# PHY407 Lab04

Fall 2023

## Background

**Numerical computations of Fourier transforms:** See §§ 7.1 and 7.2 of the textbook for general statements about the Discrete Fourier Transform (DFT). In particular, the DFT finds the coefficients $c_k$ for a set of discrete function values $y_n$ through

$$c_k = \sum_{n=1}^{N-1} y_n \exp\left(-i\frac{2\pi kn}{N}\right) \tag{1}$$

If the function if real, then the coefficients in the range $N/2 \to N$ are just the complex conjugates of the coefficients in the range $0 \to N/2$, i.e., $c_{N-i} = c_i^*$. This means we only have to calculate $\sim N/2$ coefficients for a real function with $N$ values.

Since the Fourier coefficients are complex, we usually plot their absolute value vs. $k$ to make an amplitude plot. The $c_k$'s with large magnitudes represent periodicities with $k$ values which dominate the signal. The $k$ value is related to the period $n_{cyc}$ of the signal through: $2\pi k n_{cyc} = 2\pi N$. Therefore, $n_{cyc} = \frac{N}{k}$.

Although doable, you should never really code an FFT yourself. While a good FFT algorithm does not need to exceed a few lines, the number of mistakes one can do is dizzying. Instead, there are standard library functions in Python (as well as other programming languages) for the FFT. The page `https://numpy.org/doc/stable/reference/routines.fft.html` is full of useful functions, which you may use in this lab.

**Amplitude and phase issues:** One common problem in using FFTs to analyze scientific data is finding the *correct* amplitude and phase. For example, when you Fourier transform a wave with amplitude of 2 units, you need the Fourier transform to return a wave with an amplitude of 2 unit (not $2\pi$ units, or $2/N$ units, or some other scalar multiple). As there is often ambiguity about which scale factor is used by which built-in FFT routine it is a good idea to check the results with simple functions. For example, to check the output of the `fft` function against the `ifft` function make a sine function and FFT that. Compare your *known* input parameters (amplitude, phase) against the output of the FFT.

```python
import numpy as np
def cosine_wave(time, amplitude, period, phase):
    """Generate a sine wave with known amplitude, period, and phase"""
    # What do you mean by "phase" exactly?
    return amplitude * np.cos((time/period + phase) * 2 * np.pi)

t = np.arange(0, 100, .1)
y = cosine_wave(t, 2, 20, 0.)

#plt.plot(t, y)
fft1 = np.fft.fft(y)
fft2 = np.fft.ifft(y)
fft3 = np.fft.rfft(y)
fft4 = np.fft.irfft(y)
```

```
amp1 = np.abs(fft1)
amp2 = np.abs(fft2)
amp3 = np.abs(fft3)
amp4 = np.abs(fft4)

print(amp1.max(), amp2.max())
print(amp3.max(), amp4.max()) # Are any of these numbers what you expect?
```

You should find that one FFT command includes a scale factor equal to the length of the array, while the other doesn't. There are options to the FFT function that allow you to change this behaviour. **If you change it, make sure you are consistently using the options**. You will also find that the "amplitude" only includes half of the input signal. This happens because there are positive and negative phase components and each carries some of the signal – in this case half. The distinction between the positive and negative components can sometimes be ignored for one dimensional FFTs, in which case the rFFT functions are useful. For two dimensional FFTs the two components contain information about the *direction* the signal is travelling.

Finding the phase can also be ambiguous. The Fourier transform in `numpy.fft` returns a complex number $r_k + ij_k$, and the phase can be found from trigonometry using the `numpy.arctan2` function. You should experiment to make sure you understand what phase is being returned by numpy.fft .

**Mars physics:** Your prof loves making fun of Elon Musk's impractical Mars colonization plans. Yes, she has actually done some reading about conditions on Mars, and in this lab she's making you do some too.

The atmosphere of Mars, much like Earth and Venus, is heated daily by the Sun as it passes overhead, but the heating is not a simple function of time. While there is a smooth increase and decrease in heating during the day (approximating a sine function with local noon at the peak) the night "heating" is effectively constant — once the Sun has set, the same amount of sunlight reaches the ground until morning (none). As a result of this heating and cooling pattern, the surface temperature on Mars doesn't cycle in a single 24 hour sine function, but as a series of sine functions with periods of 24,12,8,4,3 hours and so on, all integer factors of 24 hours. Similarly the surface air pressure on Mars goes through a similar daily cycle composed of waves of shorter periods.

As an added complication, the air on Mars is 95% carbon dioxide, which freezes during Martian winter. A third of the Martian atmosphere literally freezes out of the sky twice a year, contributing to the "seasonal weather".

The data file `msl_rems.csv` you are given contains surface air pressure measurements from the Mars Science Laboratory (instruments aboard a cute little rover driving around in a crater) taken 24 times each Mars day. Since the Mars day is slightly longer than an Earth day (by 2.7%) it is usually called a Sol, and 1/24 of a Sol is a Martian "hour" (where this hour has almost 100 extra seconds). Sometimes, the Mars year is broken up into heliocentric longitude, measured as the number of degrees around the orbit since the start of the Martian year at $L_s = 0$.

## Questions

1. **Getting rich on the stock market**

   If your prof's research budget is running short, she might need to play the stock market in order to fund her next dark matter experiment. The file `sp500.csv` contains the closing value (second column) for each business day (first column) from late 2014 until 2019 of the S&P 500 stock index (a measure of the growth of the largest 500 companies in the United States).

   (a) Write code to load the second column of data from `sp500c.csv` (closing value), and plot it against "business day number" starting at 0. (The data in the file only exists for "business days", so it skips weekends and holidays; therefore if you plot the "actual day number", there will be gaps that mess up your analysis.) **Submit your code.**

(b) Write code to calculate the coefficients of the FFT of the data (closing value vs business day) using the `rfft` function from `numpy.fft`, and test that inverting the results with `irfft` returns the original data. **Submit your code, and explanation and output of whatever test you implemented.**

(c) Suppose we want to analyze long-term trends in the market, by *removing* any variation with a period of 6 months or shorter. Write code to set some of the Fourier coefficients (figure out yourself how many, and which ones) to zero, and perform the inverse FFT. This is an example of a "low–pass filter", keeping all frequencies *lower* than a threshold. Plot the result on the same graph as the original (closing value vs business day) data; adjust the linestyle as needed to make the two lines clear, and add a legend. Explain the shape of the filtered data (the first and last datapoints may be weird, but you can ignore these.)

**Submit the code, plot, and explanation.**

2. **Fun on Mars**

The file `msl_rems.csv` contains seven columns. The first column is time (in Sols elapsed since the first Martian mission) such that integer times are local "midnight". The second is heliocentric longitude, which you can ignore. Then there are five pressure data columns, with units of Pa: `A`, `B`, `C`, `D`, `E`.

(a) Write code to: take an array of times (in Sol) and an array of corresponding pressure values (in Pa), calculate the FFT of the dataset, and return the amplitude (Pa) and period (Sol) and phase (Sol) of any specified mode in the dataset. Test this function on a simple dataset by generating a cosine wave of known amplitude, period, and phase. **Submit your code, and explanation and output of your test**

(b) Apply your code from the previous part to column `A`, which contains test data. (Optional: it may be helpful to plot the pressure vs time data to make sure you see a wave.) Plot the amplitude and phase as a function of period (a log period scale is needed, and there should be a single peak in the FFT at a period of approximately 1 Sol). Report the period, amplitude and phase of the dominant mode. **Submit your plots and written answer.**

(c) Now apply your code to dataset `C`. This contains cleaned pressure data (meaning some noise has been removed and gaps have been filled) using only the diurnal (once per day) and sub–diurnal waves. Plot the amplitude and phase as a function of period; you should see 4 peaks in the FFT. Report the period, amplitude, and phase of each of these modes. **Submit your plots and written answers.**

(d) Dataset `D` contains less clean (meaning it includes more noise) pressure data than `C`. Repeat the previous part, and compare your answers: briefly describe how the noise affects the FFT. **Submit your plot and written answer.**

(e) Now apply your code to dataset `E`, which contains cleaned-up pressure data including the seasonal pressure cycle, short term 'weather' cycles, and diurnal tides. Plot the amplitude as a function of period. Find the 4 largest modes with periods longer than 100 Sols – these are the seasonal cycles – and report their periods and amplitudes. Also find the 4 largest modes with periods shorter than, or just above, 1 Sol – these are the Martian "tides", responding to the rotation of Mars and the heating from the Sun – and report their periods and amplitudes. **Submit the plot and written answers.**