a complex function of ν – *i.e.,* it is the complex gain of this system to input $\exp(j2\pi\nu n)$ as a function of ν.

2. Is $H(\nu)$ a Hermitian Symmetric function of ν?

3. Plot magnitude and phase of $H(\nu)$ for the following two cases (you may wish to plot the magnitude on a dB scale and also look at it on a linear scale):

   (a) $b_0 = b_1 = 0.5$

   (b) $b_0 = 0.5$, $b_1 = -0.5$

   (c) How would you characterize these two filters – *i.e.,* what frequencies do they pass?

4. Show that for the MA filter in (1) that when the input is $x[n] = \cos(2\pi\nu_0 n)$ the output is

$$y[n] = |H(\nu_0)|\cos(2\pi\nu_0 n + \angle H(\nu_0)) \tag{2}$$

5. Write a script to pass the following input signals through the two MA filters in part 3:

$$x[n] = \cos(2\pi\nu_0 n)$$
$$x[n] = \sin(2\pi\nu_0 n)$$

   for each consider $\nu_0 \in \{0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.7, 0.9\}$.

   Using the reasoning from part 4 above, the output for the above cases should be a sinusoid at the same frequency as the input with an amplitude gain and phase shift: measure these from your script by comparing $x[n]$ and $y[n]$ and compare against what you expect from the analysis above.

   Note: you may consider how to automate the measurement of the amplitude and phase differences between two sinusoids of the same frequency. This will be discussed in the Lab section with the TA. Similarly, initial transient effects from starting the filter at rest will be discussed in Lab.

6. Consider an ARMA filter with

$$[a_0, a_1, a_2] = [1.0, -1.1429805, 0.4128016] \tag{3}$$
$$[b_0, b_1, b_2] = [0.06745527, 0.13491055, 0.06745527] \tag{4}$$

   Repeat the numerical experiments in part 5 above for this ARMA filter. Compare the measured magnitude and phase associated with sinusoidal inputs to the "frequency response" as computed by `scipy.signal.freqz`. Does the "frequency response" value at the frequency ν_0 correspond to the observed gain and phase shifts?

   Note: Check the webpage for `scipy.signal.freqz`, but also see the notebook examples distributed in lecture for example usage.

3 Computer Problem/Lab: Image Filtering

In this problem you will learn how to read and write grayscale images in Python and about two-dimensional convolution and MA filters.

For a one-dimensional, discrete time (index) signals, convolution is defined as

$$y[n] = x[n] * h[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k]$$

For two-dimensional signals (e.g., images and image filters) convolution is

$$y[i][j] = x[i][j] * h[i][j] = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} x[m][n]h[i-m][j-n]$$

Note that the index set here is not time, it is spatial location in the image and is there for indexed by two integers – *e.g.,* i for horizontal and j for vertical.

1. Find $y[i][j] = x[i][j] * h[i][j]$ for the signals shown below. Plot a heat-map of the result using `matplotlib.pyplot.matshow`.

$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\iddots$
$\dots$	0	0	0	0	0	0	0	$\dots$
$\dots$	0	$x[-2][1]=1$	$x[-1][1]=1$	$x[0][1]=1$	$x[1][1]=1$	$x[2][1]=1$	0	$\dots$
$\dots$	0	$x[-2][0]=1$	$x[-1][0]=1$	$x[0][0]=1$	$x[1][0]=1$	$x[2][0]=1$	0	$\dots$
$\dots$	0	$x[-2][-1]=1$	$x[-1][-1]=1$	$x[0][-1]=1$	$x[1][-1]=1$	$x[2][-1]=1$	0	$\dots$
$\dots$	0	0	0	0	0	0	0	$\dots$
$\iddots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

and

$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\iddots$
$\dots$	0	0	0	0	0	$\dots$
$\dots$	0	$h[-1][1]=1/4$	$h[0][1]=1/2$	$h[1][1]=1/4$	0	$\dots$
$\dots$	0	$h[-1][0]=1/2$	$h[0][0]=1$	$h[1][0]=1/2$	0	$\dots$
$\dots$	0	$h[-1][-1]=1/4$	$h[0][-1]=1/2$	$h[1][-1]=1/4$	0	$\dots$
$\dots$	0	0	0	0	0	$\dots$
$\iddots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Note: In both cases, all values at indices not shown are assumed to be zero and you only need to show the output over the region where it is non-zero.

2. In this part you will filters some images using the following filters: (a) a 2D moving average filter, (b) a Gaussian blur filter, (c) a vertical edge detector, and (d) a horizontal edge detector.

   The $(M \times M)$ moving average filter is given by

$$\mathbf{H}_{MA} = \frac{1}{M^2}\mathbf{1}\mathbf{1}^{\mathrm{t}}$$

   where $\mathbf{1}$ is an $(M \times 1)$ all ones vector.

The Guassian blur has

$$\mathbf{H}_{GB}[i][j] = \frac{K}{2\pi\sigma^2} \exp\left(\frac{-(i^2+j^2)}{2\sigma^2}\right)$$

This is a 2D bell-shaped curve with the term σ is selected to control the width. With $M = 2L+1$, a reasonable choice is $\sigma = L/2$ – *i.e.,* this makes the edge fo the filter in the horizontal and vertical directions the 2σ points. The constant K is selected so that $\sum_{i,j} h[i][j] = 1$ – *i.e.,* the DC gain of the filter is 1.

The vertical edge detector is

$$\mathbf{H}_V = \begin{bmatrix} \mathbf{-1} & \mathbf{0} & \mathbf{+1} \end{bmatrix}$$

which is a $(M \times 3)$ matrix.

The horizontal edge detector filter kernel is the transpose of the vertical edge detector kernel: $\mathbf{H}_H = \mathbf{H}_V^{\mathrm{t}}$.

Use $M = 2L+1$ with $L = 9$ for each of these filters to the following test images:

- Grayscale test image
- airport
- male

For each test image provided, use `scipy.signal.convolve2d` (or equivalent in Matlab) to apply the filter. Provide a side-by-side comparison of the input and output images and discuss, qualitatively, the performance of each filter on the images.

Methods for reading, writing and manipulating images in Python will be discussed in lab.

4 Computer Problem/Lab: Filtering a Fourier Series

In Problem 3.4, you explored the Fourier series and plotted truncated FS sums. In this problem, we consider filtering these signals using an LTI system. Specifically, an n^{th} order Butterworth low-pass filter is defined by

$$|H(j\omega)| = \frac{1}{\sqrt{1+\left(\frac{\omega}{\omega_c}\right)^{2n}}}$$

where ω_c is the cut-off frequency.

For even values of n, the frequency response can be written as

$$H(j\omega) = \frac{1}{\prod_{k=\lceil (n-1)/2 \rceil}^{n-1} \left[\left[j\left(\frac{\omega}{\omega_c}\right)\right]^2 - 2\cos\left(\frac{(2k+1)\pi}{2n}\right)\left[j\left(\frac{\omega}{\omega_c}\right)\right] + 1\right]} \tag{5}$$

where $\lceil \cdot \rceil$ is the ceiling operation – *e.g.,* $\lceil 3 \rceil = 3$, $\lceil 3.01 \rceil = \lceil 3.5 \rceil = \lceil 3.99 \rceil = 4$. As a concrete example, for $n = 2$ we have

$$\begin{aligned} H(j\omega) &= \frac{1}{\left[j\left(\frac{\omega}{\omega_c}\right)\right]^2 - 2\cos(3\pi/4)\left[j\left(\frac{\omega}{\omega_c}\right)\right] + 1} \\ &= \frac{1}{\left[j\left(\frac{\omega}{\omega_c}\right)\right]^2 + \sqrt{2}\left[j\left(\frac{\omega}{\omega_c}\right)\right] + 1} \end{aligned}$$

We will also consider a high-pass version of this filter which can be derived using

$$|H_{HP}(j\omega)|^2 = 1 - |H(j\omega)|^2$$

It can be show that the associated $H_{HP}(j\omega)$ can be obtained by using (5) with (ω/ω_c) replaced by (ω_c/ω).

1. Produce magnitude and phase plots for $H(j\omega)$ and $H_{HP}(j\omega)$ for $\omega_c = 2\pi f_c$ and $f_c = 3500$ and $n = 2, 4, 6, 8$. Plot the magnitude in dB – *e.g.,* $10\log_{10}(|H(j\omega)|^2)$ – and use a log scale for ω. This is a so-called Bode plot. Note that you can produce these plots by using complex arithmetic in Python and you do not need to simplify the expression for $H(j\omega)$ for $n = 4, 8$.

   (a) What is the value of $|H(j\omega)|^2$, in dB, for $\omega = 0$, $\omega = \omega_c$, and as $\omega \to \infty$.

   (b) Discuss the behavior of $|H(j\omega)|^2$ at large frequencies as a function of n. How does this manifest itself in the plots?

   (c) Can you determine what the the phase is at $w = 0$ and as $\omega \to \infty$ from the analytical expression? Does this match your plots?

2. In this part we will consider the following Fourier Series

$$\begin{aligned} \text{(a): square wave:} \quad & X_k = \frac{1}{2}\text{sinc}(k/2) - \delta[k] \\ \text{(b): triangle wave:} \quad & X_k = \frac{1}{2}\text{sinc}^2(k/2) - \delta[k] \\ \text{(c): delta train:} \quad & X_k = 1 \end{aligned}$$

   This will make the waveforms have amplitude 0.5 and the will be even functions.

   Plot the input and output of the LPF and HPF above with $n = 2$ and when $T = 1/1000$. For the input use $K = 19$ from problem 3.4 for the truncation of the FS.

   Discuss the results and the relationship between f_c and $f_0 = 1/T$. Also, is using this low-pass filter the same as truncating the Fourier Series? If not, how does it differ? You may also try using a higher cut-off frequency – *e.g.,* $f_c = 8000$.

Note: It is a good exercise to program up the frequency response for the butterworth filter as described above. Some of this is automated in Python, however. For example, the following command will find an equivalent butterworth filter

```
b, a = signal.butter(n, omega_c, 'low', analog=True)
```

Specifically, this will provide two arrays b, and a and

$$H(j\omega) = \frac{\prod_{k=0}^{L_N} b_k (j\omega)^{L_N - k}}{\prod_{k=0}^{L_D} a_k (j\omega)^{L_D - k}}$$

This above command will return a b array of size 1 – divide the a vector by b and you will have the equivalent of (5). The frequency response can then be computed using `signal.freqs`

```
w, h = signal.freqs(b, a)
```

See the example in the documentation for `scipy.signal.butter`.

5 Computer Problem/Lab: Correlational and Matched Filtering

In many problems in communications, signal processing, and radar, it is desired to detect or locate a known signal in noise. For the most common type of noise encountered in engineering (broadband, Gaussian noise or additive white Gaussian noise), the optimal processing for these tasks is provided by performing a signal correlation.

The correlation between $z(t)$ and $p(t)$ is given by

$$r(t) = z(t) \star p(t) = \int_{-\infty}^{\infty} z(\tau) p^*(\tau - t) d\tau = \int_{-\infty}^{\infty} z(\lambda + t) p^*(\lambda) d\lambda$$

5.1 Correlation Properties and Matched Filters

(a) Show that

$$r(t) = z(t) \star p(t) = z(t) * p^*(-t)$$

In other words a correlation between $z(t)$ and $p(t)$ can be obtained by passing $z(t)$ through a LTI system with impulse response $p^*(-t)$. The filter with impulse response $p^*(-t)$ is called a *matched filter* – *i.e.,* a filter matched to $p(t)$.

(b) Consider the case when $z(t) = p(t)$. What is the output of the correlator, $r(t)$? What is $R(f)$ in terms of $P(f)$ for this case? What is $r(0)$? Sketch $r(t)$ for the following cases:

- **rect-pulse** $p(t) = \sqrt{E_p}\text{rect}(t)$
- **duo-binary-pulse** $p(t) = \sqrt{\frac{E_p}{2}}\left[\text{rect}(t + 1/2) - \text{rect}(t - 1/2)\right]$

5.2 Binary Digital Communications

A simplified model for a common binary digital communication signal is

$$z(t) = d\, p(t) + w(t)$$

where $d \in \{-1, +1\}$ is the data bit and $w(t)$ broadband noise. A decision on this bit is made by computing $r(t) = z(t) \star p(t)$, sampling at $t = 0$, and comparing against 0 as shown in Fig. 1. Specifically, the decision is:

$$r(0) \underset{\hat{d}=-1}{\overset{\hat{d}=+1}{\gtrless}} 0$$

where $\hat{d}$ is the decision.

In this part, you will simulate this processing using $f_s = 1/T_s$ samples per second with $f_s \geq 16$ suggested – *i.e.,* you will simulate the continuous time processing using f_s samples for each unit interval of t. This will yield

$$z(nT_s) = d\, p(nT_s) + w(nT_s)$$

or, overloading the notation a bit,

$$z[n] = d\, p[n] + w[n]$$

where $z[n] = z(nT_s)$, $p[n] = p(nT_s)$, and $w[n] = w(nT_s)$.

The noise sequence can be generated by using `np.random.normal(0, sigma, N)` where sigma is

$$\sigma = \sqrt{\frac{N_0}{2} f_s}$$

and N is the length of the array desired.

The performance measure for this system is the bit error rate (BER) which is he fraction of trials (or probability) for which $\hat{d} \neq d$ – *i.e.,* the fraction of trials for which the receiver makes and error. The BER is a function of E_p/N_0 which we often measured in dB – *i.e.,* $10\log_{10}(E_p/N_0)$.

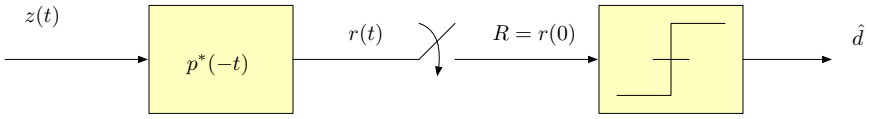

Figure 1: A receiver for the binary digital communications case.

(c) In order to visualize the effect of the matched filter, set the data bit to be $d = +1$, use the rectangular pulse,[1] and generate the noise as described above. Plot $r(t)$ and note the value of this at $t = 0$. This should show the signal component of the matched filter output increasing in magnitude and being disturbed by noise. Do this for no noise, $E_p/N_0 = -3$ dB and $E_p/N_0 = 6$ dB. Repeat this for a few noise and data realizations for each setting of E_p/N_0. Optionally, may also wish to consider a higher signal to noise ratio – *e.g.,* $E_p/N_0 = 12$ dB.

(d) Repeat this experiment K times, except each time generate a data bit randomly using `np.random.choice([-1, +1])`. Specifically, for each of the K trials, generate value for d, the noise, and do the above processing to obtain a decision $\hat{d}$. Compute the BER for $E_p/N_0 = -3$ dB and $E_p/N_0 = 6$ dB. You should choose K to be large enough that you observe ≥ 100 errors.

Optional: Another useful visualization in this case is a histogram of the values of $R = r(0)$ conditioned on the value of d. Specifically, generate a histogram for the cases where $d = -1$ and another histogram for the cases where $d = +1$.

Optional: Produce a BER curve by plotting your BER vs. E_p/N_0 where E_p/N_0 is in dB and the y-axis for BER is on a log scale. In the context of this part of the problem, E_p is the energy per bit so, this is typically denoted by E_b and the plot is vs E_b/N_0.

5.3 Delay Estimation

Suppose that the following is available

$$z(t) = p(t - t_0) + w(t)$$

where t_0 is an unknown delay. A good estimate of t_0 can be obtained by computing $r(t) = z(t) \star p(t) = z(t) * p^*(-t)$ and then finding the value of t that maximizes this quantity. Specifically,

$$\hat{t}_0 = \arg\max_t r(t)$$

You will simulate this using the same technique as in the digital communications problem above – *i.e.,* use the rect-pulse and same method for generating the noise.

(e) Simulate the case where $t_0 = 1.5$. For this delay value, plot the matched filter output $r(t)$ for the cases of no noise, $E_p/N_0 = -3$ dB and $E_p/N_0 = 6$ dB. Repeat this for a few noise and data realizations for each setting of E_p/N_0. In each case, note what the delay estimate value is $\hat{t}_0$.

(f) Repeat this experiment K times – *i.e.,* each time generating the noise, and processing it to obtain a delay estimate $\hat{t}_0$. Compute the mean-squared error (MSE) for this estimate by average $(t_0 - \hat{t}_0)^2$ over the K noise realizations. Do this for $E_p/N_0 = -3$ dB and $E_p/N_0 = 6$ dB using $K > 5000$.

Optional: Produce a histogram of the delay estimate for the E_p/N_0 values considered.

Optional: Produce a plot of the MSE (in dB) vs. E_p/N_0 in dB.

[1] You may choose to use rect$(1/2 - t)$ in place of rect$(-t)$ for the matched filter. This makes the matched filter causal. It delays the output by 1/2, so where we have referred to $r(0)$ here, that will be mapped to $r(1/2)$. It is common to assume a non-casual matched filter for analysis, but when implemented in practice, a delay is inserted to make it causal. Depending on how you do this in Python/Matlab, you may need to make the matched filter causal.

6 Computer Problem/Lab: Windows for FIR Filter Design

There are many examples of infinite (length) impulse response (IIR) filters that do not follow a constant-coefficent, linear difference equation (*e.g.,* ARMA filter). Specific examples include: ideal filters (low-pass, high-pass, band-pass), differentiators, fractional-sample delays, and interpolators/up-samplers. In these examples, we have an ideal impulse response $h[n]$ that is non-zero over the range $n \in \{0, pm1, \pm 2, \ldots\}$. The practical way to implement this filter is to window $h[n]$ to obtain

$$g[n] = w[n]h[n]$$

where $w[n] = 0$ for $|n| > M$. This means that $g[n]$ is $L = 2M + 1$. In examples in lecture, we used a rectangular window which corresponds to simply truncating $h[n]$.

After windowing, the filter impulse response can be delayed to make the resulting FIR causal – *i.e.,* the implemented filter is $g[n - M]$.[2]. Because of this, an alternative notation is to use

$$g[n] = w[n]h[n - M]$$

where $w[n]$ is defined to be zero for $n < 0$ and $n \geq L$. The windows defined below follow this second convention.

6.1 Window Definitions

There are many different windows that have been proposed and are used in practice. Like many engineering design problems, there is no one right answer and, instead, window design is about trade-offs. In this problem, you will consider the following windows. In all cases, $w[n] = 0$ for $n \notin \{0, 1, \ldots L - 1\}$.

- **rect:** This is just the truncation used in the lecture examples:

$$w[n] = 1, \qquad 0 \leq n < L$$

- **Bartlett (Triangle):** This is a triangle shape that decays linearly from its center to zero at the edges:

$$w[n] = \frac{2}{L-1}\left(\frac{L-1}{2} - \left|n - \frac{L-1}{2}\right|\right) = \frac{2(M - |n - M|)}{L-1} \qquad 0 \leq n < L$$

  This window is available in Python as `np.bartlett`

- **Hanning:** This window is available in Python as `np.hanning` and is given by

$$w[n] = \frac{1}{2}\left[1 - \cos\left(\frac{2\pi n}{L-1}\right)\right] \qquad 0 \leq n < L$$

- **Hamming:** This window is available in Python as `np.hamming`

$$w[n] = 0.54 - 0.46\cos\left(\frac{2\pi n}{L-1}\right) \qquad 0 \leq n < L$$

- **Kaiser:** This window is available in Python as `np.kaiser`

$$w[n] = \frac{I_0\left(\beta\sqrt{1 - \left(\frac{2n}{L-1}\right)^2}\right)}{I_0(\beta)} \qquad 0 \leq n < L$$

[2]It is not necessary to have L be odd, but it makes the math a little simpler and it is common to choose L odd in practice.

where β is a parameter that adjusts the window shape significantly. In this problem, we will use $\beta = 2.5$. Setting $\beta = 0$ yields the rect window and the numpy documentation claims that using $\beta = 5$ provides a window similar to the Hamming window while using $\beta = 6$ yields a window similar to the Hanning window. Because of the flexibility of this window with varying beta, it is often used in automated filter design routines such as Matlab's.

- **Vorbis:** Vorbis is an open-source software solution for audio encoding and decoding – *e.g.,* similar to mp3 and AAC. The Vorbis window is used in this standard and the related voice codec called Opus. The Vorbis window is defined as

$$w[n] = \sin\left[\frac{\pi}{2}\left(\sin^2\left(\frac{\pi(n+0.5)}{L}\right)\right)\right] \qquad 0 \le n < L$$

6.2 General Window Trade-offs

A plot of $|W(\nu)|$ in dB for each of these windows will have some common characteristics. First, the spectrum is made up of "lobes." This is similar to what is observed by plotting $|\text{sinc}(f)|^2$ in dB – *i.e.,* , there are nulls at integer values of f in that case and the plot between these nulls are called lobes. The *main lobe* is the lobe near $\nu = 0$ and the other lobes are called *side-lobes*. Factors of interest are the main lobe width and the side-lobe roll-off. The width of the main lobe is the value of ν at the first null. The side-lobe roll-off is the rate at which the side-lobes decay relative to the main-lobe. For example, in continuous time, the $|\text{sinc}(f)|^2$ spectrum has main-lobe width of 1 and has a side-lobe roll-off of $1/f^2$ or 20 dB per decade.

Since the window multiplies $h[n]$ in the time domain, in the frequency domain the spectrum of $h[n]$ is circularly-convolved with $W(\nu)$. It follows that an ideal window would be an impulse in frequency, which is not achievable due to the fact that the window is finite length in time. In practice, one seeks a window with low main-lobe width and fast side-lobe roll-off.

6.3 Problem Statement

In this problem we will consider using the above windows to create an FIR filter to approximate an ideal low-pass filter with impulse response (as derived in lecture)

$$h[n] = 2\nu_0 \text{sinc}\left(\frac{n}{2\nu_0}\right)$$

We will consider $\nu_0 = 1/8 = 0.125$. This could be used, for example, as an anti-aliasing filter for down-sampling a discrete time signal by a factor of 4. For example, if you have a 48 kHz digital audio file that you wanted to downsample to 12 kHz, you would first us a LPF with cut-off frequency $\nu_0 = 1/8$ to eliminate the frequencies above the Nyquist band for $F_s = 12$kHz.

Produce the following set of plots for the two cases of $M = 10$ and $M = 50$ – *i.e.,* , $L = 21$ and $L = 101$, respectively.

1. Plot all of the windows on a common plot.
2. Plot $|W(\nu)|$ in dB for all of the windows on a common plot. Normalize each $W(\nu)$ so that the plot is at 0 dB for $\nu = 0$.
3. For each window plot the following (use a common y-axis scale when you want to make comparisons):
   (a) $w[n]$ (stem plot)
   (b) $|W(\nu)|$ in dB. Normalize $W(\nu)$ so that the plot is at 0 dB for $\nu = 0$.
   (c) $g[n]$ (stem plot)
   (d) $|G(\nu)|$ in dB
4. Plot $|G(\nu)|$ in dB for all of the windows on a common plot – zoomed in around the $\nu = \nu_0$ point.

6.4 Qualitative Comparisons

Based on your results above, address the following:

- Rank the windows according to:
  - "Compactness" in time; more compact windows have most of the energy around their center – *i.e.,* the rect window is the least compact.
  - "Smoothness" in time – *i.e.,* shapes without abrupt changes are smoother. In continuous time, continuous, infinitely differentiable functions are smooth.
  - Main-lobe width in frequency.
  - Side-lobe roll-off in frequency.
- When considering the convolution in the frequency domain, what is the effect of the main-lobe width on $|G(\nu)|$? What is the effect of higher side-lobes on $|G(\nu)|$?
- What is the effect of the increasing the window length?
- All things considered, what window would you use for this FIR low-pass filter design?

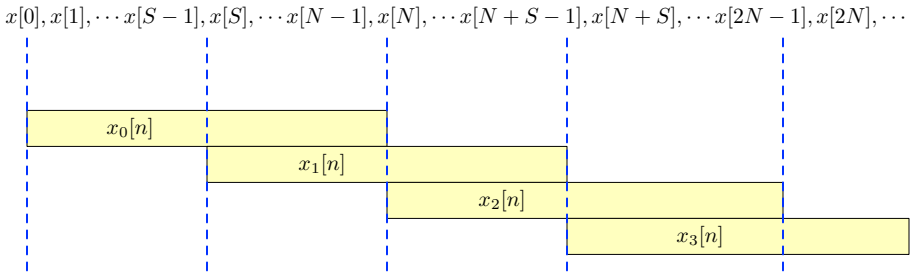

Figure 2: Conversion of the signal $x[n]$ into overlapping frames. For the diagram shown, the frame skip is half a frame – *i.e.,* $S = N/2$.

7 Computer Problem/Lab: Spectrograms

It is often valuable to view the short-term frequency composition of signals. For example, consider a frequency chirp signal that starts at frequency f_0 at time t_0 and ramps the frequency up linearly to frequency f_1 at time t_1. The Fourier transform of this chirp signal will show frequency content across the band from f_0 to f_1. However, the short-term spectral properties of this signal will be lost by taking the Fourier transform over the full time horizon. Specifically, inspecting the Fourier transform will not provide any indication that, at any point in time, the signal is very localized in frequency.

This motivate Short-term Fourier Transform (STFT) analysis wherein short time chunks of the signal (frames) are analyzed in frequency and the results are stitched together in time to give a *spectrogram* – *i.e.,* a view of the signal with time on the x-axis and frequency on the y-axis. During the semester, we have viewed spectrograms generated by Audacity for audio signals. In this problem, you will develop your own spectrogram code and explore audio signals in the time-frequency plane.

7.1 Spectrogram Definition and Notation

Consider a signal $x[n]$ that is defined for $n = 0, 1, \ldots$ and defined the m^{th} frame as

$$x_m[n] = \begin{cases} x[n + mS] & \in \mathcal{Z}_N \\ 0 & n \notin \mathcal{Z}_N \end{cases}$$

where $\mathcal{Z}_N = \{0, 1, \ldots N-1\}$. The parameter N is the frame length and S is the frame-skip (or frame shift) – $1/S$ is referred to as the frame rate. This framing of the signal is shown in Fig. 2. Note that the frames typically overlap meaning that $S < N$ and a common choice is $S = N/2$ when N is even, as shown nominally in Fig. 2.

The DFT (FFT) of each frame is then taken, typically with a window applied. Specifically, the spectrum for frame m is $\mathbb{DFT}\{w[n]x_m[n]\}$ and the spectrogram is the magnitude squared of this, typically plotted in dB

$$S_m[k] = |\mathbb{DFT}\{w[n]x_m[n]\}|^2 \qquad [S_m[k]]_{\text{dB}} = 20\log_{10}\left(|\mathbb{DFT}\{w[n]x_m[n]\}|\right)$$

For this problem, you will use a Hanning window, but feel free to explore. An example spectrogram is shown in Fig. 3.

7.2 Problem Statement

In this problem, you will generate spectrograms for the two audio signals we have used during the semester:

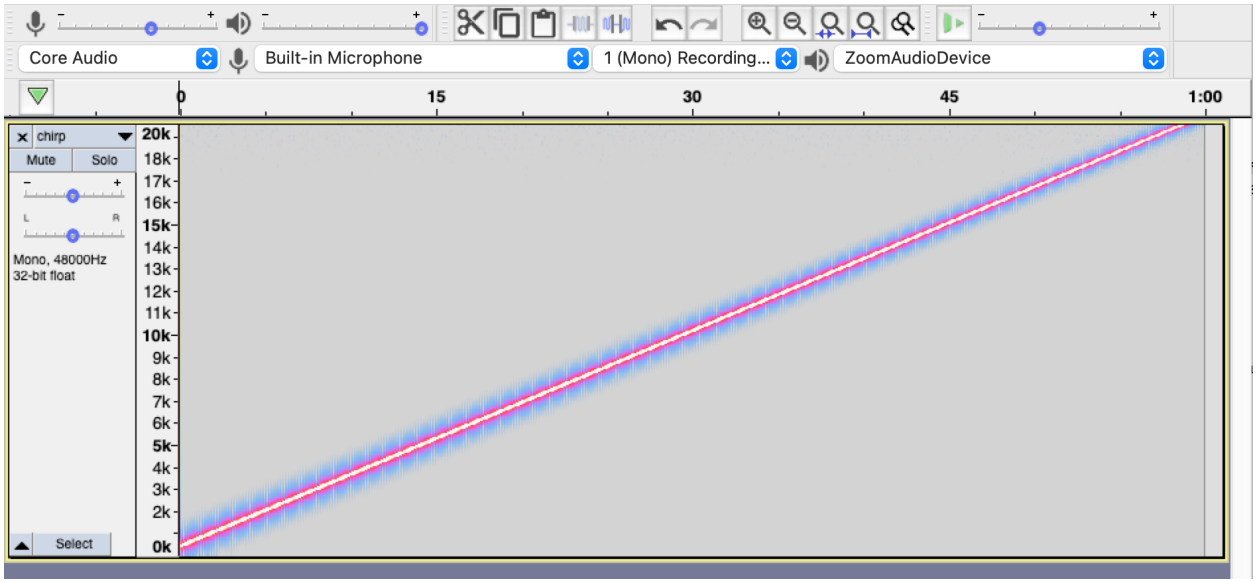

Figure 3: An example spectrogram for an audio chirp signal generated in Audacity.

- `chirp.wav`
- `chugg_welcome.wav`

For each of these audio signals, do the following:

1. Generate a spectrogram with y-axis in kHz and x-axis in seconds – label these axes accordiningly. Plot the $y-$axis up to the Nyquist frequency for the file's sample rate. Note that you can use `numpy.fft.rfft()` to exploit the spectral symmetry around $\nu = 0.5$. For plotting, you can use `matshow()` or the more basic routine `imshow()`

   (a) Produce one spectrogram using $N_{\text{FFT}} = N$ – *i.e.*, the FFT size equal to the frame length.

   (b) Produce a second spectrogram using $N_{\text{FFT}} = 2^{16}$.

   (c) Explain how these two spectrograms differ.

2. Use a low-pass to filter out frequencies above 2 kHz, then repeat the two spectrograms above – *i.e.*, $N_{\text{FFT}} = N$ and $N_{\text{FFT}} = 2^{16}$.

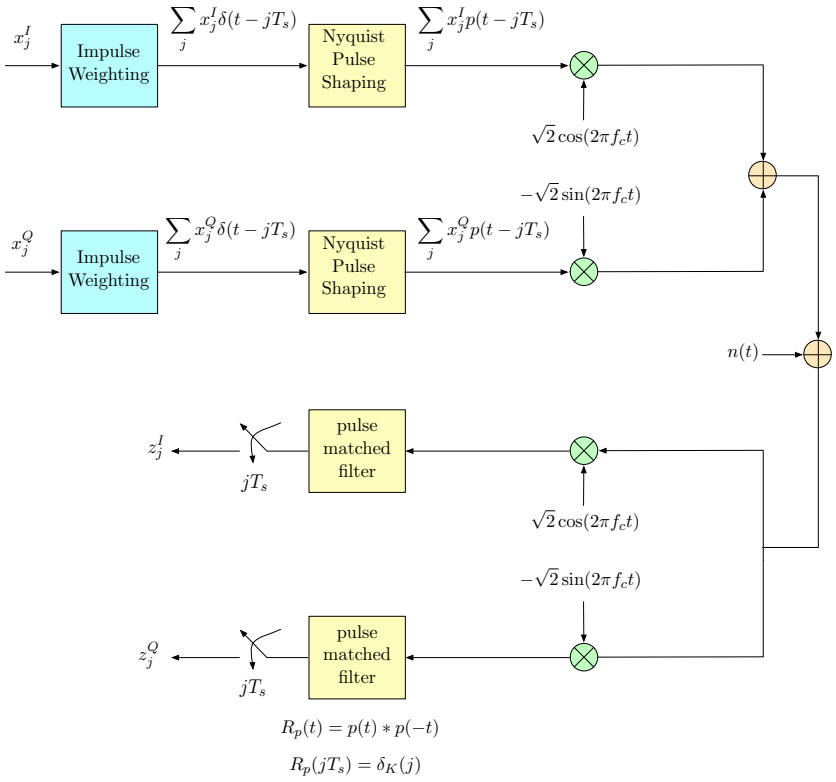

Figure 4: Digital communication system using I/Q modulation with Nyquist pulses.

8 Project: Digital In-phase/Quadrature Modulation Systems

The goal of this project is to understand, simulate, and teach your classmates about the digital communication system in Fig. 4. This system comprises a transmitter (everything before the noise addition), a channel (the noise addition), and a receiver (everything after the noise addition). Every symbol time (T_s), a two-dimensional digital symbol is sent by this transmitter and the receiver attempts to make a decision regarding which of the allowable digital symbol values was transmitted. Below we provide more details on the operation of this system and the relation to previous problems worked in the EE301L.

8.1 Background Information

In this system, a sequence of two-dimensional digital symbols

$$\mathbf{x}_j = \begin{bmatrix} x_j^I \\ x_j^Q \end{bmatrix} \tag{6}$$

is mapped onto a sinusoidal carrier waveform and transmitted. Specifically, the transmitted signal is

$$y(t) = x_I(t)\sqrt{2}\cos(2\pi f_c t) - x_Q(t)\sqrt{2}\sin(2\pi f_c t) \tag{7}$$

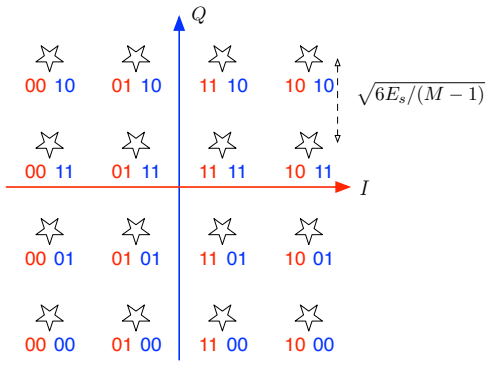

Figure 5: The 16QAM constellation with Gray labeling.

where the so-called in-phase (I) and quadrature (Q) carrier channel signals are

$$x_I(t) = \sum_j x_j^I p(t - jT_s) \tag{8a}$$

$$x_Q(t) = \sum_j x_j^Q p(t - jT_s) \tag{8b}$$

and $p(t)$ is the pulse shaping filter impulse response. It it convenient to work with a unit energy pulse

$$\int_{-\infty}^{\infty} |p(t)|^2 \, dt = 1 \tag{9}$$

8.1.1 Digital Modulation Constellations

The digital modulation format is defined by the finite set of values that $\mathbf{x}_j$ is allowed to take. For example, the $M = 16$ Quadrature Amplitude Modulation (QAM) signal set (*i.e., constellation*) is shown in Fig. 5. Note that, for this 16-QAM modulation, x_j^I and x_j^Q are selected from $\{-3A, -A, +A, +3A\}$ independently to define $\mathbf{x}_j$ as one of the 16 points in the plane. The energy in a symbol is defined as the average of the norm-squared over all of the signal points. For 16-QAM, it can de shown that the signal separation and this symbol energy are related by

$$d = 2A = \sqrt{\frac{6E_s}{16}} \tag{10}$$

Since the source information is typically in binary format, each of these 16 points are labeled by 4 information bits – *i.e.,* each time that a 16-QAM symbol is transmitted, 4 information bits are sent.

There are many types of signal constellations used in this type of modulation. Another common format is phase shift keying (PSK). Figures 6 and 7 show an $M = 8$ PSK constellation, each with a different bit labeling.

8.1.2 Importance of Pulse Shaping

The choice of pulse shaping is important because the (power) spectrum of the modulated signal $y(t)$ is

$$S_y(f) = C\left[|P(f - f_c)|^2 + |P(f + f_c)|^2\right] \tag{11}$$

where C is a positive constant. It follows that the amount of bandwidth required to transmit the signal is determined by $P(f)$ and hence $p(t)$. An obvious choice for $p(t)$ is a rectangular pulse

$$p_{\text{rect}}(t) = \frac{1}{\sqrt{T_s}} \text{rect}\left(\frac{t - T_s/2}{T_s}\right) \tag{12}$$

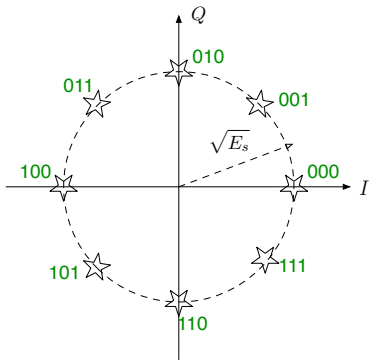

Figure 6: The 8PSK constellation with natural binary labeling.

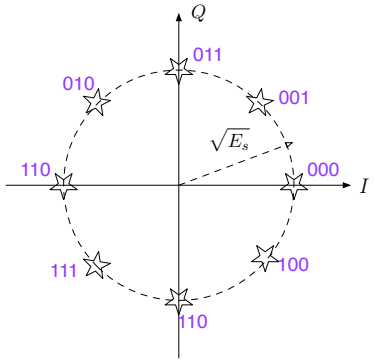

Figure 7: The 8PSK constellation with Gray labeling.

Note that for this choice of pulse, the signal $x_I(t)$, for example, depends only on x_j^I for the interval $t \in [jT_s, (j+1)T_s)$. This is the case for any pulse $p(t)$ that is non-zero only for $t \in [0, T_s)$.

Restricting the pulse duration to one symbol time is not desirable in terms of the spectral properties. For example, for $p_{\text{rect}}(t)$ we have that

$$|P_{\text{rect}}(f)|^2 = T_s \text{sinc}^2(T_s f) \tag{13}$$

which has its first spectral null at $f = 1/T_s$ and has side-lobes that roll of as $1/f^2$. Note that any pulse that is non-zero only for $t \in [0, T_s)$ will have similar properties.

Is it possible to use a pulse that lasts longer than T_s seconds? This would improve the spectral properties, but it will also cause *intersymbol interference (ISI) – i.e.,* the signal $x_I(t)$, for example, on the interval $t \in [jT_s, (j+1)T_s)$ will depend not only x_j^I but on x_m^I for values of $m \neq j$. In other words the pulses associated with different information symbols will overlap in time. The next section outlines how a pulse lasting longer than a symbol duration can be used, thus reducing the bandwidth required.

8.1.3 Nyquist Pulse Shaping

The optimal receiver signal processing is as shown in Fig. 4 which includes an I/Q down-converter and a matched filter on the I and Q channels. From Problem 4.11, we know that the input to the matched filter on the receiver I channel is $x_I(t)$ plus double-frequency terms, plus noise effects from the channel noise. Let's ignore the noise for now. The double frequency terms will be filtered out by the matched filter, which passes frequencies around $f = 0$. Thus, neglecting the $2f_c$ terms and noise, the output of the I channel matched-fliter is[3]

$$z_I(t) = x_I(t) * p(-t) \tag{14}$$

and the symbol-spaced matched-filter output sample signal on the I channel is $z_j^I = z_I(jT_s)$. Similar modeling applies to the Q-channel in the receiver.

A pulse satisfying the *Nyquist criterion for no ISI* is one in which z_j^I only depends on x_j^I – *i.e.,* there is no ISI at the output of the matched filter. Any pulse that is non-zero only on $[0, T_s)$ will satisfy this property, but there are designs that last longer than one symbol time that also satisfy this property. In this project you will explore using pulses that are longer than one symbol time. A Nyquist pulse, one that satisfies the Nyquist criterion for no ISI, is one that introduces ISI in the continuous time transmitted waveform (*e.g.,* $x_I(t)$), but has no ISI in the matched filtered samples (*e.g.,* z_j^I).

8.1.4 Post Matched Filter Decision Device

When a Nyquist pulse is used, a decision on the M-ary digital symbol $\mathbf{x}_j$ is made by a "slicer" – *i.e.,* a device that maps the post matched filter output $\mathbf{z}_j$ to a decision $\hat{\mathbf{x}}_j$. This simply selects the point in the constellation that is closest (in the Euclidean sense) to $\mathbf{z}_j$. This may be viewed as partitioning the I/Q plane into decision regions – *i.e.,* regions where $\mathbf{z}_j$ will be mapped to a given decision. For example, the decision regions for 16 QAM are illustrated in Fig. 8. Note that this is a direct generalization of Correlation/Matched-Filter computational problem that you have already worked. In that case, no quadrature channel was used, the constellation was $\{-1, +1\}$, and the decision regions were just $\{z > 0\}$ and $\{z \leq 0\}$.

A symbol error is made if $\hat{\mathbf{x}}_j \neq \mathbf{x}_j$. When a symbol error is made, the number of bit errors depends on the bit labels of $\hat{\mathbf{x}}_j$ and $\mathbf{x}_j$. In a simulation one would run trials, count errors, and use the ratio of the number of symbol errors to the numbers of symbol trials as the symbol error rate, an approximation of the symbol error probability P_s. An estimate of the bit error probability P_b can be computed in a similar way using the simulated bit error rate. Again, this is a generalization of the Correlation/Matched-Filter computational problem.

[3]Here we assume that $p(t)$ is real so that $p^*(t) = p(t)$.

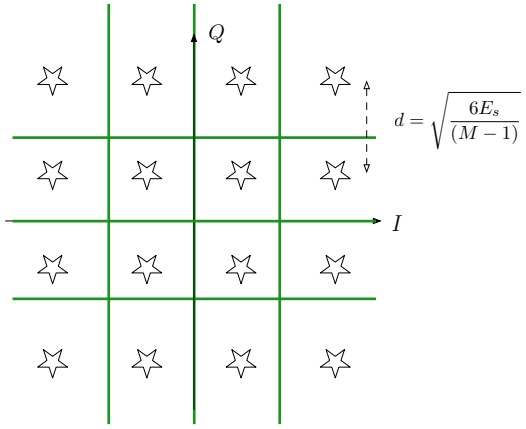

Figure 8: The post matched filter decision device – *i.e.,* a slicer – for 16 QAM.

8.2 Project Work Guidelines

You are free to explore various aspects of this system as part of your project. Below are some guidelines and suggestions to aid you in formulating your project. You will also meet with your project mentor to structure your project.

8.2.1 Analytical Work

Below are a list of results that should review or develop to ensure that you understand the theory behind the system shown in Fig. 4.

1. Review Problem 4.11 and verify that the input to the matched filter on the I channel is $x_i(t)$ plus $2f_c$ terms, plus the effects of the channel noise.
2. Review the solution to the Correlation/Matched-Filter computational problem.
3. Review Problem 4.27.
4. Write $z_I(t)$ in (14) (noise and $2f_c$ terms neglected) in terms of $p(t)$ and x_j^I and show that the condition for z_j^I to depend only on x_j^I (no ISI) is that

$$R_p(mT_s) = \delta[m] \tag{15}$$

   where $R_p(t) = p(t) * p^*(-t)$ is the pulse correlation function. This is the Nyquist criterion in the time domain.
5. State the above condition in terms of $S_p(f) = \mathbb{FT}\{R_p(t)\} = |P(f)|^2$. Note that this is the problem considered in Problem 4.27 with $X(f) = S_p(f)$. This is the Nyquist criterion in the frequency domain.
6. Show that the rect pulse in (12) satisfies the Nyquist criterion. Show this in the time and frequency domains.
7. Show that $|P(f)|^2 = X(f)$ where $X(f)$ is from Problem 4.27 also satisfies the Nyquist criterion. Show this in time and frequency.

8. A common choice for a pulse shape is to choose $S_p(f) = |P(f)|^2$ to be a raised cosine (RC) spectrum. This RC spectrum is similar to $X(f)$ in Problem 4.27, but with smoother shape. This design is

$$|P(f)|^2 = \begin{cases} T_s & |f| < \frac{1-\beta}{2T_s} \\ \frac{T_s}{2}\left[1 - \sin\left(\frac{\pi T_s}{\beta}\left(f - \frac{1}{2T_s}\right)\right)\right] & \frac{1-\beta}{2T_s} \le |f| \le \frac{1+\beta}{2T_s} \\ 0 & |f| > \frac{1+\beta}{2T_s} \end{cases} \tag{16a}$$

$$\beta \in [0, 1) \quad \text{fractional excess bandwidth} \tag{16b}$$

$$R_p(t) = \text{sinc}(t/T_s)\frac{\cos(\beta\pi t/T_s)}{1 - 4\beta^2(t/T_s)^2} \tag{16c}$$

$$P(f) = |P(f)| \tag{16d}$$

$$p(t) = 4\beta\frac{\cos((1+\beta)\pi t/T_s) + \sin((1-\beta)\pi(t/T_s))\left[4\beta(t/T_s)\right]^{-1}}{\pi\sqrt{T_s}\left[1 - (4\beta t/T_s)^2\right]} \tag{16e}$$

   Plot $|P(f)|^2$, $R_p(t)$ and $p(t)$ for various values of β and note why this this is a Nyquist pulse (time and frequency). Note that $p(t)$ is referred to as a root-raised cosine (RRC) pulse.

Following these steps, you will have demonstrated that the output of the I and Q matched filters is

$$\mathbf{z}_j = \mathbf{x}_j + \mathbf{n}_j \tag{17}$$

where $\mathbf{n}_j$ is the effects of the channel noise.

A resource for this material is my EE564 Detection Theory and Performance Analysis Slides. In particular, the theoretical performance for PSK and QAM is provided on slides 28-36.[4] The Nyquist condition and the RRC pulse results are developed in slides 82-94.[5]

8.2.2 Simulation Work

You may choose to simulate all of or part of the system in Fig. 4. If you choose to simulate the entire block diagram, plus the slicer, then you will need to choose a value for f_c. In this case, it is suggested that you normalize to a unit sample time (*i.e.*, $T_s = 1$) and choose the carrier frequency to be ≥ 100. The simulation will have to done using a sampled version of the signal models as was the case in the Correlation/Matched Filter computational problem. Here, the sample rate should be higher than twice the carrier frequency, so using $F_{\text{sample}} = 4f_c = 400$ should suffice. This means that you will be running your simulation with a sample time of $T_{\text{sample}} = 1/400$ or 400 samples per symbol time. In this case, you can generate the noise samples using `np.random.normal(0, sigma, N)` where sigma is

$$\sigma = \sqrt{\frac{N_0}{2}F_{\text{sample}}}$$

and N is the length of the array desired.

To simulate the matched filter for a RRC pulse, you will have to window it (an RRC pulse is infinite length) and delay it to make it start at zero. You may wish to do an experiment with just this windowed RRC and the associated matched filter to verify that the Nyquist condition is still satisfied in practice. In other words, windowning the RRC pulse will cause the Nyquist condition to not hold exactly, so it is an engineering design trade-off to choose the window length and the degree of approximation for the Nyquist condition. In practice, a choice of $\beta = 0.35$ is a good choice for the RRC excess bandwidth.

To characterize your simulation, you can produce a scatter plot for the values of $\mathbf{z}_j$ at various SNR values (*e.g.*, E_s/N_0 values). At higher SNR, the values of $\mathbf{z}_j$ should cluster around the actual transmitted symbols and at lower SNR, these clusters should be larger resulting in decision errors.

[4] The Q-function is the integral of the tail of a Gaussian bell and can be computed in Python using `scipy.stats.norm.sf`.
[5] These slides are beyond the scope of EE301L, but you will be reproducing some of these results.

A full simulation would include a symbol error rate and a bit error rate computation so as to produce a curve of SER and BER vs. E_s/N_0.

More details and help on the simulation can be obtained from the project mentor.

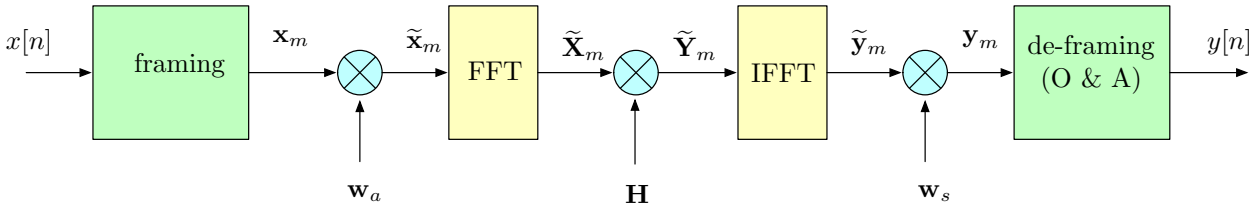

Figure 9: Block diagram for real-time frequency domain filtering using overlap and add. All boldface quantities are vectors of size N, which is the frame length and the FFT size.

9 Project: Frequency Domain Filtering Using Overlap and Add with FFTs

The goal of this project is to understand, simulate, and teach your classmates about the filtering system in Fig. 9 which is a practical way to implement a filter in the frequency domain. Note that, in practice, one cannot compute the DTFT, apply a frequency gain, and then apply the inverse DTFT since this requires an infinite time horizon.

This system in Fig. 9 builds on the concept of a spectrogram explored in a previous computational problem. Specifically, the portion of the diagram in Fig. 9 mapping the input $x[n]$ to $\widetilde{\mathbf{X}}_m$ is the spectrum used to compute the spectrogram considered in the Spectrograms computational problem.[6] A vector notation is adopted in Fig. 9 where $\mathbf{x}_m$ is the m^{th} frame, $\mathbf{w}_a$ represents the analysis window, and $\widetilde{\mathbf{X}}_m$ is the m^{th} short-term Fourier transform (STFT) frame.

After producing $\widetilde{\mathbf{X}}_m$, a desired frequency domain filter $\mathbf{H}$ is applied and the inverse FFT is applied. The output is then windowed by a synthesis window $\mathbf{w}_s$ to produce the output frame $\mathbf{y}_m$. These are then "de-framed" using an overlap and add technique to produce the output signal $y[n]$. The goal of this system is to implement a discrete time filter with frequency response $H(\nu)$ having frequency samples $H(k/N)$ that comprise the FFT bin gains contained in $\mathbf{H}$.

If the identity filter is applied (*i.e.,* $\mathbf{H} = \mathbf{1}$), then we would expect $y[n] = x[n]$. This *perfect reconstruction* criterion requires some conditions on the analysis and synthesis windows. Also, since multiplication of two DFTs yields circular convolution in the time domain, the system in Fig. 9 is clearly not exactly the same as multiplying the DTFT of $x[n]$ by $H(\nu)$. In this project you will explore the conditions needed for the windows for perfect reconstruction and explore the effective frequency response of the entire system.

9.1 Background Information

The same framing convention used in the Spectrograms computational problem is employed here

$$x_m[n] = \begin{cases} x[n+mS] & \in \mathcal{Z}_N \\ 0 & n \notin \mathcal{Z}_N \end{cases} \tag{18}$$

where, again, we will consider the frame skip $S = N/2$ where N is the even-valued frame length. In other words, two adjacent frames overlap by 50% as shown in Fig. 10 which is duplicated from the Spectrograms computational problem. The analysis window is applied to this via

$$\widetilde{x}_m[n] = w_a[n]x_m[n] \tag{19}$$

where $w_a[n]$ is the analysis window (length N). We then take the N-point FFT of this windowed frame to obtain

$$\widetilde{X}_m[k] = \mathbb{DFT}\left\{\widetilde{x}_m[n]\right\} \tag{20}$$

[6] In the Spectrogram problem we computed the magnitude of $\widetilde{\mathbf{X}}_m$.

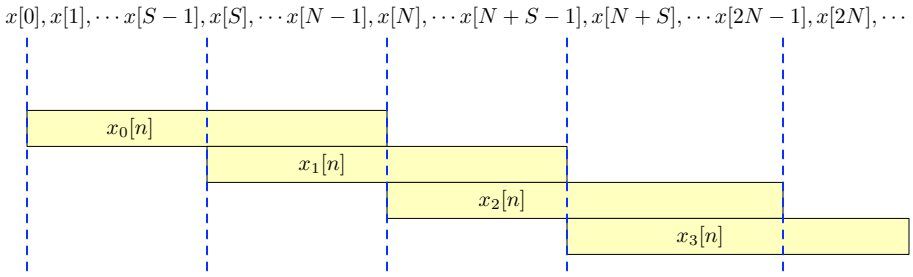

Figure 10: Conversion of the signal $x[n]$ into overlapping frames. For the diagram shown, the frame skip is half a frame – *i.e.,* $S = N/2$ – which will be assumed throughout this project.

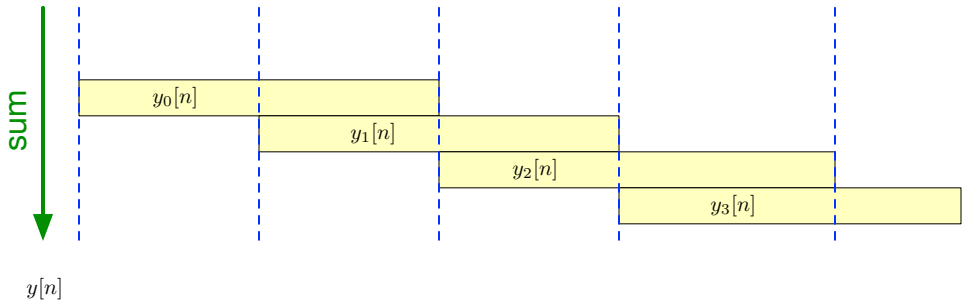

Figure 11: The de-framing operation using overlap and add. The output at $y[n]$ is the sum of two frames for each value of n.

The desired frequency domain filter gains $H[k]$ are then applied to this spectrum

$$\widetilde{Y}_m[k] = H[k]\widetilde{X}_m[k] \tag{21}$$

The inverse FFT of this frequency domain signal is

$$\widetilde{y}_m[n] = \mathbb{DFT}^{-1}\left\{\widetilde{Y}_m[n]\right\} = x_m[n] \circledast_N h[n] \tag{22}$$

where $h[n] = \mathbb{DFT}^{-1}\{H[k]\}$ and $\circledast_N$ denotes circular convolution. This signal is then windowed by the synthesis window producing

$$y_m[n] = w_s[n]\widetilde{y}_m[n] \tag{23}$$

Finally, these output frames are combined using the overlap and add method diagrammed in Fig. 11. This is the de-framing operation in Fig. 9. Note that each time n, the system output $y[n]$ is the sum of two values of $y_m[n]$ – *i.e.,* for $y_m[n]$ and $y_{m-1}[n]$.

The signals $x_m[n]$, $w_a[n]$, $\widetilde{x}_m[n]$, $\widetilde{X}_m[k]$, $H[k]$, $\widetilde{Y}_m[k]$, $\widetilde{y}_m[k]$, and $y_m[k]$ are all length N, so can be interpreted as vectors. This is the interpretation of Fig. 9.

9.2 Project Work Guidelines

You are free to explore various aspects of this system as part of your project. Below are some guidelines and suggestions to aid you in formulating your project. You will also meet with your project mentor to structure your project.

9.2.1 Analytical Work

Below are a list of results that you should review or develop to ensure that you understand the theory behind the system shown in Fig. 9.

1. Review the Spectrograms Computational problem solution.
2. Determine the condition on the analysis and synthesis windows for perfect reconstruction. This is the Princen-Bradley condition [1].[7]
   (a) Show that using the Vorbis window for both analysis and synthesis provides perfect reconstruction. This can be done analytically or computationally.
   (b) Show that using the rectangular window and the Hanning window as an analysis/synthesis pair provides perfect reconstruction. This can be done analytically or computationally.
   (c) Show that the square-root of the Hanning window can be used as the analysis and synthesis window for perfect reconstruction.
   (d) Can you suggest a pair of windows, other than those above, that would satisfy the Princen-Bradley condition?
3. The circular convolution will cause corruption at the edges of the frame. Explain the intuition for the overlap and add approach in dealing with this undesired effect – *i.e.,* use the rect window for analysis and the Hanning window for synthesis for this intuitive explanation.
4. Is the overall system that maps $x[n]$ to $y[n]$ LTI? Is it approximately LTI? If you believe it is LTI, is it simple to compute the overall frequency response for this system?
5. Discuss the computational complexity of the system in Fig. 9. How does this compare to performing a convolution in the time domain with an FIR filter with impulse response having length $\leq N$. How does it compare to using an ARMA filter with order L much smaller than N?
6. Discuss the processing delay encountered by the system in Fig. 9 – *i.e.,* if it is used for a real-time system.

9.2.2 Simulation Work

You may choose to use computer experiments and simulations to better understand the system in Fig. 9. Below are some suggestions:

1. Audio signal filtering. Use a wav file as the input $x[n]$ and filter the signal using the system of Fig. 9. You will want to choose a reasonable frame size. For real-time, interactive processing it is typical to use a frame duration of about 20 msec. If you use an audio sample rate of 48 kHz, this is $N = 960$. As a baseline, use the Vorbis window for both analysis and synthesis.
   (a) Design the frequency domain filter to be a low-pass filter with cut-off 2 kHz. What is the value of ν_0 for this discrete time filter? What is the value of k_0 – *i.e.,* the cut-off frequency in $H[k]$ for this filter?
   (b) Filter a sample audio filter using the overlap-and-add approach. Compare the output with the result of the filters considered at the end of DTFT example Python notebook – *e.g.,* a windowed ideal LPF and a Butterworth (AR) filter.

[7] The cited paper deals with Discrete Cosine Transforms (DCT) and is therefore a bit difficult to read and directly apply here. When the DCT is used in this overlap and add method with framing, it is called the Modified DCT (MDCT). See the MDCT wikipedia page for a brief description of windows that satisfy the Princen-Bradley condition.

(c) If this system can be approximated as LTI, plot the magnitude of the frequency response for the overall system. Note that you may use the cosine-in/cosine-out method for measuring the frequency response that was developed in the Two-Tap DT Filter computational problem if needed.

(d) Explore the effects of window choice. Specifically, compare the results obatined above with that of using a rect-Hanning and Hanning-rect window combination. Which set of windows do you think provides the best filter results?

2. Other considerations to explore are:

(a) Numerical examples of the Princen-Bradley condition for various window combinations.

(b) The effects (*e.g.,* audio artifacts) if a window pair is used that does not provide perfect reconstruction.

(c) Examples of the system output when an a unit impulse is input at different times.

(d) Comparison of $\widetilde{y}_m[n]$ and the corresponding system output $y[n]$ for simple cases – *i.e.,* to build intuition regaridng the O & A method.

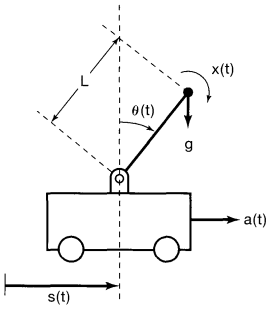

Figure 12: The inverted pendulum system considered. Diagram copied from [2] Figure P11.56.

10 Project: Feedback Control of an Inverted Pednulum

The goal of this project is to understand, simulate, and teach your classmates about the inverted pendulum system considered in Oppenheim, et. al., Problem 11.56. In this problem, the inverted pendulum on a cart in Fig. 12 is considered. As described in the problem, the dynamics of this system are governed by a nonlinear differential equation

$$\theta''(t) = \frac{g}{L}\sin\theta(t) - \frac{1}{L}a(t)\cos\theta(t) + x(t) \tag{24}$$

where $x(t)$ is a perturbing input on angular measurement $\theta(t)$.[8]

The problem walks you through a series of steps to show that, for small $\theta(t)$, the nonlinear DE can be linearized and that a proportional-derivative (PD) controller can be used to stabilize the linear model.

10.1 Project Work Guidelines

You are free to explore various aspects of this system as part of your project. Below are some guidelines and suggestions to aid you in formulating your project. You will also meet with your project mentor to structure your project.

10.1.1 Analytical Work

Below are a list of results that you should review or develop to ensure that you understand the theory behind the system shown in Fig. 12.

1. Work Problem 11.56 from [2].
2. For the solution to 11.56(c), develop a root-locus digram as you change the parameter K_1 around the the value for the solution. Similarly, develop a root-locus digram as you change the parameter K_2 around the the value for the solution. In other words, hold one parameter fixed at the designed solution and vary the other to create a root locus diagram.
3. Investigate other models for the inverted pendulum on a cart and explain how they differ from the model in problem 11.56. For example, see the notes by Daleh, et. al., example 6.3 on page 63.

[8] Note that this assumes that the mass of the cart is $M = 1$, the mass of the pendulum is $m \ll M$, and the mass of the rod is negligible.

10.1.2 Simulation Work

You may choose to use computer experiments and simulations to better understand control of the system in Fig. 12. Below are some suggestions:

1. Simulate the nonlinear dynamical system in (24) and apply the controller developed in Problem 11.56(c). Plot the step response of the open loop and closed loop systems.

2. Use your simulation to consider initial conditions with $\theta(0)$ large enough that the linearization of the model in (24) no longer applies. Specifically, compare the PD controller from Problem 11.56(c) acting on the linear model and the nonlinear model when the initial conditions are non-zero.

3. There are numerous extensions to this problem, including a double pendulum and more accurate/complex dynamical models. There are also many examples of real-time control systems that have been applied to this problem. Furthermore, there are real problems, such as rocket landing, that map to this problem. Research these topics and provide a summary of the most interesting examples in your presentation to the class.

References

[1] J. Princen and A. Bradley, "Analysis/synthesis filter bank design based on time domain aliasing cancellation," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 34, no. 5, pp. 1153–1161, 1986.

[2] A. V. Oppenheim, A. S. Willsky, and I. T. Young, *Signals and Systems*. Englewood Cliffs, New Jersey: Prentice-Hall, 1983.