AMATH 582 Homework 1

Yiyun Dong

Department of Physics
University of Washington
Email: yiyund@uw.edu
Github: yeewantung

Abstract

The project investigates the 20 ultrasound measurements to find the position of the marble in the intestine of the dog. We implemented Fast Fourier Transform and filtering to reveal the trajectory of the marble.

1 Introduction and Overview

Fourier transform [7] decomposes a time-domain function into a frequency distribution in the frequency domain. It is widely used to extract frequency information from the function. In order to deal with the discrete measurements in practice, several algorithms are introduced to conduct Discrete Fourier transform on computing services, such as Fast Fourier transform (FFT) [6].

Fourier transform is widely applied in different field. It is shown to be useful in solving the differential equations, finding the conjugates in Quantum Mechanics, Fourier transform spectroscopy and signal processing [7]. Spectral analysis is an important aspect of signal processing. It is analogous to producing a score from a piece of music [1]. The goal is to start with a signal and identify the strength of the frequency components that make up the signal.

Since the frequency information could be easily extracted by Fourier transform, Fourier transform is also used to focus on a single frequency and localize a signal in the frequency domain. This is extremely useful in radar detection and medical imaging, when the position of an object need to be determined.

In this project, a dog accidentally swallowed a marble. The marble is travelling in the digestive system of the dog and putting the dog in danger. The vet used ultrasound to detect the marble and achieved 20 spatial measurements over time. If we want to save the dog, we have to develop a method to find the position of the marble out of 20 measurements. We use the Fourier transform and filtering technique to process the measurement data and find the trajectory of the moving marble in the intestine of the dog. It is expected to see the vet apply our result to break the marble inside the dog body.

2 Theoretical Background

As we learned from our textbook [2], Fourier transform decomposes a time-domain signal into a frequency distribution in the frequency domain.

$$\begin{aligned} F(k) &= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{-ikx} f(x)\, dx \\ f(x) &= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{ikx} F(k)\, dk \end{aligned} \tag{1}$$

In practice, Fourier transform of data is implemented in computer by an algorithm called Fast Fourier transform [6].

When dealing with signals that are constantly moving in the time domain, noise and environmental disturbances often prevent us from visually identifying the location of the signal. In class, we learned a way to solve this problem using Fourier transform and filter function. The frequency of the signal from an object is a single and stable frequency. Different from the signal from an object, randomly occurring noise has a random frequency, random phase and random intensity. The intensity of noise in different measurement data are destructively canceled out by taking the average of the Fourier transforms of measurements. Then after averaging the Fourier transforms of different measurement data, the position corresponding to the signal frequency ends up having the highest intensity.

As we also learned in the lecture, Fourier introduced the concept of representing a given function $f(x)$ by a trigonometric series of sines and cosines [2].

$$f(x) = \frac{a_0}{2} + \sum_{i=1}^{\infty} (a_n \cos nx + b_n \sin nx) \quad x \in (-\pi, \pi]. \tag{2}$$

It tells us that the Fourier transform is linear and decomposes a function into a continuous series of sinusoidal waves. It inspires us that we can use a filter function to retain information near the signal frequency and filter out information other than the signal frequency. And hence only the spatial information of the needed signal is kept after the inverse Fourier transform. This process could filter out the information corresponding to the noise and environment disturbances. The position corresponding to the signal in space can therefore be revealed by applying inverse Fourier transform to the filtered information.

3 Algorithm Implementation and Development

1. Average the Fourier transforms of the measurements.
2. Find the wave number corresponding to the maximum intensity of the averaged Fourier transform data.
3. Apply a filter function to the Fourier transform of each measurement and do inverse Fourier Transform.
4. Find the location corresponding to the maximum intensity of each inverse Fourier Transform measurement.
5. Save the locations as a trajectory of signal over time and make a 3D plot.

Algorithm 1: Algorithm of Fourier Transform and Filtering

Import data from `Testdata.mat`
for $j = 1 : 20$ **do**
  Fourier Transform j from `Undata`
end for
`Utn` = Take average of the Fourier Transform of measurements
Find the wave number k corresponding to `Maximum(Utn)`
Make a filter function out of the wave number k
for $j = 1 : 20$ **do**
  Fourier Transform j from `Undata`
  Apply filter to Fourier Transform j
  `Ufn` = Inverse Fourier Transform j
  Save the location x corresponding to `Maximum(Ufn)` in trajectory
end for
Plot the trajectory

4 Computational Results

The dog swallowed the marble and there are a total of 20 measurements using ultrasound to measure the position of marble in the intestine of dog. These measurements are based on the distribution of signal intensity in three spatial dimensions. Due to noise and environmental disturbance, it is not obvious which signal is from marble. So I adopted the above method, and averaged these measurement data after Fourier transforms. The obtained frequency information is as shown in the Figure 1.

Notice that the wave number possessing the strongest intensity is at $[k_x, k_y, k_z] = [1.8849555921538759, -1.0471975511965976, 0.0]$. This is the wave number of the ultrasound signal reflected by the marble.

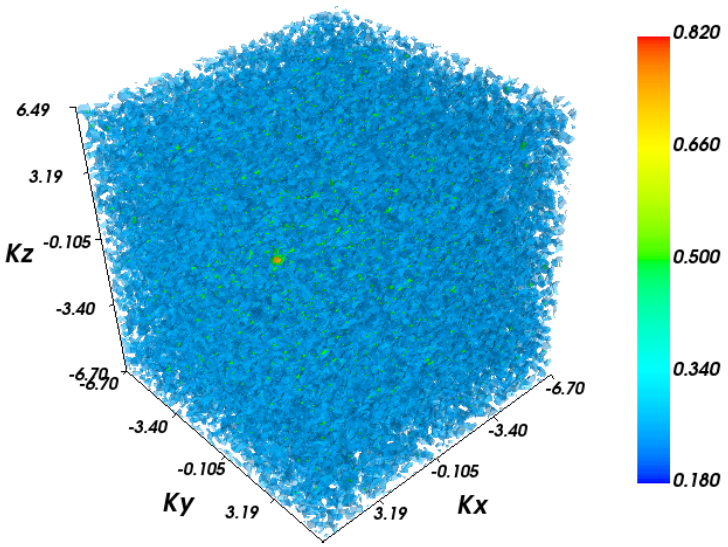

Figure 1: Fourier transform of the ultrasonic data averaged over measurements. Blue: low intensity. Red: high intensity.

The filter function I chose is a Gaussian function with center at the wave number $[k_{c,x}, k_{c,y}, k_{c,z}] = [1.8849555921538759, -1.0471975511965976, 0.0]$ corresponding to the maximum intensity. The variation of the Gaussian function is choose to be $\sigma^2 = 2.5$. The explicit function form of the filter is,

$$\text{filter}(k_x, k_y, k_z) = \exp(-0.2 \cdot ((k_x - k_{c,x})^2 + (k_y - k_{c,y})^2 + (k_z - k_{c,z})^2)) \tag{3}$$

After filtering the frequency, the information corresponding to the ultrasound signal of marble is left and inverse Fourier transformed back to the spatial space. Then trajectory of the marble is obtained, as shown in the Figure 2.

If the vet breaks the marble at the position of the 20^{th} measurement, then he should target the position $[x, y, z] = [-5.625, 4.21875, -6.09375]$.

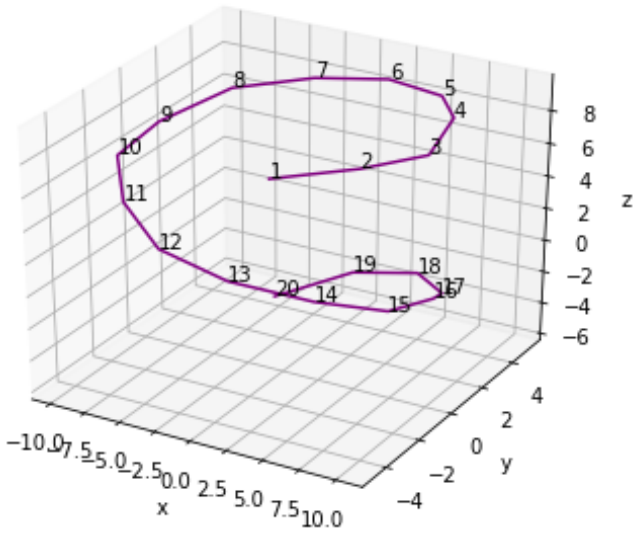

Figure 2: Trajectory of the marble

Measurement	x	y	z
1	4.6875	-4.6875	9.84375
2	8.4375	-2.8125	9.84375
3	10.3125	-0.46875	9.375
4	8.90625	2.34375	9.375
5	6.09375	4.21875	8.90625
6	1.40625	5.15625	8.4375
7	-3.28125	4.6875	7.96875
8	-7.5	3.28125	7.5
9	-9.84375	0.9375	6.5625
10	-9.84375	-1.40625	6.09375
11	-7.03125	-3.28125	5.15625
12	-2.8125	-4.6875	4.21875
13	1.875	-4.6875	3.28125
14	6.5625	-3.75	2.34375
15	9.375	-1.875	0.9375
16	9.84375	0.46875	0.
17	7.96875	2.8125	-1.875
18	4.21875	4.6875	-2.8125
19	-0.9375	5.15625	-4.21875
20	-5.625	4.21875	-6.09375

Table 1: The position of marble in measurements $1 - 20$

5 Summary and Conclusions

In this project, in order to help the veterinarian break the marble in the dog's body, we need to find the specific location of the marble. Ideally, ultrasonic measurements possess information on the location of objects. However, in reality, noise and environmental disturbances prevent us from identifying the actual position of the object if we only observe ultrasound images. Therefore, removing the noise and highlighting the signal become our goal. This involves two steps. The first step is to make use of the randomness of the noise signal, namely, the fact that the average of the noise signal over time is zero. The second step is to use the filter function to filter out the frequency other than the object signal. Because only the frequency information about the object signal is retained, most of the time-domain information after the inverse Fourier transform comes from the object signal. The specific location of the object by these two steps.

The choice of filter function is a problem in this process. According to Heisenberg's uncertainty principle, the frequency and position of an object cannot be accurately measured simultaneously. If too few frequencies are kept, the time domain information of this object will be overly spread out in the spatial distribution. If too much frequency information is kept, the noise and disturbances we worry about cannot be mostly removed, which will affect the determination of the object's position.

In fact, the filter function selected ultimately retains most of the frequency information about the object and a small amount of irrelevant information. The spatial information obtained by the inverse Fourier transform therefore contains the position information of the object and some unrelated signal intensity. If we assert the position of the object has the maximum intensity of the spatial information, we end up getting not perfectly accurate result. For a more detailed discussion, we need to calculate the confidence interval for the position of the object computed in this way.

References

[1] *Fourier Methods: Why So Prominent?* URL: https://allsignalprocessing.com/2019/10/22/fourier-methods-prominent/m.

[2] Jose Nathan Kutz. *Data-driven modeling & scientific computation: methods for complex systems & big data*. Oxford University Press, 2013.

[3] *Matplotlib - mplot3d*. URL: https://matplotlib.org/mpl_toolkits/mplot3d/tutorial.html.

[4] *MLab reference - Plotting functions*. URL: https://docs.enthought.com/mayavi/mayavi/auto/mlab_helper_functions.html.

[5] *NumPy Reference*. URL: https://docs.scipy.org/doc/numpy/reference.

[6] *Wikipedia - Fast Fourier transform*. URL: https://en.wikipedia.org/wiki/Fast_Fourier_transform.

[7] *Wikipedia - Fourier transform*. URL: https://en.wikipedia.org/wiki/Fourier_transform.

Appendix A Python Functions used

Below are the Python functions implemented in the code with a brief explanation.

- `numpy.linspace(start, stop, num=50)` returns evenly spaced numbers over a specified interval [5].
- `numpy.meshgrid(*xi, **kwargs)` return coordinate matrices from coordinate vectors [5].
- `numpy.reshape(a, newshape, order='C')` gives a new shape to an array without changing its data [5].
- `numpy.fft.fftn(a)` compute the N-dimensional discrete Fourier Transform of `a` [5].
- `numpy.fft.ifftn(a)` compute the N-dimensional inverse discrete Fourier Transform of `a` [5].
- `numpy.fft.fftshift(x, axes=None)` shift the zero-frequency component to the center of the spectrum of `a` [5].
- `mlab.contour3d(x, y, z, scalars, ...)` plots iso-surfaces for a 3D volume of data [4].
- `axes.plot3d(xs, ys, ...)` plots 3D data [3].

Appendix B Python Codes

```
#!/usr/bin/env python
# coding: utf-8
# author: Yiyun Dong

# In[1]:


# import packages
from scipy.io import loadmat
import numpy as np
from mayavi import mlab
from mpl_toolkits import mplot3d
import matplotlib.pyplot as plt
get_ipython().run_line_magic('matplotlib', 'inline')


# In[2]:


# load data from file
Undata = loadmat('Testdata.mat')['Undata']


# In[3]:


# Spatial domain
L = 15
# Fourier modes
n = 64
# x, y, z: grid points in 3d position space; x2: all grid points;
x2 = np.linspace(-L, L, n+1)
```

```
x = x2[0:n]
y = x
z = x
# k: grid points in frequency space; ks: fftshift to zero as center
k = (2*np.pi/(2*L))*np.concatenate((np.arange(0, n/2), np.arange(-n/2, 0)))
ks = np.fft.fftshift(k)
# meshgrid coordinate
[X, Y, Z] = np.meshgrid(x, y, z)
[Kx, Ky, Kz] = np.meshgrid(ks, ks, ks)


# In[4]:


def isosurface(X, Y, Z, V, show = False, axesrange = None):
    # an substitution to isosurface in Matlab

    def t(a):
        """Transpose to coerce np.meshgrid output match np.mgrid output. a must
        be a 3d-array."""
        return a.transpose([1, 0, 2])

    if show:
        mlab.clf()
        mlab.figure(bgcolor=(1, 1, 1))
        fig = mlab.contour3d(t(X), t(Y), t(Z), t(V), opacity = 0.8, transparent = True)
        if axesrange == None:
            axes = mlab.axes(color = (0, 0, 0), nb_labels = 5)
        else:
            axes = mlab.axes(color = (0, 0, 0), nb_labels = 5, extent = axesrange)
        axes.title_text_property.color = (0.0, 0.0, 0.0)
        axes.label_text_property.color = (0.0, 0.0, 0.0)
        mlab.show()
    else:
        fig = mlab.contour3d(t(X), t(Y), t(Z), t(V), opacity = 0.9, transparent = True)

    return fig


# In[5]:


# plot raw data
for item in Undata:
    Un = np.reshape(item, (n, n, n), order = 'F')
    fig = isosurface(X, Y, Z, abs(Un), show = True, axesrange = [-20, 20, -20, 20, -20, 20])


# In[6]:


# de-noise: average of Fourier Transform
Utmean = np.zeros((n,n,n))
for item in Undata:
```

```
    Un = np.reshape(item, (n, n, n), order = 'F')
    Ut = np.fft.fftn(Un)
    Utmean = Utmean + Ut
Utmean = np.fft.fftshift(Utmean)/20
fig2 = isosurface(Kx, Ky, Kz, abs(Utmean)/(abs(Utmean).max()), show = True)


# In[7]:


# The center frequency (cf) is at
idx = np.unravel_index(abs(Utmean).argmax(), abs(Utmean).shape)
[cfx, cfy, cfz] = [Kx[idx], Ky[idx], Kz[idx]]
[cfx, cfy, cfz]


# In[8]:


# filter
filterfunc = np.exp( -0.2 * ( (Kx - cfx)**2 + (Ky - cfy)**2 + (Kz - cfz)**2 ) )


# In[9]:


# trajectory 3 (coordinates) * 20 (trajectory)
trajx = np.zeros(len(Undata))
trajy = np.zeros(len(Undata))
trajz = np.zeros(len(Undata))

for j in range(len(Undata)):

    # filter data to expose the marble
    Un = np.reshape(Undata[j], (n, n, n), order = 'F')
    Ut = np.fft.fftn(Un) # Fourier Transform
    Utn = np.fft.fftshift(Ut) # Shift center to zero
    Uftn = filterfunc * Utn # Apply filter
    Ufn = np.fft.ifftn(Uftn) # Inverse Fourier Transform

    # find the coordinates of the marble (marble is supposed to exert the maximum intensity)
    idx = np.unravel_index(abs(Ufn).argmax(), abs(Ufn).shape)
    [trajx[j], trajy[j], trajz[j]] = [X[idx], Y[idx], Z[idx]]


# In[11]:


fig3 = plt.figure(figsize=(6,5))
ax = plt.axes(projection='3d')
ax.plot3D(trajx, trajy, trajz, 'purple')
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_zlabel('z')
```

```
plt.savefig('trajectory.png')


# In[12]:


# In the 20th measurements, the position of the marble is at
[trajx[-1], trajy[-1], trajz[-1]]
```