Assignment A4: Sounds and Spectra

Note: Unlike the other assignments, there is just one part here: A4. Also unlike the other assignments, there is no self-check or draft. You will just submit your final version.

Put your Name and Case ID here

Use `464-A4-yourcaseid.ipynb` for your notebook name.

If you will use your own data files, put these in a folder named `464-A4-yourcaseid-files`, which should also contain your notebook. Make sure this folder contains all the files needed to reproduce your results.

Overview

This assignment focuses on different types of sound structures and their relation to perceptual qualities.

References

The following material provides both background and additional context. They are linked in the Canvas page for this assignment. Refer to these for a more detailed explanation and formal presentation of the concepts in the exercises. Note that the readings contain a lot more material than what's needed for the exercises. Feel free to skim or ignore the parts that aren't directly relevant.

- Schnupp, Nelken, and King (2012). *Auditory Neuroscience*. Ch. 1 Why Things Sound the Way They Do.
- Müller (2015) *Fundamentals of Music Processing*. Ch. 2 Fourier Analysis of Signals.

Learning objectives

- Explain and illustrate sound structures such as harmonics, fundamental frequencies, pitch, envelope, and timbre.
- Write functions to synthesize sounds with specific structures from parameters.
- Explore the extent to which analytic sound features relate to perceptual qualities.
- Explain the concepts of analysis and synthesis.
- Compute Fourier transforms using the fast Fourier transform and interpret the results.
- Compute and plot the signal power spectrum.

Exercises

1. Spectral Structure

These questions will focus on the concepts of harmonics, spectra, and how they relate to some basic aspects of our perception of pitch and timbre.

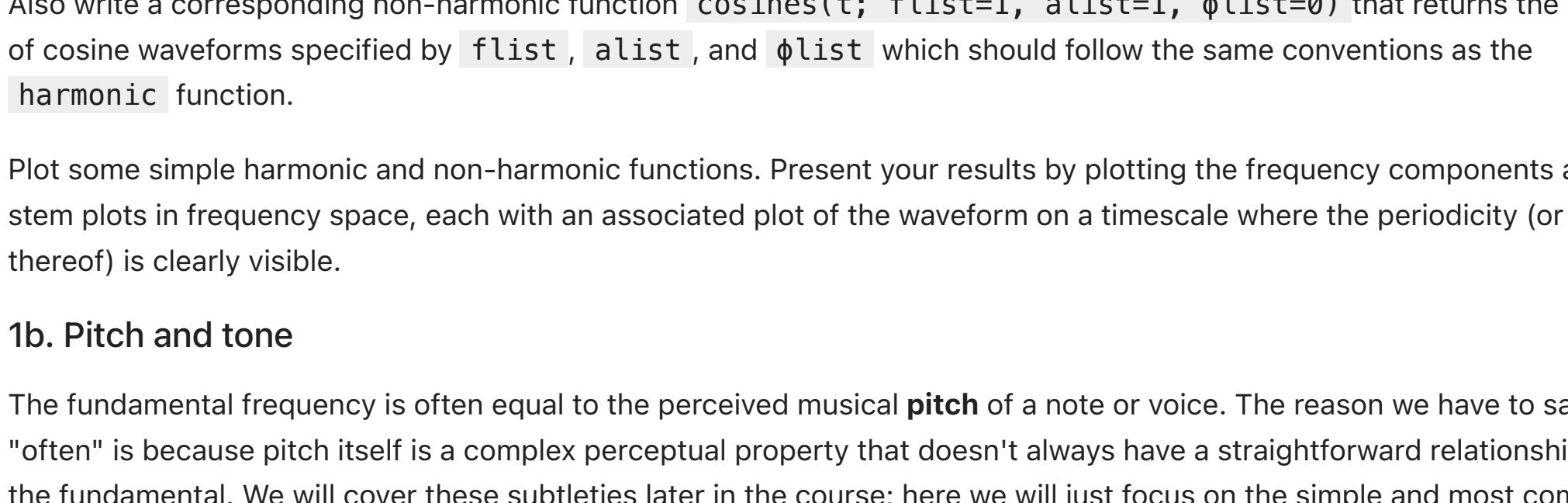
1a. Harmonics

Harmonics are combinations of frequencies each of which is an integer multiple of a **fundamental** frequency:

$$f_n = n f_1$$

Harmonics arise from the modes of vibration of a physical objects or resonant cavities. The first harmonic, f_1, is the fundamental frequency, the second harmonic is $f_2 = 2f_1$, and so on. A sound of composed of just a single frequency is called a **pure tone**.

The figure below shows a 100 Hz fundamental with an octave harmonic. The plot on the left is the spectrum, the plot on the right is the amplitude waveform.



Like we have done previously, we will first write functions to generate signals with known properties that we can use to test our analysis methods. Here, we will write two functions: one to generate a harmonic waveform and another to generate a sum of arbitrary frequencies.

Write a function `harmonic(t; f1=1, alist=1, ϕlist=0)` that returns the value at time `t` of a harmonic function with fundamental frequency `f1` (in Hz) and an array `alist` whose values specify the amplitudes of each harmonic starting with the fundamental. `ϕlist` specifies the phases (in radians) for each harmonic. A scalar value for `alist` indicates a single frequency (the fundamental) and specifies its amplitude. A scalar value for `ϕlist` specifies a common phase. Use the cosine function to generate each harmonic, so that the phases of the frequencies match that of the Fourier transform, which will we be using later.

Also write a corresponding non-harmonic function `cosines(t; flist=1, alist=1, ϕlist=0)` that returns the sum of cosine waveforms specified by `flist`, `alist`, and `ϕlist` which should follow the same conventions as the `harmonic` function.

Plot some simple harmonic and non-harmonic functions. Present your results by plotting the frequency components as stem plots in frequency space, each with an associated plot of the waveform on a timescale where the periodicity (or lack thereof) is clearly visible.

1b. Pitch and tone

The fundamental frequency is often equal to the perceived musical **pitch** of a note or voice. The reason we have to say "often" is because pitch itself is a complex perceptual property that doesn't always have a straightforward relationship to the fundamental. We will cover these subtleties later in the course; here we will just focus on the simple and most common case.

Use your functions above to generate two different waveforms with the same harmonic amplitudes (use several harmonics to get a richer sound) but different fundamental frequencies. Write the result to a .wav file, so you can listen to it. Plot your results like above.

Note 1: You should write to the .wav file using a higher frequency, e.g. 44.1 kHz, which is the standard for audio, but when you plot it, either plot a brief segment or plot it at lower sampling frequency, e.g. one sample per pixel.

Note 2: If you play a synthesized sound with an amplitude that doesn't start at zero, you will often hear a click or a pop due to the abrupt onset. To avoid this, you can specify a common phase of $-\pi/2$ to convert these functions to sum sine waves so that the waveform starts at zero. Below we will apply envelopes to the sound to produce more natural sounding onsets.

Do the sounds at with different fundamentals sound similar? How you describe the sound *tone*, i.e. how it sounds? Observe that, in general, we perceive the sound with harmonics *holistically* i.e. as a single percept and not as a combination of the individual frequencies of the harmonics.

1c. Timbre

The pattern of harmonics, i.e. their relative amplitudes, is related to how a voice or musical note sounds, e.g. the difference between the sound of a piano vs that of a violin. This is called **timbre** or musical tone. Like pitch, timbre is a complex perceptual phenomenon. Here, we just want to focus on how it relates to harmonics.

Use your code from the previous question to experiment with different harmonic patterns. Find two set of values that have a similar sounding tone for different fundamentals, and, conversely, sound very different for the same fundamental.

Present your results like you did above by plotting the harmonic patterns as stem plots in frequency space with associated plots showing a few periods of the waveforms.

1d. Coding Challenge (optional exploration)

Write a function that constructs a waveform from a list of notes, their durations, and a timber function for the note harmonic structure. The notes (which should include rests) could be specified in terms of fundamental frequency, musical notation, or musical intervals in a specified key. The challenge is to design an interface and implementation that is both flexible and elegant with minimal coding.

2. Noise

The opposite of sounds with harmonic structure are sounds composed of noise, i.e. random amplitude fluctuations. Different types of noise are characterized by the distribution of energy across frequency bands.

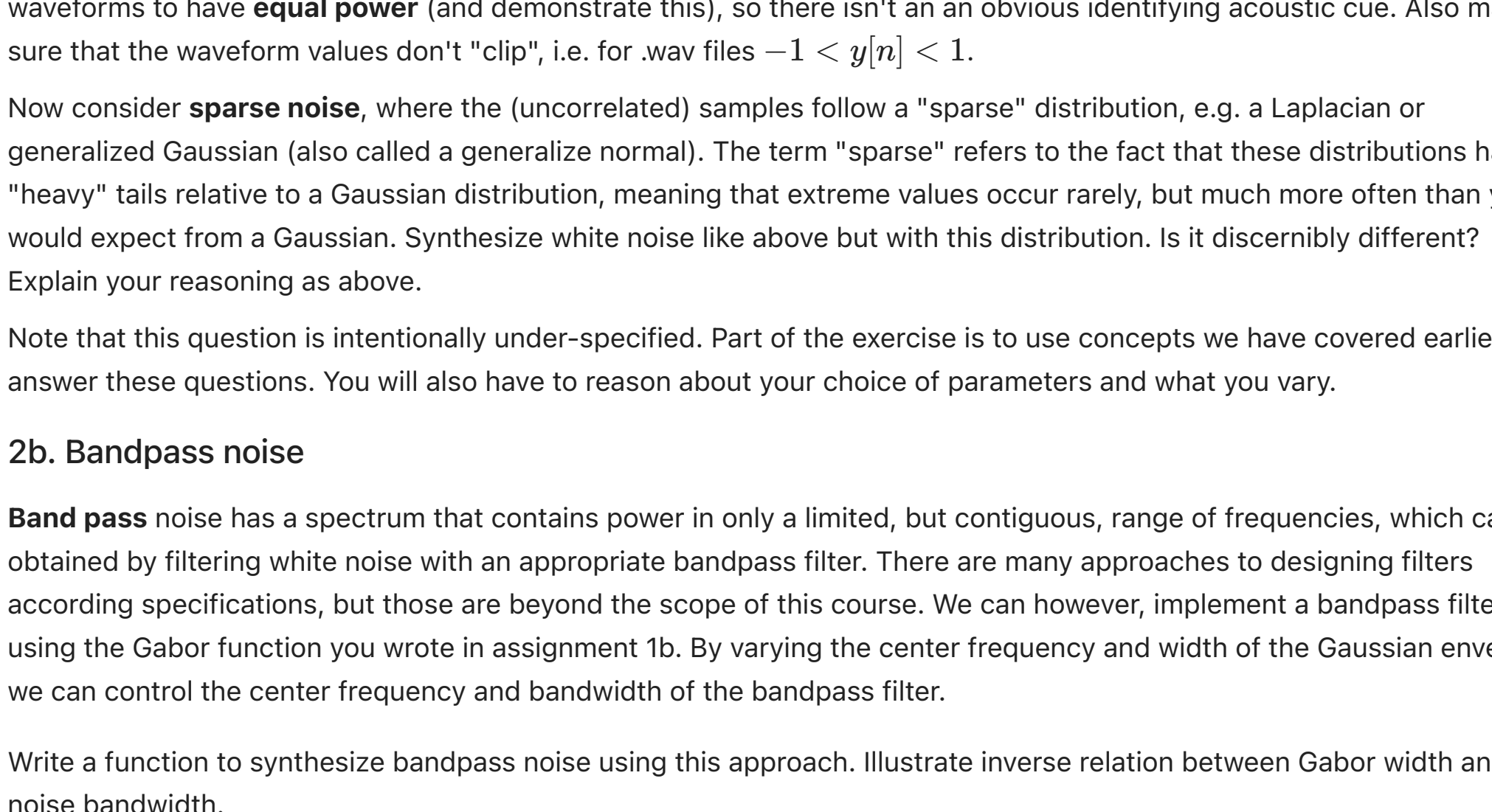
2a. White noise

In this exercise you will generate three types of white noise, plot histograms of their amplitude distributions, and write them to sounds to see if you can hear any differences. A first approximation of cochlear computation is to encode sound in terms of their component spectral energies. The motivating question is: Do two waveforms with equal spectra sound the same?

In "**white noise**", the spectral energy is distributed evenly across frequencies. (Aside: This is somewhat analogous to white light, which is where name is derived from, but since we only have three types of cones, what we perceive as the color "white" does not necessarily have a flat spectrum; the light only has to result in equal activation of the three cones, but even that is subject to the complexities of color perception.)

For digital signals, another property of the white noise is that the samples are **uncorrelated**. Thus, you can have **uniform white noise** or **Gausian white noise** for amplitudes that are distributed as a uniform or Gaussian distribution, respectively, but still have uncorrelated samples.

Examples of uniform and Gaussian white noise.



Do you think it would be possible to hear the difference these two types of noise? Reason from the assumption that the auditory system encode sounds by breaking it to frequencies, and then test and illustrate this idea empirically. Scale your waveforms to have **equal power** (and demonstrate this), so there isn't an an obvious identifying acoustic cue. Also make sure that the waveform values don't "clip", i.e. for .wav files $-1 < y[n] < 1$.

Now consider **sparse noise**, where the (uncorrelated) samples follow a "sparse" distribution, e.g. a Laplacian or generalized Gaussian (also called a generalize normal). The term "sparse" refers to the fact that these distributions have "heavy" tails relative to a Gaussian distribution, meaning that extreme values occur rarely, but much more often than you would expect from a Gaussian. Synthesize white noise like above but with this distribution. Is it discernibly different? Explain your reasoning as above.

Note that this question is intentionally under-specified. Part of the exercise is to use concepts we have covered earlier to answer these questions. You will also have to reason about your choice of parameters and what you vary.

2b. Bandpass noise

Band pass noise has a spectrum that contains power in only a limited, but contiguous, range of frequencies, which can be obtained by filtering white noise with an appropriate bandpass filter. There are many approaches to designing filters according specifications, but those are beyond the scope of this course. We can however, implement a bandpass filter using the Gabor function you wrote in assignment 1b. By varying the center frequency and width of the Gaussian envelope, we can control the center frequency and bandwidth of the bandpass filter.

Write a function to synthesize bandpass noise using this approach. Illustrate inverse relation between Gabor width and noise bandwidth.

Pitch and periodicity

In the first exercise you have seen that harmonic sounds are **periodic**, where the **period** is the time required to complete one cycle of the repeating waveform. For pure tones and simple harmonic sounds, the subjective perception of **pitch** is usually determined by the fundamental frequency. Pitch is an example of an invariant perceptual quality, e.g. human voices or musical instruments can have very different harmonic structures, and still be perceived as having the same pitch.

How could we determine the pitch from the waveform? One approach is to estimate the periodicity by looking for lags (or time delays) where the signal is most similar to itself. A common approach is to use **correlation** as a proxy for similarity.

3. Correlation

The **auto-correlation** of a (discrete time) function is defined as

$$R_{xx}[n] = x[n] \star x[n] = (x \star x)[n] = \sum_k x[k-n]x[k]$$

where k ranges over the overlapping extents of $x[k-n]$ and $x[k]$ (or equivalently $-\infty, +\infty$, where the functions are defined to be zero outside of their extents). The resulting function is the inner product of time-shifted versions x with itself. The time shift n is called a **lag**, which for discrete time functions is in samples.

Cross-correlation is often used to determine the locations of a shorter signal (or feature) g in a larger waveform. It is defined as

$$R_{gx}[n] = g[n] \star x[n] = (g \star x)[n] = \sum_k g[k-n]x[k]$$

where k is a discrete ranges over the overlapping extents of g and x. Local maxima in the cross-correlation are the locations where g most closely correlated x.

This operation looks similar to convolution, but it is not (see the mathematical challenge below). It is common notation to omit the argument $[n]$ and write $g \star x$ or $h * x$. To keep the operations distinct, it is also common to write convolution with a "$*$" ("asterisk") operator and correlation with a "$\star$" ("star") operator. We have also used a different variable, g, for the operand to emphasize the distinction from an impulse response function, which is commonly denote by h.

Commutivity. Unlike convolution, correlation is *not* commutative, i.e. we have

$$h * x = x * h, \text{ but}$$
$$g \star x \neq x \star g$$

In other words, the cross-correlation operation of g on x is not the same as that of x on g. For this reason, it is helpful to keep the operation order conceptually distinct and read the cross-correlation function $(g \star x)[n]$ as "g *scans* x", i.e. x remains stationary while g shifts with changing values of n to produce the value of function $R_{gx}[n]$ at lag n.

Normalization. Auto-correlation and cross-correlation are often normalized to remove the dependence on signal scale. In the case of auto-correlation, it can be shown that the maximum value of $(x \star x)[n]$ is at the origin and falls off for any shift, i.e $n \neq 0$. Therefore, a natural normalized auto-correlation function can be obtained by dividing the definition above by its value at zero

$$\rho_{xx}[n] = \frac{\sum_k x[k-n]x[k]}{\sum_k x[k]x[k]} = \frac{\sum_k x[k-n]x[k]}{||x||^2}$$

Here, we have noted that the value of R_{xx} at zero is equivalent to the square of the vector norm $||x||$

$$||x|| = \sqrt{\sum_k (x[k])^2} = \sqrt{R_{xx}[0]}$$

Similarly, normalized cross-correlation is defined as

$$\rho_{xy}[n] = \frac{\sum_k x[k-n]y[k]}{||x|| \cdot ||y||}$$

This can be recognized as an inner product of two unit vectors (with zeros for any missing dimensions): the normalized cross-correlation is the cosine of the angle between vector x across shifts and the vector y.

Note that auto- and cross-correlation are also commonly defined (equivalently) using random processes and the statistical concept of correlation, but we omit that discussion here.

3a. Auto-correlation vs self-convolution

Show that auto-correlation is symmetric around zero but self-convolution (i.e. $x * x$) is not. Use both a mathematical argument and an illustration.

3b. Cross-correlation vs convolution

The definition of cross-correlation is similar in form to that of convolution, but they are not equivalent. Describe how the two definitions differ and show that $g[n] \star x[n] = g[-n] * x[n]$. Why is the kernel "backwards" in convolution, i.e. what computation is it modeling? Use this insight to explain why using convolution in A3b to do matched filtering was not entirely correct (assuming you used straight convolution without flipping the kernel). Why was (would) this approach still able to work? When would the operations be equivalent?

3c. Cross-correlation vs sum-squared error

Another commonly used measure of similarity is squared error. Write a mathematical expression for computing the sum squared error (SSE) of a signal $g[n]$ at all locations in a waveform $x[n]$? Contrast this with cross-correlation. When would it make more sense to use one vs the other?

3d. Mathematical challenge (optional exploration)

Prove that convolution is commutative but correlation is not for both the discrete forms above and the continuous forms below

$$(g \star x)(t) = \int_{-\infty}^{\infty} g(\tau - t)x(\tau)d\tau$$
$$(x * h)(t) = \int_{-\infty}^{\infty} x(\tau)h(t-\tau)d\tau$$

Provide an illustration (or plot from your implementation below) where this result is visually obvious.

4. Implementation and Application

4a. `convolve`, `autocorr`, and `crosscorr`

Write functions to implement `convolve(x, y)`, `autocorr(x; normalize=true)`, and `crosscorr(x, y; normalize=true)`. (Note that you can just modify your implementation of convolve from A3b, but here we are not using the `h0` argument.) Assume the input vectors are zero outside their limits. All functions should return the output of applying the operation over the entire overlapping extent, i.e. the length of the output should be the sum of the lengths of the inputs minus one. The normalization keyword for `autocorr` and `crosscorr` indicates the output should be normalized as described above.

4b. Pitch estimation

Use your auto-correlation function to estimate the pitch of the harmonic signals you synthesized above in the first exercise. You should plot the auto-correlation function and show how you derive the pitch estimate.

4c. Estimating time-delay

Generate a signal of random noise and a time-shifted version of it. Use cross correlation to estimate time delay between the two signals. Plot the cross-correlation function and show how you derive the delay in time units (e.g. microseconds) from the lag in samples.

4d. Experimental challenge: Lateralized noise (optional exploration)

Continuing exercise 3c, generate a stereo waveform of about a second in duration composed of time-shifted noise with a range of time delays between $-700\ \mu s$ and $700\ \mu s$. Output the result to a sound file and describe your experience of listening to the sounds. Are they clearly lateralized?

For a bigger challenge try and synthesize a waveform with continuously varying delay from completely left to completely right and back in a loop. Is the sound clearly localized? Do you perceive sound motion?

5. Spectral Analysis

Spectral analysis decomposes a waveform signal into its frequency components. This is most commonly done using the **Fourier transform** which maps a signal in the **time domain** to a representation in the **frequency domain**. This done with calculus using an analytic function. In this course, we will be working with sampled signals, so we will use The Discrete Fourier Transform (DFT), which maps length-N signals to a set of N frequency components. Note that the number of free parameters has to be the same in both representations for the transform to be invertible. Also note that we assume the discrete signal is length N, but if the signal is periodic, then it is sufficient to represent just a single period.

Below we provide some background but the exercises will involve only calling library functions to perform the forward and inverse transforms. This material is provided mainly for conceptual context.

Basis representation

A Fourier transform is an instance of the more general concept of **basis representation**. With the right set of functions, we can represent a signal (to an arbitrary precision) in terms of a **linear combination** of **basis functions**

$$y(t) = \sum_k c_k \Psi_k(t)$$

where each of the basis functions, i.e. the vectors $\Psi_k(t)$, are scaled by **coefficients** c_k.

Equivalent representation 1: sine and cosine amplitudes

In a Fourier representation, the basis functions are sinusoids, and there are three standard ways of expressing this basic idea. The first representation consists of the coefficients in a sum of sines and cosines of different frequencies. The function (which is in continuous time, we assume we are sampling above Nyquist) is

$$y(t) = \frac{a_0}{2} + \sum_{k=1}^{N} a_k \cos(kt) + b_n \sin(kt)$$

Note that we have a discrete set of frequencies, indexed by k. Here the frequency is just k for simplicity (which is radians per second), but we could write it as $2\pi k$ to express it in cycles per second. The Fourier transform computes the values of the coefficients [^1]. This gives a pair of coefficients for each frequency.

Equivalent representation 2: amplitude and phase

We can apply the trigonometric identity

$$a\cos(\theta) + b\sin(\theta) = A\cos(\theta + \phi),$$

where $A = \mathrm{sgn}(a)\sqrt{a^2 + b^2}$ and $\phi = \arctan(-b/a)$, to write an equivalent representation in terms of the amplitude and phase

$$y(t) = \frac{a_0}{2} + \sum_{k=1}^{N} A_n \cos(kt + \phi_k)$$

Equivalent representation 3: complex exponentials

A third representation uses Euler's formula

$$\exp(j\theta) = \cos(\theta) + j\sin(\theta)$$

to write the Fourier series as a sum of complex exponentials. (Here we have used j to indicate an imaginary number, which is the standard in engineering because i is often used for an index.) Letting ω_0 be an arbitrary fundamental frequency (now in Hz), we then have

$$y(t) = \sum_{k=-N}^{N} c_n \exp(j\, 2\pi\, \omega_0 k\, t)$$

Each term in the series is a harmonic of the fundamental frequency ω_0.

Although it might seem opaque if you're unfamilar with complex exponentials, this representation is the standard representation used in engineering due to its mathematical elegance and ease of deriving the forward and inverse transforms using exponentials.

Standard packages for the fast Fourier transform (**FFT**), which we will use below, return the coefficient representation as a vector of complex numbers. The complex coefficients are equivalent to the sine and cosine amplitudes as the Cartesean coordinates in the complex plane, i.e. $a + ib$; the polar form of the coefficients, i.e. $re^{i\theta}$ is equivalent to the magnitude and phase representation. To reiterate, the basic idea to understand is that the Fourier transform decomposes a waveform into oscillatory components.

Inverse transforms an efficiency

Another important idea is that, under very general conditions (e.g. integrable), the transform can represent *arbitrary* signals and is fully invertible, i.e. we can recover the exact waveform from the coefficients using the **inverse transform**. Another way to think about it is that the transform does not loose any information about the signal. The expectation is that the transform will yield in some sense a "simpler" or more efficient representation, but that need not always be the case. For example, a step function can be represented with just the time and amplitude of the step whereas a series requires summing an arbitrary large number of waveforms to achieve an arbitrarily accurate approximation (for a continuous function, it is exact for a discrete function), but in the limit the series will converge to the signal. Even with discontinuous signals, such as steps, the series will converge to the midpoint of the discontinuity.

Notes:

How to compute the coefficients is too far afield for this course (see EECS 246 or 313), but the standard algorithm is the fast Fourier transform or "FFT".

5a. Forward and inverse FFT

Here you will use a standard package FFT to do spectral analysis, e.g. FFTW.jl for julia or scipy.fft for python. Both packages have all the functions used below and use the same function names. Consult the documentation for usage details.

Show that for a random vector the inverse of the forward transform yields the original vector with zero summed difference (modulo numerical imprecision).

As discussed above, by default `fft` returns a complex valued vector of coefficients. Passing these to `ifft` will perform the inverse transform but return a complex result, but with zeros for the imaginary values. You can convert these back to a real numbers for the comparison or use the real-signal variants (`rfft` and `irfft`) which will do this automatically.

5b. Spectral analysis

Use the FFT to spectral analysis of the harmonic signals you synthesized above in exercise 1. Plot the frequency magnitude as a function of frequency. The component frequencies can be obtained using the `fftfreq` (or `rfftfreq`) convenience functions. There isn't a corresponding function to obtain the coefficient magnitudes, but without worrying about edge cases, for both real-valued signals the component magnitude of Fourier coeffient Y_k is

$$a_k = 2|Y_k|/N$$

where N is the length of the vector. Show that the spectral magnitudes approximately match those of your synthesized waveform. Why might the spectrum not exactly match the amplitudes used to generate the signal? Can you synthesize a waveform composed of a small number of haromnics that spectral analysis can decompose exactly?

Note: The "edge cases" mentioned above means that the equation is correct if the frequencies at zero (i.e. the signal mean) and Nyquist have zero magnitude. If those frequencies are present, omit the factor of two for each, and note that the Nyquist frequency is only a component when N is even.

Grand challenge: Sound synthesis

To what extent do the acoustic features above describe the perceptual qualities of a real sound?

Obtain a clean recording of musical or harmonic sound. Then try to synthesize it using and increasing number of acoustic features, e.g. first haromnics, then include harmonic amplitudes, then the envelope, etc. How much like the real sound does your synthesized sound sound at each stage? If at the end, it still does not sound like the real sound, what might be missing?

Tests and self checks

For this last assignment there are no self-checks or draft submission, so you will just be submitting your final version.

Submission Instructions

Please refer to the Assignment Submission Instructions on canvas under the Pages tab.

In [ ]: