AMATH 582 Homework 1

Tomas Perez

January 27, 2020

Abstract

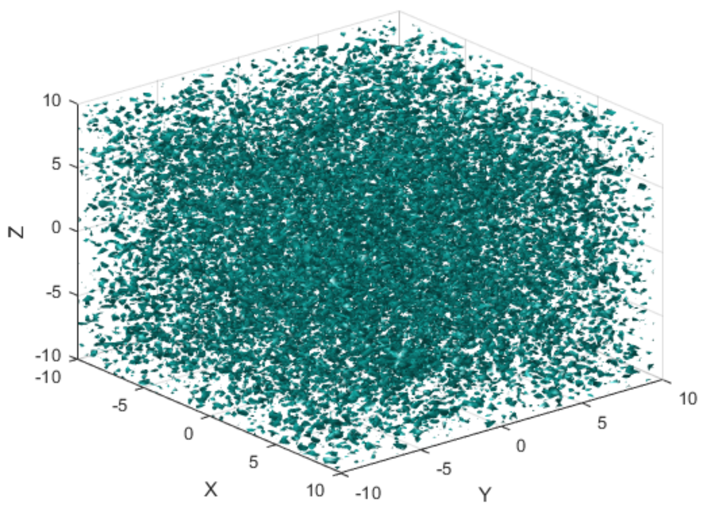
In this paper, Fast Fourier Transform (FFT) is implemented in an effort to locate a submarine within the Puget Sound area. The goal of this paper is to detect the submarine's acoustic signals and determine its trajectory. The data obtained is quite noisy, so averaging and filtering techniques used to help clean up the data. Averaging to obtain the central frequency and constructing a Gaussian filter around the central frequency gives us a clear plot of the submarines trajectory and allows us to see it's current location.

1 Introduction and Overview

There is a submarine traversing the Puget Sound area. It is believed that the submarine has new technology that emits an unknown acoustic frequency. Data is obtained over a 24-hour span in half-hour increments. The submarine is moving around the Puget Sound, making the data highly noisy. In order to find the trajectory of the submarine, the data needs to be cleaned up.

The data set is a broad spectrum recording of acoustic signals represented as 49 columns of data gathered over a 24-hour period in half-hour increments. Each 3-Dimensional plot of data contains the acoustic signal from the submarine and a fair amount of noise at that increment of time.

Figure 1: Noisy Spatial Data



As it stands, the data is too noisy to provide any useful information on the location of the submarine. To reduce the noise in the data, a Fast Fourier Transform (FFT) is used to transform the data from the time domain to the frequency domain. The acoustic frequency emitted by the submarine, called the central frequency, can be obtained by averaging the spectrum of frequencies. This requires the assumption that the noise in the data is white-noise. Meaning that the noise can be modeled by a normally-distributed random variable with a mean of zero and unit variance. When averaging the signal over all the data, we should see that the noise will zero out. Once the central frequency is obtained via averaging, we can filter over the central frequency to further remove unwanted noise. In this study, a Gaussian filter is used. The de-noised spectrum is then transformed back into the time domain using inverse FFT , giving us the necessary information to find the path of the submarine over time and it's current location.

2 Theoretical Background

In this paper, we use Fourier transforms to find the submarine in the Puget Sound. Fourier analysis tells us that any function $f(x)$ in the interval $x \in (-L, L]$ can be represented by a trigonometric series of sines and cosines. This greatly simplifies the evaluation of certain equations, including our evaluation of the acoustic signals from the submarine.

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

or in the complex representation:

$$f(x) = \sum_{n=-\infty}^{\infty} c_n e^{\frac{in\pi}{L}x} \quad x \in (-L, L] \tag{2}$$

With the orthogonality of trigonometric functions $\cos(x)$ and $\sin(x)$, we can find the coefficients, a_n and b_n, with the following:

$$a_n = \frac{1}{L}\int_{-L}^{L} f(x) \cos\frac{n\pi}{L}x \, dx \tag{3}$$

$$b_n = \frac{1}{L}\int_{-L}^{L} f(x) \sin\frac{n\pi}{L}x \, dx \tag{4}$$

or in the complex representation:

$$c_n = \frac{1}{2L}\int_{-L}^{L} f(x) e^{-\frac{in\pi}{L}x} \, dx \tag{5}$$

The expansions of Equation (1) gives us a periodic function that spans the interval $x \in (-L, L]$. The Fourier transform of this function over the interval $x \in [-\infty, \infty]$ can be defined as:

$$F(k) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty} f(x) e^{-ikx} \, dx \tag{6}$$

with inverse,

$$f(x) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty} F(k) e^{ikx} \, dx \tag{7}$$

2.1 Fast Fourier Transform

For the analysis in this study, we specifically use an algorithm called Fast Fourier Transform. This method has several key benefits. First, it rapidly computes the discrete Fourier transform (DFT). Normal computation of the DFT takes $O(N^2)$ operations, but FFTs of size 2^n are significantly more efficient, dropping necessary operations to $O(N \log N)$. Given the integral kernel e^{ikx}, the solutions on the interval $x \in (-L, L]$ will have periodic boundary conditions. The DFT is defined by the formula,

$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{ikn2\pi}{N}} \quad k = 0, ..., N-1 \tag{8}$$

2.2 Noise Reduction and the Gaussian Filter

The largest hurdle in our problem, is the noise that surrounds the target signal in our data. A major assumption made in this study is that all frequencies other than our central frequency is considered white noise. Thus, our goal is to remove that excess noise. Noise is often represented as a normally-distributed random variable with a mean of zero and unit variance. If this is the case, then averaging the 49 available measurements should in the Fourier domain should zero out any white noise, leaving behind the frequency signature k_c generated by the submarine. To further reduce excess noise and smooth the data, we can filter over the captured central frequency of the submarine. One of the simplest and most common filters is the Gaussian filter:

$$G_f(k) = e^{-\tau(k-k_0)^2} \tag{9}$$

This can be further expanded to 3-dimensional case of the Gaussian filter used in this study:

$$G_f(k) = e^{-\tau((K_x-f_x)^2)+(K_y-f_y)^2)+(K_z-f_z)^2)} \tag{10}$$

3 Algorithm Implementation and Development

The algorithm used to find the submarines path and current location is as follows:

1. Load the data and construct the spatial domain.
2. Compute the wave numbers (k) and rescale them by $\frac{\pi}{L}$.
3. Create spatial and spectral meshgrids for the time and frequency domains.
4. Reshape the data into a 64x64x64 array representing a 3-dimensional image of the Puget Sound area.
5. Average the transformed signals and take the maximum frequency component as the center frequency.
6. Use the central frequency to construct the Gaussian filter.
7. Multiply the shifted Fourier transformed signals with the filter.

8. Inverse shift and transform the data back into the spatial domain.
9. Take the maximum to find the locations of the submarine over time.
10. Plot the path and determine its current position.

Algorithm 1: Submarine Hunting Algorithm

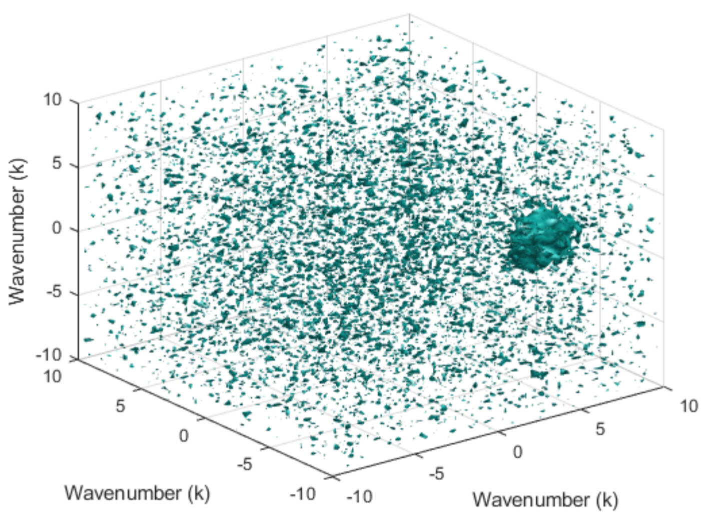
Import data from `subdata.mat`
for $j = 1 : 49$ **do**
  Reshape the data into a 3D matrix and extract measurement j from `subdata`
  Sum the FFTs of j
end for
Take the max, and find the central frequency k_c
Construct the Gaussian filter $G_f(k)$
for $j = 1 : 49$ **do**
  Reshape the data into a 3D matrix and extract measurement j from `subdata`
  Take the FFT of j and filter
  Inverse shift and IFFT
  Find max values and append to the path
end for
Plot the path of the submarine and final position

---

4 Computational Results

Figure 2 shows the averaged and normalized spectral data. The large mass represents the center frequency $k_c = [fx, fy, fz]$.

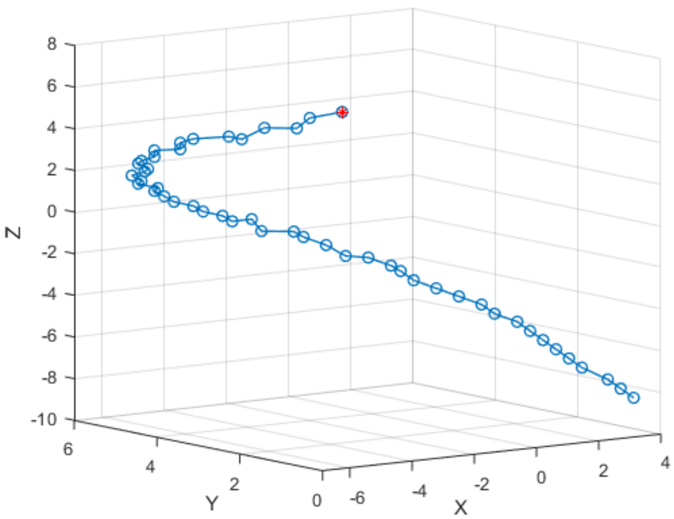
Figure 2: Averaged Spectral Data



$$k_c = [5.3407, -6.9115, 2.1991] \tag{11}$$

Figure 3 shows the path of the submarine in the Puget Sound area. The submarine is moving in an upward spiral, with it's current position being located at coordinates [-5, 0.9375, 6.5625].

Figure 3: Submarine Path



5 Summary and Conclusions

This study shows the advantages Fourier analysis in signal processing and how the use of different techniques resolve issues with highly noisy data. Averaging the spectral data allowed us to find the frequency signature k_c emitted by the submarine. The Gaussian filter $G_f(k)$ was constructed around this center frequency to further filter out the excess noise and smooth the data. After applying the filter to the Fourier transformed signals, and taking the inverse transform, we have clear positions of the submarine taken from the max values of the de-noised data.

Appendix A MATLAB Functions

- `y = linspace(x1,x2,n)` returns a row vector of `n` evenly spaced points between `x1` and `x2`.
- `[X,Y] = meshgrid(x,y)` returns 2-D grid coordinates based on the coordinates contained in the vectors `x` and `y`. X is a matrix where each row is a copy of `x`, and `Y` is a matrix where each column is a copy of `y`. The grid represented by the coordinates `X` and `Y` has `length(y)` rows and `length(x)` columns.
- `ks = fftshift(k)` rearranges a Fourier transform `X` by shifting the zero-frequency component to the center of the array.
- `U = fftn(X)` returns the multidimensional Fourier transform of an N-D array using FFT algorithm.
- `U = ifftn(X)` returns multidimensional discrete inverse Fourier transform of an N-D array using FFT algorithm.
- `U = reshape(X,V)` reshapes `X` using size vector `V`.
- `[row,col] = ind2sub(V,I)` returns the arrays row and col containing the equivalent row and column subscripts corresponding to the linear indices `I` for a matrix of size `V`
- `fv = isosurface(X,Y,Z,V,val)` computes isosurface data from the volume data `V` at the isosurface value specified in `val`.
- `[M,I] = max(A)` returns the maximum element of an array and the index into the operating dimension that corresponds to the maximum value of `A`

Appendix B MATLAB Code

```
close all; clear all; clc
load subdata.mat
L=10; % spatial domain
n=64; % Fourier modes
x2=linspace(-L,L,n+1); xp=x2(1:n);
k=(2*pi/(2*L))*[0:(n/2-1) -n/2:-1]; ks=fftshift(k);
[X,Y,Z]=meshgrid(xp,xp,xp);
[Kx,Ky,Kz]=meshgrid(ks,ks,ks);
%graph of unfiltered unaveraged data
Un(:,:,:)=reshape(subdata(:,1),n,n,n);
close all, isosurface(X,Y,Z,abs(Un),0.4)
axis([-10 10 -10 10 -10 10]), grid on, drawnow
%title('Noisy Data')
xlabel('X'), ylabel('Y'), zlabel('Z')
%% average the signal
Uavg=zeros(n,n,n);
for j=1:49
    Un(:,:,:)=reshape(subdata(:,j),n,n,n);
    Uavg = Uavg + fftn(Un);
end
Uavg=abs(fftshift(Uavg))./49;
[M,I]=max(Uavg(:));
%% graph of averaged unfiltered spectral data
figure(2)
isosurface(Kx,Ky,Kz,abs(Uavg)./abs(M), 0.2)
axis([-10 10 -10 10 -10 10]), grid on, drawnow
%title('Isosurface of Averaged Spectral Data')
xlabel('Wavenumber (k)'),ylabel('Wavenumber (k)'),zlabel('Wavenumber (k)')
%% central freq and filter
[I1,I2,I3]=ind2sub(size(Uavg),I);
fx=Kx(I1,I2,I3); fy=Ky(I1,I2,I3); fz=Kz(I1,I2,I3);
filter=exp(-((Kx-fx).^2 + (Ky-fy).^2 + (Kz-fz).^2)); % gauss filter
%% positions of the sub
xpos = zeros(1,49); ypos = zeros(1,49); zpos = zeros(1,49);
for j=1:49
    Un2(:,:,:)=fftn(reshape(subdata(:,j),n,n,n));
    Unft=filter.*fftshift(Un2);
    Unfs=ifftshift(Unft);
    Unf=ifftn(Unfs);
    Unf=ifftn(Unft);
    [M,I]=max(abs(Unf(:)));
    [I1,I2,I3]=ind2sub(size(Unf),I);
    xpos(j)=X(I1,I2,I3);ypos(j)=Y(I1,I2,I3);zpos(j)=Z(I1,I2,I3);
end
end_position = [xpos(49), ypos(49), zpos(49)] % ending sub position
%% plot the path of the sub
plot3(xpos, ypos, zpos,'-o','Linewidth',[1]), grid on;
hold on
plot3(xpos(49),ypos(49),zpos(49),'r*');
%title('Sub Path')
xlabel('X'), ylabel('Y'), zlabel('Z')
```

Listing 1: Submarine Hunting Code