ECE 420

Aiden Sirotkine

Fall 2025

Contents

Chapter 1

ECE 310 Overview

1.1 Sampling

1.1.1 Shannon-Nyquist Theorem:

Proves that discrete samples can perfectly reconstruct a continuous signal.

- Reconstruction
- Up-Down Sampling
- z-transform
- CTFT
- DTFT
- DFT (FFT)
- Yea its literally all fourier transforms lmao

1.1.2 Discrete LTI System

Convolution

- Impulse Response
- Frequency Response

1.1.3 Filters

Digital Filters

- FIR vs IIR
- Linear Phase

1.2 310 vs 420

- How do we get the data?
  ECE 310 = offline (batched)
  ECE 420 = online (stream)
  We use buffering techniques
  overlapped added
  windowing
- Who Computes?
  ECE 310 = Computer code
  ECE 420 = Mobile app/phone
  RUN-TIME IS IMPORTANT
  time-domain processes and FFT'S

1.3 Skillsets

Android app dev
C++ and Java.
Do not need to know crazy circuit shenanigans
You do not need a fancy UI for the DSP final project.
The DSP algorithm is more important.

1.4 Signals

IMU signals = 1d signals that are acceleration + Gyro
Audio signal is also 1d in the 20 - 20,000 Hz range
Image signals are 2d signals with visible light and video.

1.5 Android Device

You can loan a tablet
Need a specific OS
Needs gradle compiler.

1.6 Code

Need Python and need Android Studio

1.7 Basic Practice

1. Develop and Test DSP algorithms in hihg-level languages (Python)
2. Port tested algorithms into Android platform (C++, Java)

1.7.1 Attendance Quiz

There will be a quiz and you answer the question in the quiz and that gives you the attendance credit.

1.8 Labs

1. Lab Quiz (PrairieTest)
2. Demo Lab
3. Office Hours

Prelab (Individual)
Quiz (Individual) Need laptop and ID
Lab Demo (Group). Groups are randomized every week.
This class should be very chill I'll be so fr I'm very glad I picked this class.

Chapter 2

Sampling

Take amplitude at various points in time.

You then reconstruct the continuous signal using just the sampled points.

2.1 CTFT

$$X_a(\Omega) = \int_{\infty}^{\infty} x_a(t) e^{j\Omega t} dt$$

Scale by Ts and make periodic every 2π to get

2.1.1 DTFT

$$X(\omega) = \sum_{\infty}^{\infty} x[n] e^{j\omega n}$$

To avoid overlap,

$$B < \frac{\pi}{T_s} \qquad f < \frac{1}{2T_s}$$

Where T_s is the sampling period so $1/T_s$ is the sampling rate, and B is the angular speed

2.1.2 Nyquist rate

$$F_s = \frac{1}{T_s} > 2f$$

2.1.3 Audio Nyquist Rate

The band max is 20kHz, so the Nyquist rate is

$$\frac{1}{T_s} = 2f = 40Khz$$

So the sampling rate is 40kHz

2.2 Digital Filtering

In Lab 2, we're going to want to take out certain frequencies from our entire noise space.

We can try to use a continuous-time bandstop filter. We could use an RLC circuit, but that has all sorts of consequences.

Instead, we can use a digital filter.

Digital Filters are equivalent to analog filters IF we have Nyquist rate sampling.

2.2.1 Digital Filter

Big silly equation that's in the lecture slides

$$y[n] = (b_0x[n] + b_1x[n-1] + \ldots + b_Kx[n-K]) - (a_0y[n] + a_1y[n-1] + \ldots + a_Ly[n-L])$$

FIR, if no feedback (L=0)
IIR, if feedback (L $\neq$ 0)

2.2.2 Large N

- CLose to desired response
- Sharper transition
- Less ripples

  BUT

- More computation/memory
- Longer Delay
- (for IIR) possible worse performance

2.2.3 FIR and Convolution

$$y[n] = (b_0 x[n] + b_1 x[n-1] + \ldots + b_K x[n-K])$$

$$y[n] = \sum_{k=0}^{K} b_k x[n-k]$$

2.2.4 Batch vs Block Process

h = filter x = batch samples h*x = ideal output

makes an N-size output

We can have multiple convolution functions, and they can cause discontinuities if we just add them together.

2.2.5 Convolution by Circular Buffer

Challege 1: Block Processing

Audio samples come as a buffer,

which means discontinuities between buffers

Solution: Use a buffer (as a global variable) to store the samples from the previous buffer.

something something more words from the lecture slides.

$$y[n] = \sum_{k=0}^{K} h[k] x[n-k]$$

Assume $K = 2$ and $h[n], y[n] = 0$

Consider a Circular Buffer 0, 1, 2

idk fix later

2.3 OpenSL ES

Open Sound Library for Embedded Systems

Use default sampling rate (48kHz)

The library gives you weird 8 bit sampling

Use a bitwise operature to turn the 8 bit sampled data into the original 16 bit data.

Chapter 3

Spectral Analysis

Lab2 and Quiz 2 on digital filtering and Audio notch filtering.

Spectral analysis give you signal in the time domain.

You can also see signal in the frequency domain.

It shows the relative distribution of signal "energy" in a different basis

The most common choice is the Fourier basis (frequency)

The magnitude/log, phase and other post-processing are possible.

3.1 Fourier Transforms

Turns time domain into frequency domain or vice versa.

3.1.1 Continuous Time, Continuous Frequency

You use a CTFT, $X_A(\Omega)$

$$X_a(\Omega) = \int_{-\infty}^{\infty} x_a(t) e^{-j\Omega t} dt \qquad \Omega = 2\pi f$$

3.1.2 Discrete Time, Continuous Frequency

You use a DTFT $X(\omega)$

$$X(\omega) = \sum_{-\infty}^{\infty} x[n] e^{-j\omega n} dt \qquad \Omega = 2\pi f$$

3.1.3 Discrete Time and Discrete Frequency

DFT X[k]

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j2\pi kn/N}$$

3.1.4 Continuous Time and Discrete Frequency

Fourier Series $\{a_k\}$

3.2 CTFT vs DTFT vs DFT

- CTFT use an integral and everything is continuous
- DTFT takes in a discrete input and you use a discrete sum, but you get a continuous output
- DFT's exist because you cant integrate or sum to infinity on a computer.

  The relation between all of them is

$$\frac{f}{F_s} = \frac{\omega}{2\pi} = \frac{k}{N}$$

3.2.1 Consequences to DFT Truncation

In order to use DFT, the length of the input samples must be **finite**

3.3 Time-Windowing

Duration and bandwidth are inverse.

3.3.1 Rectangular Window

$$W_R = \begin{cases} 1 : 0 < t < T \\ 0 : \text{ otherwise} \end{cases}$$

It's just a step function.

If we have a function of just

$x_a(t) = A\cos(\omega_1 t)$

Then when we take our DFT, we get

$X_a(j\Omega) = A\pi\delta(\Omega - \omega_1) + A\pi\delta(\Omega + \omega_1)$

So we take our Discrete Fourier Transform to get

$$\tilde{x_a}(t) = w_R(t)x_a(t) = \frac{1}{2}Aw_R(t)e^{j\omega_1 t} + \frac{1}{2}Aw_R(t)e^{-j\omega_1 t}$$
$$\tilde{X}_a(\Omega) = w_R(t)x_a(t) = \frac{1}{2}Aw_R(t)(\Omega - \omega_1) + \frac{1}{2}Aw_R(t)(\Omega + \omega_1)$$

3.3.2 Hamming Window

Instead of a step function, it's more of a bump.

There are not artifacts on the side, but the main lobe is more wide.

3.4 Zero-Padding

When the window length L is less than the DFT length N, you add $N - L$ zeros to the end of the sequence. However, it **only** increases the resolution of the DFT, **not the DTFT**.

If you want an actually sharper frequency-domain image, you have to increase the length of the DTFT.

3.5 STFT

So far, we've assumed that signals are periodic and stationary. If this is not true, we have to change our Fourier transform parameters.

A STFT can cut out a segment from a signal and move the window with a shift m

$$X(\Omega, t) = \int_{-\infty}^{\infty} w(t-\tau)x(t)e^{-j\Omega t}\,dt$$

$$X(k, m] = \sum_{n=0}^{N-1} w[n-m]x[n]e^{-j2\pi kn/N}$$

3.6 Uncertainty Principle

Time resolution and frequency resolution cannot be improved simultaneously in the spectrum

3.7 Spectrogram

Magnitude of STFT.

Every timestamp, you perform an STFT to get a 2d plot of frequency over time.

3.7.1 Resolution vs Window Size

Larger windows mean finer frequency resolution. Smaller windows mean.

A Hamming window is a very good option for a spectrogram.

Rectangular windows are very noisy with prominent side-lobes. They are not good for spectrograms.

Chapter 4

Source-Filter Model

Excitation Generator $\rightarrow$ LTI System h(t) (Linear and Time Invariant).

The excitation parameters are

- amplitude (loudness)
- frequency (pitch)
- phase/delay
- Type (voiced, unvoiced, silenced)

The parameters of an LTI system are

- IMPULSE RESPONSE
- resonance frequency

In the frequency space, you can just multiply the generated signal and the LTI system's frequency response.

An LTI system **cannot** create new frequencies in the output.

4.0.1 Speech

Given speech, you can see multiple different syllables, but it has very dynamically changing frequencies and amplitude.

Speech contains an envelope of frequencies, and human speech is contained in a very narrow bandwidth ($< 2000Hz$)

The sampling rate for speech is usually only 4kHz or 8kHz because humans are not very high pitched.

4.1 Characterization of Frames

- Voiced Sounds
- Unvoiced Sounds (P)
- Silence/noise (no active speech)

4.2 Pitch Detection Algorithm

Figure out if sound is voiced or unvoiced.

If voiced, find the frequency. Voiced signals are touder and more sustained. Unvoiced signals are more abrupt.

Pitch calculation is found with autocorrelation

$$R_{xx}[l] = \frac{\sum_{n=0}^{N-1} x[n]x^*[n-l]}{\sum_{n=0}^{N-1} |x[n]|^2}$$

Auto-correlation is an N length vector that spans the possible available lags.

l is defined by a circular shift

$$x^*[< n-l >_N]$$

So the lag wraps around if you're using a negative number (this is already done in python).

The way you figure out the frequency is with good old unit analysis because lag is in samples and sampling rate is samples per second.

$$fs * \frac{1}{l} = \frac{samples}{s} * \frac{1}{samples} = \frac{1}{s} = Hz$$

4.3 Complexity

big O notation.

the autocorrelation function is $O(n^2)$ Because finding the autocorrelation for a single lag is $O(n)$, and then

It is very similar to a circular convolution

$$y[l] = \sum_{n=0}^{N-1} x[n]h[l-n]$$

Convolution involves flipping the out-of-phase portion

The reason we use circular convolution is because of now DFT's work

The autocorrelation can be written as

$X[k]X^*[k]$

4.4 Uncertainty Principle

Time resolution and frequency resolution cannot be improved simultaneously in the spectrum.

4.5 Time-Windowing

Effect of time-windowing: Blurring and spreading the original spectrum. To improve the frequency resolution, use a longer time window.

Rectangular window means higher sidelobes. Increase T and decrease $\Delta\Omega$.

4.6 Challenges of Pitch Detection

4.7 Summary

- Source-filter model for speech signal
- ptich detection by autocorrelation
- autocorrelation and its boost by FFT

4.8 310 Review

Given a discrete time system, how can we use it to filter a continuous time signal. If we sample at 10kHz and get a 1kHZ signal. We can do a DTFT to get a discrete frequency

$$\frac{f}{Fs} = \frac{\omega}{2\pi} = \frac{k}{N}$$

So given a frequency f and a sampling rate Fs, our DTFT signal should be

$$2\pi\frac{f}{Fs} = \omega = 0.2\pi$$

You can do the inverse to get the frequency Ω from ω

What if we want to change the pitch from 1kHz to 1.2kHz? The way we can do that is by changing the sampling rate.

$$\frac{f}{Fs} = \frac{f_2}{Fs_2} \rightarrow 0.1 = \frac{1200}{Fs_2} \rightarrow Fs_2 = 12000kHz$$

4.8.1 Upsampling

perform zero-insertion on the signal (different from zero-padding because it increases the wavelength of every wave in the signal). We do not lose any data from upsampling. Upsampling in the frequency domain can be written as

$$y[n] = \begin{cases} x[n/M] : n\%M = 0 \\ 0 : \text{otherwise} \end{cases}$$

$$Y(\omega) = \sum_{-\infty}^{\infty} y[n]e^{-k\omega m} = \sum_{-\infty}^{\infty} x[n/M]e^{-k\omega m} = \sum_{-\infty}^{\infty} x[l]e^{-k\omega ml}$$

So ω is compressed by M

$$Y(\omega) = X(M\omega)$$

Remove the aliasing spectrum to get just the correct compressed frequency

4.8.2 Upsampling with Interpolation

Use a low pass filter (LPF)

$$x \rightarrow\uparrow M \rightarrow LPF(\frac{\pi}{M})$$

By doing this, we can get

$$y[n] = \sum_{k=-\infty}^{\infty} s[k] sinc(\frac{\pi(k - kM)}{M})$$

Fill the missing samples with an interpolation kernel function. upsampling **increases wavelength** $P_0 \rightarrow MP_0$. A consequence of this is that **frequency decreases**.

4.9 Downsampling

Remove features

$$x[n] \rightarrow\downarrow L \rightarrow y[n]$$

You do a long proof to get the final change in the frequency space

$$y[n] = x[Ln]$$

$$Y(\omega) = \frac{1}{L} \sum_{n=0}^{L-1} X\left(\frac{\omega - 2\pi}{L}\right)$$

The signal is stretched by L and shifted by $2\pi n$. To prevent aliasing, we want a smaller bandwidth

$$LB < \pi \rightarrow B < \frac{\pi}{L}$$

Downsampling **compresses wavelength** and a consequence of that is that **frequency increases**.

4.10 Putting the 2 together

$$x \rightarrow LPF(\pi/L) \rightarrow\downarrow L \rightarrow\uparrow M \rightarrow LPF(\frac{\pi}{M})$$

You can upsample first, but nothing useful will happen. You should always downsample first.

4.11 Modifying Pitch P_0

You can resample to change the pitch (upsampling first)

$$x \rightarrow\uparrow M \rightarrow LPF(\frac{\pi}{max(M,L)}) \rightarrow\downarrow L \rightarrow y$$

$$P_1 = \frac{M}{L}P_0$$

This allows you change pitches by only a certain fraction. This stretches or compresses the entire spectrum. Both pitch and vocal tract response (color of your voice) change! How do we modify only pitch?

4.12 TD-PSOLA

This algorithm can modify the fundamental pitch of a signal **without affecting the formants** (vocal tract response).

TD = time domain PS = Pitch Synchronous (operate around reference points)

OLA = overlap add (the synthesized signal overlaps and are added together to form the final output)

Excitation Generator x(t) $\rightarrow$ Linear System h(t) $\rightarrow$ y(t) = x(t) * h(t)

You use a dirac delta convolution.

$$x[n] = \sum \delta[n - P_0 k] \qquad \hat{x}[n] = \sum \delta[n - P_1 k]$$

$$\hat{y}[n] = \hat{x}[n] * h[n] = \sum h[n - P_1 k]$$

It essentially adds empty space in between periods to change the frequency without changing the essence of the signal.

4.12.1 Finding Epochs

- Signal peaks
- provide the reference point
- lab4 algorithm (autocorrelation)

4.12.2 Epoch Mapping

You can find the inputs with pitch and waveform analysis. Output epochs are regularly spaces

4.12.3 Signal Synthesis by Windowing

If you just use the algorithm directly, you will get huge discontinuities in the new signal. You need to use a window

You can't use a regular bell shaped window because the peaks are not necessarily in the center of the period, so the signal will be ruined.

Instead, you use a window the size of 2 periods, so that the window peak is at the signal peak.

4.12.4 Block Processing Challenges

there are 2 main issues

- the windowed interval may stretch across multiple **input** frames
- the windowed interval may stretch across multiple **output** frames

The way to fix this is by storing the buffers in 'past', 'present' and 'future' for both the input and output. Let the buffer spill into both the past and future frames to keep everything continuous.

Chapter 5

Images

Audio is a 1d signal, while pictures are a 2d signal (x, y for each signal point). A digitized image is a bunch of pixels. You can use linear transformations to change the locations of each pixel. You can also change the brightness and colors of pixels. You also blur and sharpen images (not really an individual pixel change). This is known as spacial filtering (Examining/changing local regions).

5.1 Intensity

1 value for 1 pixel.

- Binary (0, 1)
- Grayscale (0, 255)
- color (3 channel RGB, etc)

5.1.1 Intensity Transformation

$$s = T(r) \qquad s, r \in [0, L-1]$$
$$s = r + C$$

Just increase the intensity by a constant. This is called **enhancing**.

$$s = L - 1 - r$$

This is known as the **negative image**.

$$s = \Big\{ 255 : a < s < bs : \text{otherwise}$$

This is called **intensity-level slicing**. It essentially changes values within a certain interval to a constant without changing anything outside the interval.

5.2 Histogram Processing

Histograms represent a distribution of numerical data. Observing the histogram can tell you spikes of certain data. The histogram range is called the **contrast** or **dynamic range**. Consider how the previous transformations affect the histogram. For the lab, we want a low-contract, low-dynamic range image to turn into a high-contrast, high dynamic range

5.2.1 Histogram Equalization

Turn your histogram into a normal distribution. The probability distribution becomes flat and the cdf is just y=x.

$$s = h(r) = round\left(\frac{cdf[r] - cdf_{min}}{MN - cdf_{min}}(L-1)\right)$$

M, N are the width and height of the image. L is the total number of gray levels. CDF_{min} is the smallest **non-zero** value that the cdf can have. cdf[r] is the **total number** of pixels **at or below** the value r.

5.3 Spatial Filtering

Spatial filtering is a predefined operation that is performed on a pixel **and** its neighborhood.

5.3.1 2D Convolution

a 1d convolution is

$$g(t) = f(t) * w(t) = \sum_k f(k)w(t-k)$$

but a 2d convolution is

$$g(x,y) = f(x,y) * w(x,y) = \sum_i \sum_j f(i,j)w(x-i, y-j)$$

Basically, each pixel is defined by a kernel function interacting with the pixel and everything around it. Depending on where you start the kernel, you can get output sizes that are different from the input sizes.

5.4 Convolution Output Domain

5.4.1 Valid

kernel does not go outside original image. Output size $N - K + 1$

5.4.2 Same

Same size N as input image.

5.4.3 Full

Maximum sized convolution output size N + K - 1

5.5 Smooth Spatial Domain

'average' the neighborhood of pixels to reduce sharp transitions.

5.5.1 Median filter

Replace the value of the pixel with the median of the neighborhood. Effectively removes impulse noise and salt and pepper noise.

5.6 Sharpening Spatial Filters

Edge detection. Highlights transitions in intensity. similar to derivative operator. It could also be a high pass filter in the frequency domain.

5.6.1 Gradient

Take the first derivative of the image.

$$\nabla f = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

There was a formula for the numerical gradient kernel but I didn't really understand it at all.

5.6.2 Laplacian

Second derivative. Difference of first derivative

$$\begin{aligned} L &= (f(x+1) - f(x)) - (f(x) - f(x-1)) \\ &= f(x+1) + f(x-1) - 2f(x) \end{aligned}$$

Much sharper than gradient because it uses more data and gives more information. However, it also amplifies noise.

5.7 Prototype

Do python code and **make a presentation**. You can use a library, but you have to implement your own ideas and data.

Chapter 6

3D Signal Processing

Video data is of the form $f(x, y, t)$. Video processing algorithms operate on a frame by frame basis. Frame rate is in frames per second (fps), and our algorithm needs to be fast enough to not decrease the fps of the video. 60fps is the usual benchmark.

6.1 Tracking

Track a Region of Interest (ROI). Follow the object as it traverses. The two challenges of this are the fact that the target moves, and we need the computation time to be very fast.

6.1.1 Function

The main paper we use takes advantage of a correlation filter. We use a filter such that, given the input signal, returns a sharp signal at the center of our green box and nothing everywhere else.

Given the next frame (slight change in data), the filter should still return a sharp dot at the square, but it will be blurry. Then, we recalculate the filter for the new box at frame 2.

$$f_1 \& s_1 \Rightarrow w_1 \rightarrow f_2 * w_1 = s_2 \rightarrow f_2 \& s_2 = w_2 \rightarrow \dots$$

6.1.2 Filter Itself

We use a circular filter

$$\min_w \sum_i (w^T x_i - y_i)^2$$

6.2 Ridge Regression

$$\min_w \sum_i (w^T x_i - y_i)^2 + \lambda ||W||^2 \qquad w = (X^H X + \lambda I)^{-1} X^H y$$

Matrix inversion is an $O(N^3)$ operation, so how to we reduce the time complexity of this algorithm?

6.2.1 Circulant Matrices

All circulant matrices can be diagonalized with a DFT.

$$\hat{w} = \frac{\hat{x} \odot \hat{y}}{\hat{x}^* \odot \hat{x} + \lambda}$$

6.2.2 Kernelized Correlation Filter

idk it was on the slides.

6.2.3 Lab 8: Digit Recognition Via Machine Learning

You get an extra 3% by doing this lab. THIS IS INDIVIDUAL WORK. No lab quiz. Due by the end of the semester, but need to demo at TA office hours.

Chapter 7

Machine Learning

We're going to make a program that input hand-written digits and outputs the ASCII code of the number we think matches.

7.0.1 Rule Based Programming

- Program a list of command sor rules
- Know every step required to complete programs task
- Stupid machine learning

7.0.2 Actual Machine Learning

- Algorithm figures out rules via trial and error and large well described/labeled datasets
- Can discover rules and correlations that humans cannot see

7.1 Regression

Predict a continuous value based off a dataset with 2 float values per item

You figure out how close the model is to being perfect by using the least squared error.

7.2 Classification

Predict one of many discrete options based off a dataset with an item and a class per item.

7.3 Clustering

A form of unsupervised learning where you divide data into certain classifications **without labels**.

7.4 2D Image Recognition

Because we're in an ECE course, we treat the AI as a mapping from input to output.

7.5 Gradient Descent

You take the partial derivation of the cost function to find the minimum.

7.6 Multiple Features

You do the same gradient descent and have a similar cost function except you're in a multi-dimensional space now.