Discrete Fourier Transforms (DFT): a brief introduction
ENGN/PHYS 225 (March 29, 2021)

Foreword

Fourier's big idea is that we can **synthesize** any *periodic* function $f(t)$ from a ***series*** of cosine and sine waves. We have to mix just the right amount of each harmonic frequency to properly construct $f(t)$. On the flip side, we can we **analyze** $f(t)$ to determine what frequencies are present and with what intensity. That's all fun and good for analytical functions of a continuous variable x or t (or whatever) we write on paper. But what about real world signals that aren't actually continuous?

This document briefly relates how we extend Fourier's theory into the world of *discrete data*. The DFT works on data that is sampled at discrete time intervals. This encompasses all real-world data ever collected—video data, electrical measurement, pressure, temperature, you name it. The DFT is very commonly used to ascertain what frequencies are present in the data (e.g. See Figures 1 and 2. In fact, the DFT might be one of the single most broadly useful techniques anywhere in STEM? Because everything in the natural world tends to oscillate!

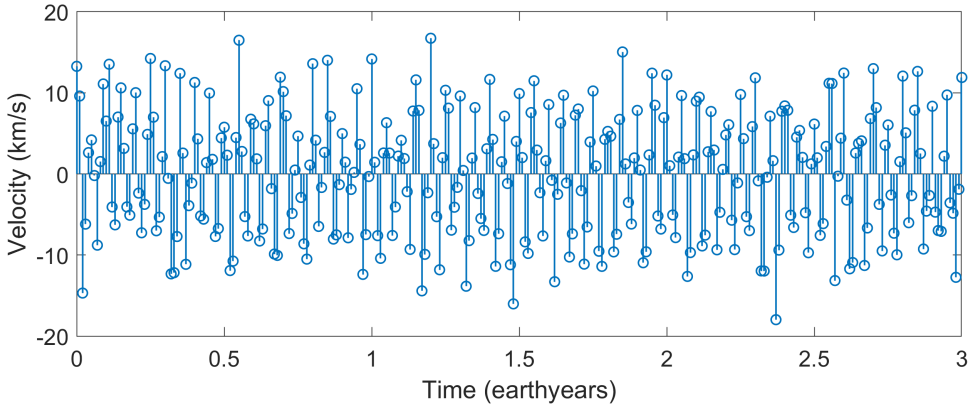

Figure 1: Hunting for planets via oscillations in orbital velocity discretely sampled in time.

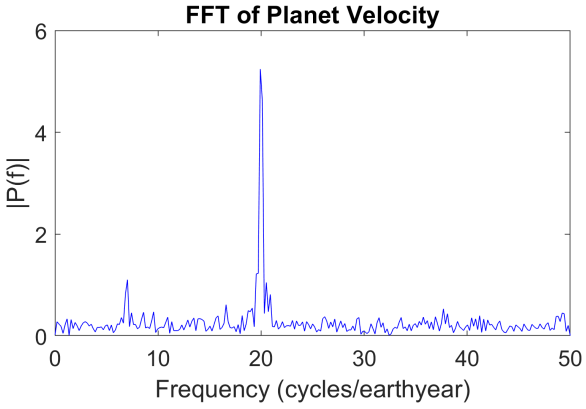

Figure 2: Corresponding single-sided Fourier Spectrum with peaks at 7 and 20 cycles/earthyear.

Fourier Series

Recall our definition of a Fourier series. Assume $f(t)$ is a $2T$ periodic function:

$$f(t) = a_o + \sum_{1}^{\infty} a_n \cos\left(\frac{n\pi}{T}t\right) + b_n \sin\left(\frac{n\pi}{T}t\right) \tag{1}$$

Or in complex exponential form:

$$f(t) = \sum_{-\infty}^{\infty} c_n e^{\frac{in\pi}{T}t} \tag{2}$$

The Fourier coefficients are given by:

$$c_n = \frac{1}{2T}\int_{-T}^{T} f(t) e^{\frac{-in\pi}{T}t}\, dt \tag{3}$$

We can think of c_n as the correlation between $f(t)$ and an oscillation occurring at an angular frequency $n\pi/T$. The c_n's tell us how much the of the nth harmonic oscillating at a frequency $n\pi/T$ contributes to the overall function $f(t)$. More precisely, the intensity of the nth harmonic is given by $2|c_n| = \sqrt{a_n^2 + b_n^2}$.
Again, all of the math above works only for periodic functions of a continuous variable that we can *write down on paper*.

Discrete Fourier Transforms

Assume we have N samples of a signal acquired at a constant sampling period Δt_s (e.g. See Figure 1), acquired at a sampling frequency of $f_s = 100$ samples/earth year). Equivalently, the sampling rate is given by $f_s = 1/\Delta t_s$. The integer $k = 0, 1, 2, \ldots$ indexes the sample number. The kth sample in the data set occurs at time $t_k = k\Delta t_s$. For example, imagine we are acquiring video frames at 1/24 s. The samples are recorded at $t = 0, 1/24, 2/24, 3/24, \ldots, (N-1)/f_s$ s. Note the total duration of the data recorded is $N\Delta t_s$ seconds.

In contrast to the Fourier series formulation, instead of assuming $2T$ periodicity the DFT makes assumes the fundamental frequency is the time duration of the recording ($N\Delta t_s$). In other words, for the DFT we assume there the fundamental frequency is $1/N\Delta t_s$ Hz or $2\pi/N\Delta t_s$ rad/s. Of course, we also get the constant "dc average value" component too at 0 Hz (no oscillation). Thus, the DFT computes a frequency spectrum sampled at a discrete frequencies of:

$$\omega_n = 0,\ 1 \times \frac{2\pi}{N\Delta t_s},\ 2 \times \frac{2\pi}{N\Delta t_s},\ \ldots,\ n \times \frac{2\pi}{N\Delta t_s}, \ldots,\ (N-1) \times \frac{2\pi}{N\Delta t_s}$$

Note the frequency resolution of the DFT is $\delta f = \frac{1}{N\Delta t_s} = \frac{f_s}{N}$ Hz. For example, if we have $N = 1000$ samples acquired at $50Hz$, then $\delta f = 50/1000$ Hz $= 0.05$ Hz. The more samples, the finer the resolution will be.
Below, we briefly explore the DFT (looking in the frequency domain to analyze the underlying oscillating components of a signal that varies in time) and the inverse DFT (synthesizing a time domain signal by adding up a bunch of oscillating functions).

DFT

We can decompose a time-domain signal to using the DFT to analyze the underlying oscillating components as follows for $n = 0, 1, \ldots, N-1$:

$$F[n] = \sum_{k=0}^{N-1} f[k] e^{-i\omega_n t_k} = \sum_{k=0}^{N-1} f[k] e^{-\frac{i2\pi n}{N} k} \tag{4}$$

Carefully compare equations 4 and 3. They look awfully similar, no? The $F[n]$ are just like the c_n terms from the Fourier Series! They are also complex numbers. We typically visualize the Fourier ***frequency spectrum*** by plotting $2|F[n]|$ vs. frequency. They key is to look for prominent spikes in the frequency spectrum. Recall that the coefficient for $F[n]$ correspond to an underlying oscillation frequency of:

$$\omega_n = 2\pi n f_s/N \text{ (rad/s)} = n f_s/N \text{ (Hz)}$$

Thus, the frequencies at which we can analyze the sampled signal $f[k]$ depends on the rate at which we sample our data f_s and the number of data points we collect N.

Observe that sum in Eqn 4 just expresses the correlation coefficient between the real world signal $f[k]$ and a signal oscillating at frequency ω_n. Pretty intuitive, eh!

Again, we typically visualize the Fourier ***frequency spectrum*** by plotting $2\,|F[n]|$ vs. frequency. They key is to look for prominent spikes in the frequency spectrum.

Inverse DFT

We can do the synthesize a signal in time by adding up (superposing) a bunch of oscillating functions using the ***Inverse DFT***.

$$f[k] = \frac{1}{N} \sum_{n=0}^{N-1} F[n] e^{i\omega_n t_k} = \frac{1}{N} \sum_{n=0}^{N-1} F[n] e^{\frac{i2\pi n}{N} k} \tag{5}$$

Observe that Eqn 5 is very similar to 2. Note that we are summing over n, so we are summing up a bunch of oscillating functions, with angular frequency given by ω_n, to synthesize the time domain signal. It's worth mentioning yet again that coefficients $F[n]$ represent the contribution from a signal oscillating at angular frequency of $\omega_n = 2\pi n f_s/N$ (rad/s).

Practical Considerations

Given the forms for the DFT above, there are some extremely important properties we must consider from a lab practical perspective.

1. **Matlab implemention**: Matlab can be used to easily compute the DFT. The function is called `fft()`. This stands for "Fast Fourier Transform", which is a clever algorithm used to efficiently compute the DFT. Make sure to check out the documentation and examples in glorious detail on the mathworks website: `https://www.mathworks.com/help/matlab/ref/fft.html`.

2. **Frequency resolution** Recall that given discrete data, we get a discrete frequency spectrum. The resolution of the spectrum is given by $\delta f = \frac{f_s}{N}$. The more samples, the finer the resolution. Somewhat counterintuitively, the lower the sampling rate, the finer the resolution.

3. ***Single sided vs. double-sided spectrum***. You'll often hear this term—and see it in matlab documentation. It turns out the FFT is symmetric about half the sampling frequency $f_s/2$. See figure 3. For instance, if we sample data at $f_s = 100$ Hz, then the FFT mirrors about 50 Hz. That means the highest signal frequency we can "see" is $f_s/2$. This is often called the ***Nyquist rate***.

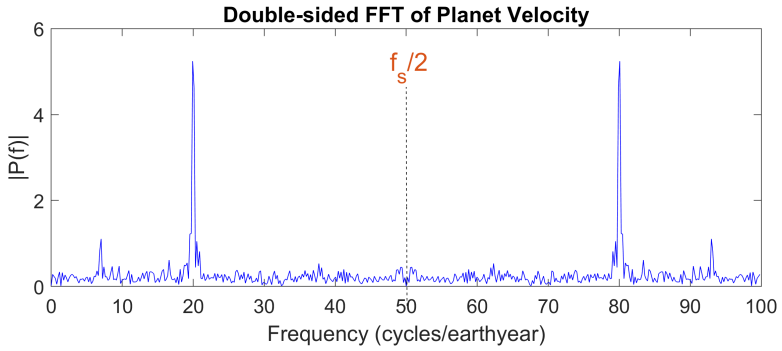

Figure 3: Double-sided frequency spectrum. The frequency spectrum mirrors about half the sampling frequency. Note the left half is the same information displayed in Fig 2.

4. **Nyquist/Shannon Sampling Theorem**:

   In a nutshell, Nyquist and Shannon said that in order to accurately determine the frequency content of a signal actually oscillating at a rate of f, we have set the sampling frequency to be at least twice as fast. ***This is one of the most important practical theorems you'll ever encounter in life***!

   $$f_s \geq 2f$$

   Good lab practice is to actually sample at about $10\times$ the highest frequency you suspect is present in the signal. Don't live on the hairy edge. Figure 4 illustrates this idea of sampling "fast enough".

5. **Nyquist Rate**: Good ole Henry Nyquist and pals made a similar statement - the reverse side of the same coin. In a nutshell, if we sample our signal at a rate f_s, then the highest frequency f we can analyze in the signal is $f = f_s/2$. Again, because we need to be able to see peaks and troughs in order to determine a full oscillation.

6. **Aliasing:** A signal that actually oscillates at a frequency higher than than the Nyquist rate will appear in the frequency spectrum as a lower-frequency. Thus, it is sort of a "fake", hence the term alias. In general, we want to avoid aliasing in the lab.

These properties all stem from the periodicity of the function:

$$s_n[k] = e^{i\omega_n t_k} = e^{\frac{i2\pi n}{N}k}$$

which appears the definition of the DFT and its inverse above.

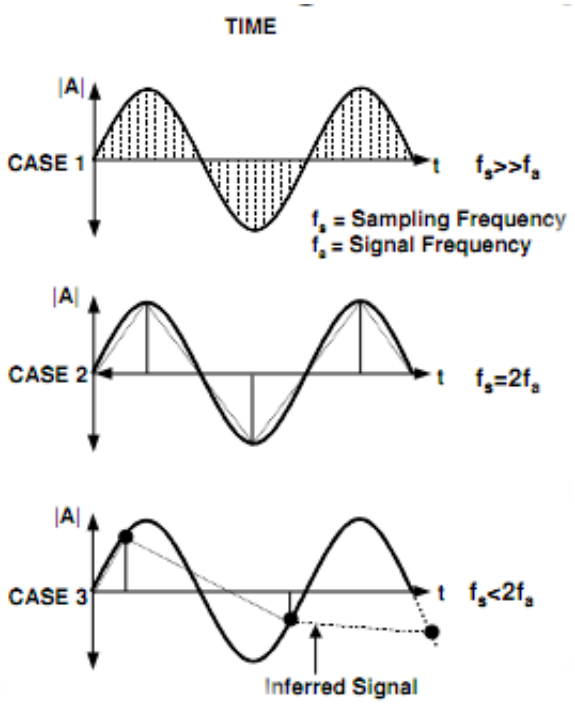

Figure 4: Sampling a signal "fast enough" ($f_s > f$), just fast enough (Nyquist samping rate $f_s = f/2$); and too slowly ($f_s < f/2$). Image adapted from:`http://www.expertsmind.com/questions/define-sampling-below-the-nyquist-rate-30174749.aspx`

In particular, if we choose $n = N$, then $s_n[k] = 1$ for all values of the integer time index k. This isn't any oscillation at all! If we choose $n = N/2$, then $s_n[k] = (-1)^k$, which oscillates $+1, -1, \ldots$ as time index k advances. This would catch the peaks and troughs only, nothing in between. This is just barely sufficient to analyze an oscillation at $\omega_{N/2} = \pi f_s$ or equivalently at $f_s/2$ which is the Nyquist rate! In fact, it can bit can be shown that:

$$F[n] = F^*[N - n]$$

Thus, the frequency spectrum $|F[n]|$ mirror images around the frequency $\omega_{N/2}$! The so-called ***single-sided spectrum***, considers only $n = 0, 1, \ldots, N/2$ terms. This is also why in the matlab examples, you'll see that we only consider the first half of the coefficients computed using the `fft` function.

Again, this result also implies that the only valid frequencies ω_n we can analyze are the ones corresponding to $n = 0, 1, 2, \ldots, N/2$. Frequencies that are higher than this are aliased. For instance, imaging there is a 60 Hz electromagnetic wave contaminating a sensitive electrical recording sampled at $f_s = 100$ Hz. The Nyquist rate is 50 Hz. 60 Hz is 10 Hz above the Nyquist rate, so it will appear in the mirror imaged (aka *aliased*) at 40 Hz (10 Hz below 50 Hz).