PHY407 Computational Lab 5
Fourier Transforms

This lab includes exercises on Fourier analysis in the time domain, where the Fourier components correspond to different time frequencies or periods, and Fourier analyis in the spatial domain, where the Fourier components correspond to different wavelengths or wavenumbers.

- In Q1, we will look at frequency analysis of functions of time.
- In Q2 we will look at a signal that depends on both a spatial coordinate and time. We will Fourier decompose in the spatial coordinate.
- In Q3 we will carry out a two dimensional Fourier analysis in two spatial dimensions.

Computational Background

- As described in the text, the Discrete Fourier Transform finds the coefficients c_k for a set of discrete function values $y_n = f(x_n)$ through:

$$c_k = \sum_{n=0}^{N-1} y_n \exp\left(-i\frac{2\pi kn}{N}\right), \tag{1}$$

  with $c_k = c^*_{N-k}$ for real $f(x)$.

- The inverse Discrete Fourier Transform gets the function values y_n from the coefficients c_k:

$$y_n = \frac{1}{N}\sum_{k=0}^{N-1} c_k \exp\left(i\frac{2\pi kn}{N}\right) \tag{2}$$

- Because of the relation $c_k = c^*_{N-k}$, there are only $N/2+1$ independent Fourier coefficients, which means we don't have to calculate as many Fourier coefficients.
- Since the Fourier coefficients are complex, we usually plot their absolute value vs k to make a Fourier Transform 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:

$$\left(\frac{2\pi kn_{cyc}}{N}\right) = 2\pi \Rightarrow n_{cyc} = \frac{N}{k} \tag{3}$$

- The DFT requires $O(N^2)$ evaluations of $\exp\left(-i\frac{2\pi kn}{N}\right)$. This is a lot and would not make the DFT useful for many operations. Luckily, there is a sneaky way to rearrange the DFT to make it much faster. When this faster algorithm is implemented, you require only $N\log_2 N$ evaluations of $\exp\left(-i\frac{2\pi kn}{N}\right)$. This is significantly fewer and hence significantly faster. The algorithm is called the "Fast Fourier Transform".

- The text goes through the details of the implementation of the FFT, but you should never really code this yourself. Instead, there are standard library functions in python (as well as other programming languages) for the FFT.

- The code `fft_ts.py` discussed in Lecture 5, Part 1, is a useful starting point for this lab. It involves the spectral analysis of a sound signal and implements the DFT as per the textbook and then also calls the FFT routines in python. The line in the script

  ```
  freq = arange(N/2+1)*2*pi/T
  ```

  is useful for laying out the dimensional frequency axis for a time series sampled at N points over a time interval T. The code also looks for specific frequencies to zero out, namely the dominant frequencies in the signal. I think that if you understand this code it will make the rest of the lab easier.

- **Re-revisiting the relativistic spring (for Q1a):** We will be looking again at the relativistic spring system; here's some computational background. We will be doing longer intergrations than in Lab01, and since the Euler method is unstable (as we'll discuss in the ODE labs) we will need a more stable method - namely, the Euler-Cromer method. The difference between the two methods is as follows: while for the Euler method, we use

  $$\begin{aligned} v^{n+1} &= v^n + dt\, a(x^n, v^n) \\ x^{n+1} &= x^n + dt\, v^n, \end{aligned}$$

  where $a(x^n, v^n)$ is the acceleration at time n, for Euler-Cromer we use

  $$\begin{aligned} v^{n+1} &= v^n + dt\, a(x^n, v^n) \\ x^{n+1} &= x^n + dt\, v^{n+1}, \end{aligned}$$

  In other words, when updating the position x^n, we now use the updated velocity v^{n+1} instead of the previous velocity v^n. This little trick turns out to stabilize the algorithm.

  A code that implements Euler-Cromer is supplied in the file

  ```
  Lab05_RelativisticSpring_EulerCromer.py
  ```

  Please use this code for Q1a. It is likely you will have to adjust the timestep `dt` and the length of integration `T` to obtain a stable estimate of the spectrum.

- **Loading data from a text file (for Q2):** If you have data in a text file you can load the entire file into an array using a command like:

  ```
  y=loadtxt("filename.txt")
  ```

  where filename.txt is the name of your file. If the data consists of N rows and M columns, then y will be an array with N rows and M columns. If you want to work with the first column of the array, you can refer to it as:

```
y[:,0]
```

The ":" means take all the rows, the "0" means in the first column. Similarly, if you wanted the 5th column you would use y[:,4] or if you wanted the 3rd row you would use: y[2,:]. All of these options give you 1D arrays, essentially picking out the column or row of interest to you. If you want to know the shape of the array (i.e. how many rows and columns) you can use the "shape" attribute. For example, for an N by M array:

```
print y.shape
```

would return (N,M). This is an array itself. If you wanted to just know the number of rows in y you could use:

```
print y.shape[0]
```

which picks out the first element of the shape array. Similarly, if you wanted to know the number of columns you would call the [1] element.

- **Using the `.wav` sound file format (for Q1b).** The `.wav` format is a standard digital sound format used by computers. Typically if you click on a `.wav` format an audio player will open up and play the sound in the file. The particular sample that you will be processing in Q1b is a stereo file with two channels, Channel 0 and Channel 1. The data format is `int16`, which is a 16 bit integer format. When you write to the new file you will need to write to that format. Here is some code to read and write that file – please adapt it to your needs:

```
#The scipy.io.wavfile allows you to read and write .wav files
from scipy.io.wavfile import read, write
from numpy import empty
#read the data into two stereo channels
#sample is the sampling rate, data is the data in each channel,
#  dimensions [2, nsamples]
sample, data = read('input_file.wav')
#sample is the sampling frequency, 44100 Hz
#separate into channels
channel_0 = data[:,0]
channel_1 = data[:,1]
N_Points = len(channel_0)
#... do work on the data...
#this creates an empty array data_out with the
#same shape as "data" (2 x N_Points) and the same
#type as "data" (int16)
data_out = empty(data.shape, dtype = data.dtype)
#fill data_out
data_out[:,0] = channel_0_out
data_out[:,1] = channel_1_out
write('output_file.wav', sample, data_filt)
```

In the code above, `sample` is the sampling rate of the data in Hz, which is typically 44100 Hz for audio data, and we have converted the data to the `int16` datatype in the two lines before the write statement.

- **Shifting the Gaussian (for Q3):** Here is some code to produce the gaussian image for Q3, referring to the text. In the code below, 'rows' is the number of rows in the data and 'cols' is the number of columns. These will have to be defined beforehand from the shape of the blurred image.

```
for i in range(rows):
     ip=i
     if ip>rows/2:
        ip -=rows   #bottom half of rows moved to negative values
    for j in range(cols):
        jp=j
        if jp>cols/2:
            jp -= cols   #right half of columns moved to negative values

        gauss[i,j]=exp(-(ip**2+jp**2)/(2.0*sigma**2))   #compute gaussian
```

Physics Background

- **Re-revisiting the relativistic spring (for Q1a):** We have done quite a bit of work on the relativistic spring system: in Lab01 we calculated position $x(t)$ and velocity $v(t)$ for various initial conditions in the classical and relativistic regime, and in Lab03 we calculated the system's period across a wide range of energies. The character of the oscillation really changed with the energy: for example the shape of the position $x(t)$ as a function of time is sinusoidal in the classical harmonic oscillator regime, but it is more triangular in the in the relativistic regime where the particle spends much of its time going close to the speed of light. In this exercise, we'll see how these changes impact the Fourier spectrum of $x(t)$, and relate the Fourier spectrum to our calculations of the period from Lab 2.

- **Signal analysis and filtering (For Q1b).** In question 1b we will take a sample sound file, looks at its Fourier spectrum, filter it for desired audio characteristics, and write the results to a new file in order to listen to the impact of our filtering. The sound file is in the `.wav` format described in the computational background. We will employ a very simple filtering method in which we will zero out the components that we want to supress. In particular, we produce a low-pass filter such that all Fourier coefficients above a cutoff frequency f_c will be set to zero: if $\hat{s}(f)$ represents the fourier coefficients of a time series $s(t)$, the low-pass filter is here defined as

$$\hat{s}_{lp}(f) = \begin{cases} 0, & f > f_c, \\ \hat{s}(f), & f \leq f_c. \end{cases} .$$

This is actually not a very good method for good quality sound filtering, but will give you a flavour of how filtering might work.

- **Looking at atmospheric data (for Q2):** In atmospheric physics, the *sea level pressure* (SLP) is the surface air pressure that would be found once the Earth's surface was raised or lowered to the geoid (the notional mean sea level). Daily maps of SLP show the location of low pressure centres (associated with cloudy weather and rain) and high pressure centres (associated with fair or sunny conditions). Such centres typically propagate eastward in the temperate latitude. SLP data (in the form of a kind of melded data and modeling product) are available from many research centres around the world. In Q2, we have extracted some SLP data for you to analyze:

  - Quantity: SLP with a mean value subtracted
  - Units: hPa (hectopascals, i.e. 100 Nm^{-2})
  - Coordinates: Latitude: 50°S. Longitude: $[0°, 360]$, sampled every 2.5°. Time: The first 120 days of 2015, sampled every day.
  - Source: NCEP Reanalysis

  To keep the analysis simple, we have extracted SLP around one latitude circle located in the Southern Hemisphere. Weather here is often observed to progress in coherent wave trains and you will use Fourier analysis to extract these wave trains. You will use the idea that SLP can be decomposed into a series of waves of the form

  $$A(t)\cos(m\lambda + \phi(t)),$$

  where m is wavenumber for longitude λ, and $A(t)$ and $\phi(t)$ are time dependent phase factors. The dataset shows that different wavenumbers m are characterized by different propagation characteristics, as we'll describe in the problem. Some of these differences are predicted by theories in atmospheric dynamics.

- **Image Deconvolution (for Q3):** Image deconvolution is one example of the use of cross-correlation techniques to get info from data that is jumbled up. Newman pages 322-325 takes you through the mathematics of how it works, so you should read through that to get an idea of what is going on, but to do the lab, you really only need the result near the bottom of page 324:

  $$\tilde{b}_{kl} = KL\tilde{a}_{kl}\tilde{f}_{kl}$$

  where $\tilde{b}_{kl}$ are the Fourier coefficients of the convoluted image, $\tilde{a}_{kl}$ are the Fourier coefficients of the pure image, $\tilde{f}_{kl}$ are the Fourier coefficients of the point spread function that convoluted the image and K and L are the length and width of the image.

  So if you take the Fourier transform of the convoluted image and divide it by the Fourier transform of the point spread function, then you get the Fourier transform of the pure image. You can then inverse-transform the pure image Fourier transform to get the pure image.

Lab Instructions

For grading purposes, only hand in solutions to the following parts. Ensure that your codes are well commented and readable by an outsider:

- Q1: For part a), hand in plots and written answers only. For Part b), hand in plots, written answers, and code.
- Q2: Plots and written answers only.
- Q3: Unblurred photo and code.

Lab Questions

1. **Fourier Transforms in the time domain (40% of the lab).**

   (a) **Re-revisiting the relativistic spring:** The purpose of this exercise is to practice extracting a frequency spectrum from a time series you generate from one of your simulations. Do the following:

   - Starting with the script described in the background material, simulate the relativistic spring system for three cases we looked at in Lab01:

     ```
     x0 = 1#m
     x0 = xc #m
     x0 = 10*xc #m
     ```

     To get a good estimate of the spectrum, generate a long time series that contains several oscillations (at least 10 periods for each case). Make sure the time step is small enough to get stable solutions - I found that Euler Cromer was unstable in the relativistic regime for too long a time step. Hand in your plots for these simulations.
   - Then find the Fourier transform of the solutions for position $x(t)$. This will give Fourier components which can be written $\hat{x}(f)$, where the hat indicates a Fourier component and f is the frequency. To plot the three cases on the same plot, given that the amplitude of the oscillation is very different in each case, plot the scaled quantity, $|\hat{x}(f)|/|\hat{x}(f)|_{\max}$, where the max indicates the maximum value of the amplitude. This quantity has a maximum value of 1 for each spectrum.

     Describe the differences in the spectrum between the three cases. Do they make sense to you?
   - Now do the same for velocity $v(t)$ instead of position (i.e. plot the spectrum of the velocity Fourier coefficients). Are there any qualitative differences between the spectra for $v(t)$ and for $x(t)$?
   - This next part is tricky: I would like you to quantitatively compare these results obtained from the simulation to the predicted period of the oscillations using Gaussian quadrature integration, as in Lab03, Equation (4) (see Lab03 Q2). In other words, you now have two estimates of the characteristic frequencies, one obtained from the spectrum, and another obtained from Equation (4) of Lab03. The simplest way to do this is to plot a vertical line at the location of the frequency $1/T$, where T is the period obtained from Equation (4) of Lab03.

What makes this tricky is that I found that it was difficult to get agreement between the two methods to within more than about 10% when you are in the relativistic regime. This is not a textbook problem, so if you have insight into this discrepancy you can let me know - perhaps it can be the basis of your course project.

(b) **Filtering an audio recording:** In this exercise I will ask you to low pass filter the sound in the sound file provided with this exercise. Do the following:

- Play the sound file `GraviteaTime.wav` to make sure your computer's audio can process it.[1] If you are having trouble with this please see Paul or Oliver.
- Now, adapting the code provided in the background material, produce a plot of the data in the file as a function of time. Produce one plot for each channel, and mark the time axis in seconds, accounting for the sampling rate given by `sample` in Hz as described in the background material. Hand this plot in.
- Now focus down this plot to a small segment of the total time series about 20-50 ms long. This will let you better see the impact of filtering. You do not need to hand in this particular plot.
- Now implement a filter in which all frequencies greater than 880 Hz are set to zero. Do this by Fourier transforming each signal to the frequency domain, setting the right coefficients to zero, then Fourier transforming the result back to the time domain. Hand in the following plots: the amplitude of the original Fourier coefficients, the amplitude of the filtered Fourier coefficients, the original time series over the short interval described above, and the filtered time series for this interval. Do this for both channels and use `subplot` to reduce the number of separate figures. The filtered time series should be a smoothed version of the original time series.
- Now output the resulting time series, for each channel, into a new `.wav` file, as shown in the background material. For our reference, call the file `GraviteaTime_lpf.wav`. Listen to it - you should hear a bass-heavy version of the tune. Hand in this file. Your grader will get to listen to your creation!

2. **Analysis of sea level pressure in the Southern Hemisphere for summer 2015 (30% of the lab).** Please carefully read the background material relevant to this problem. Your job will be to Fourier decompose in the longitudinal direction the sea level pressure at 50°S, which is provided in the file `SLP.txt`. The files `lon.txt` and `times.txt` provide the longitudes λ (in degrees) and the times (in days, starting from January 1, 2015) for this data. This dataset can be read and initially plotted by the following commands, as long as all the files are in the same folder as your code:

```
from numpy import loadtxt
from pylab import contourf, colorbar,xlabel,ylabel,title
SLP = loadtxt('SLP.txt')
Longitude = loadtxt('lon.txt')
Times = loadtxt('times.txt')
contourf(Longitude, Times, SLP)
```

[1] The music file is from Joel Franklin's *Computational Methods for Physics.* It is likely that you and your lab neighbours will get sick of hearing this tune, so keep it down a bit.

```
xlabel('longitude(degrees)')
ylabel('days since Jan. 1 2015')
title('SLP anomaly (hPa)')
colorbar()
```

The plot you obtain will show the time evolution of the SLP at this latitude for the first 120 days of 2015.

(a) To get started, make sure the code above works; hand the plot in.

(b) Now, extract the component of SLP corresponding to Fourier wavenumber $m = 3$ and to $m = 5$ for this data. Create filled contour plots in the time-longitude domain for the data thus extracted, and hand in these plots. What are some important qualitative characteristics of these plots? The theory of atmospheric wave propagation suggests that wave disturbances can propagate but the speed of propagation will depend on the scale of the wave, with shorter waves travelling eastward faster than longer waves. Is this roughly consistent with what you see? Hand in these plots.

(c) Now estimate the speed and direction of propagation for the case $m = 5$. To get the speed in m/s, the circumference of the latitude is given by $2\pi r_e \cos(\theta)$, where θ is latitude and r_e is the radius of the Earth. Note that these results will be different for other times of year and other years. Your estimate can be graphical or more quantitative - just state which method you use.[2]

3. **Deconvolution for image processing (30% of the lab).** Do Exercise 7.9 parts (a)-(c) on pages 325-326. Hints:

- I recommend using "imshow" to plot the data with a gray colormap.
- To centre the gaussian at the origin of the figure you will need to shift the formula for the Gaussian on page 324. Some code to create the appropriate gaussian is given in the background material above.
- For part (c) There is some important information at the top of page 326. Make sure to turn the page! You don't have to do part (d).

[2]This question focuses on the largest differences between the Fourier components. You can explore other wavenumbers m and you will find that the behaviour does not follow a simply predicted trend across scales.