AMATH 482 Homework 1

Seoyoung Cho

January 27, 2021

Abstract

In this paper, we explore signal processing methods to complete the task of tracking a submarine with a given signal data from underwater. To track the submarine, we effectively incorporate signal analysis techniques such as the Fourier transformation, averaging, and Gaussian filtering. These techniques allow us to reduce unnecessary noise and enhance the submarine tracking process through appropriate data conversion between real and frequency spaces. Throughout the paper, the mathematical background and implementation of such techniques will be addressed to better understand the analysis process.

1 Introduction and Overview

With a new submarine technology, the submarine emits an unknown acoustic signature every 30 minutes over a 24-hour period in Puget Sound. We are given a signal data in a 3D space domain taken for every 49 realizations. Unfortunately, with the given data, the submarine signals are barely detectable by underwater noise. Figure 1 shows how the data are hardly interpretable. The task given is to apply the signal analysis techniques learned in class to track the submarine and send P-8 Poseidon submarine tracking aircraft to find the submarine after detecting the signal emitted from it. The overall process is divided into two subsections.

1. Averaging
   Since the signal data given are in real space, we take the Fourier Transformation of data to frequency space. Then we average over 49 realizations (time) of the transformed data to find the center frequency of the submarine signal.

2. Filtering and tracking the path
   After obtaining the center frequency, we apply Gaussian filter around to remove undesired noise. After detecting the signals from the submarine, we convert the signal data in frequency space back to real space by the inverse Fourier Transformation and trace the path of the submarine. Then we can find the final position of the submarine where P-8 Poseidon should be sent at.

This paper will explain in detail the theoretical background and the implementation of the mathematical concepts such as the Fourier Transformation, Gaussian filter, and Time averaging that are applied in analyzing the signal emitted from the submarine.

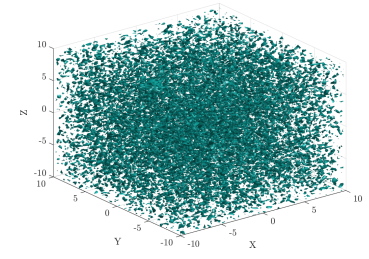


Figure 1: Noisy signal data in 3-Dimensional real space

2 Theoretical Background

This section explains the background of important mathematical concepts that arise in the process of performing the submarine signal analysis.

2.1 The Fourier Transform

The mathematical concept that builds the core of the process is the Fourier Transform. The Fourier Transform is an effective technique that decomposes signal functions in the real domain into the frequency (k) domain. This process is achieved by first breaking down the signal function into a sum of sines and cosines applying the Euler's formula:

$$e^{i\theta} = cos(\theta) + isin(\theta) \tag{1}$$

Then given a signal function f(x), $x \in \mathbb{R}$, the Fourier Transform of $f(x)$, $\hat{f}(k)$:

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

Further, the Inverse Fourier Transform recovers the signal function from the frequency function:

$$f(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \hat{f}(k) e^{ikx}\, dk \tag{3}$$

The Fourier Transform is an exceptional technique that converts the signal between real and frequency space. However, since we have a data over a finite time domain, we cannot apply the Fourier Transform. Hence, we introduce the alternative of the Fourier Transform, the Discrete Fourier Transform.

2.1.1 Discrete Fourier Transform (DFT)

As the title implies, the Discrete Fourier Transform is applied to the finite and discrete data points. Given a sequence of N equally spaced signal data, the DFT returns the sequence of corresponding numbers in frequency space:

$$\hat{x}_k = \frac{1}{N} \sum_{n=0}^{N-1} [x_n e^{2\pi i k n / L}], \quad k = \{0, 1, 2, \cdots, N-1\} \tag{4}$$

It is important to scale x with $\frac{2\pi}{L}$ because the data expressed as the DFT are required to be 2π periodic. Further, the order of k is arbitrary by misidentification with different signals based on the fact that k and $k + N$ are the same in the DFT. This fact allows performing the DFT in MATLAB to give the Fourier values correspond to the frequency ordered as $k = \{0, 1, \cdots, \frac{N}{2}, \frac{-N}{2} + 1, \frac{-N}{2} + 2, \cdots, -1\}$. Hence, we need to rearrange the transformed data placing zero-frequency data in the center. One drawback of the DFT is its slow running complexity of $\mathcal{O}(N^2)$. To increase efficiency, we use the DFT with Fast Fourier Algorithm, an algorithm that runs the DFT with the complexity of only $\mathcal{O}(N \log N)$.

2.2 Gaussian Filter

Another important concept in this process is filtering. After applying the Fourier transformation to obtain frequency data, we use filters to clean up unwanted frequencies and better detect the signals in real space. Gaussian filter is only one of the various options of filters, but we use it considering its simplicity. Given the central frequency (k_0) and the width of the filter (τ), the Gaussian filter $F(k)$ is:

$$F(k) = e^{-\tau(k-k_0)^2} \tag{5}$$

For our task, the central frequency specifies the presence of the target submarine signals. We apply Gaussian filters by simply multiplying the above equation by the signal data in the frequency domain. The filter then focuses only on signals inside the window of width (τ) around the center frequency (k_0) and reduces noise outside the frame.

2.3 Averaging

Similar to Gaussian filters, time averaging is another technique used to improve the signal detection process. Using the fact that the mean of white noise is 0, the idea of averaging is to cancel white noise in the frequency space by adding them over many realizations. Therefore, the more realizations we average the signal, the closer we will get to the true signal. It is important to average the signal in the frequency space. In real space, signals emitted by submarines will shift in location as the submarine moves. This requires detecting both signals and their movements, making the process harder. On the other hand, the shifting of the signals is not a concern when averaging in the frequency space because the signals from the submarine have the same frequency. One issue that we must be aware with averaging process in frequency space is that all the information about the time when the signals occur will be lost during the process.

3 Algorithm Implementation and Development

This section explains the implementation of the Fourier Transform, Filtering, Averaging, and Plotting trajectory algorithms using the MATLAB program. Refer to Appendix A for more explanation on actual implementation using MATLAB functions. Before implementing any algorithms, we need the starting setup:

- Load the `subdata.mat`. This data is a 2D array that compresses 64x64x64 signal data in real space over 49 time realizations.
- Define the size of the spatial domain, Fourier modes, and both real and frequency space grid vectors.
- Create both 3D spatial and frequency grid.

Now we reduce white noise in given data and find the submarine signal with averaging. Algorithm 1 is the pseudo-code of the averaging implementation:

- For each 49 realizations, convert each column of 2D `subdata` to 3D data. After running through all the realizations, we will end up with a 4D matrix where the first three dimensions represent the space and the fourth represents time.
- Since the averaging takes place in frequency space, apply 3-Dimensional Discrete Fourier Transform to the 3D data. This will take all the data into frequency space.
- Add all the transformed data in each realization. This process reduces the white noise, giving clearer signal data. Divide the sum by the number of realizations for the averaging purpose.
- Find the maximum and its index from the averaged signal. This maximum point specifies where the center frequency, a point indicator of the presence of the submarine signals, is at.

Algorithm 1: Time Averaginge Algorithm

```
for each realization do
    Apply the 3D DFT to the 3D signal data
    Add all the transformed data
end for
Divide the result data by realization to average out
Find maximum and its index from the absolute of the averaged data
Extract center frequency from the data at the maximum index
```

Once we have the center frequency, we can proceed with filtering to detect the submarine. Refer to Algorithm 2 for the pseudo-code of the filtering and submarine tracking implementation:

- Define Gaussian filter in 3D space that is centered at the center frequency with some window width value. The smaller the width, the more localized a filter is.

- Again, filtering is applied to the signal data in frequency space. Thus, multiply the Gaussian filter to each Fourier transformed data. Then the noise outside of the window range is filtered out and we can better detect the submarine signal for every realization.

Now we proceed with tracing the path of the submarine over a 24-hour period.

- To find the path, we need (x,y,z) position coordinates in real space. Taking the inverse Fourier Transformation to the filtered signal will return the signal in real space.
- Just like how we found the maximum and its index to get the center frequency, we find the same values from real space data. These values specify the location of the submarine. Hence, we can obtain the 3D position coordinates of the submarine location at each realization. We can then visualize the path of the submarine over the time domain.

Algorithm 2: Gaussian Filtering Algorithm

for each realization **do**
  Apply the 3D DFT to the 3D signal data
  Multiply the Gaussian filter to the transformed data
  Take inverse FT of the filtered data
  Find maximum and its index from the inverse transformed data
  Extract (x, y, z) coordinates from the data at the maximum index
end for

4 Computational Results

This section discusses the computational results derived after performing signal analysis in MATLAB.

4.1 Averaging

Figure 2 shows the signal data in frequency space after performing time averaging. The center frequency specifies the presence of the submarine signals and is where we should center our filter around to refine undesired noise. The center frequency is given by the maximum value of the absolute averaged signal.

$$(Kxi, Kyi, Kzi) = (5.3407, -6.9115, 2.1991)$$

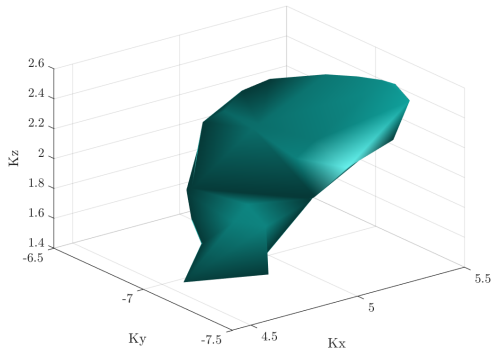


Figure 2: Averaged signal data in 3-dimensional frequency space

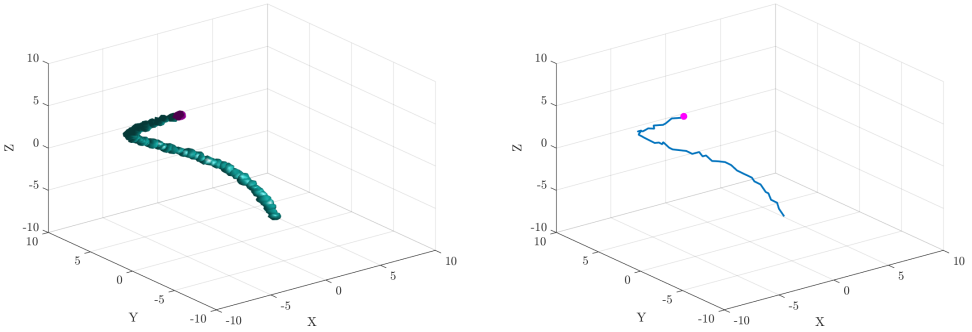
4.2 Filtering and Finding path

Figure 3(a) and (b) both visualize the submarine's path within 24 hours. Using the center frequency found after time averaging, Gaussian filter $e^{-0.02(((Kx-5.3407)^2)+((Ky+6.9115)^2)+((Kz-2.1991)^2))}$ was generated. The $\tau = 0.02$ seemed to be an optimal width value for localizing the filter. Figure 3(a) uses the `isosurface` command to represent the 3D surface of the location, and figure 3(b) uses the `plot3` command to display the trajectory line. Both figures show the final location of the submarine in magenta. For more information, Table 1 lists all (x,y) coordinates of the submarine location at each time. The final position of the submarine is highlighted in gray. We can infer that the final location is where the P-8 Poseidon submarine tracker should be sent. The following is the final coordinate of the submarine:

$$(x_{49}, y_{49}) = (-5, 0.9375)$$

5 Summary and Conclusions

We performed signal processing using the Fourier transform, time-averaging, and Gaussian filtering, which allowed us to successfully track a submarine in the Puget sound area despite the noise in the given data. Time averaging and Gaussian filters effectively attenuated noise to better detect the signals from the submarine. Furthermore, the appropriate use of the Fourier and the inverse Fourier transformation allowed conversion of data between real and frequency spaces to improve the signal analysis process. Therefore, the effectiveness of the technique was reiterated throughout this project.



(a) `isosurface` visualization (b) `plot3` visualization

Figure 3: Trajectory of the submarine in 24-hour period

Time	x	y	Time	x	y	Time	x	y
1	3.125	0	18	0.625	5.3125	35	-5.9375	5
2	3.125	0.3125	19	0.3125	5.625	36	-6.25	5
3	3.125	0.625	20	0	5.625	37	-6.5625	4.6875
4	3.4375	1.25	21	-0.3125	5.625	38	-6.5625	4.375
5	3.125	1.5625	22	-0.9375	5.625	39	-6.875	4.375
6	3.125	1.875	23	-1.25	5.9375	40	-6.875	3.75
7	3.125	2.1875	24	-1.875	5.9375	41	-6.875	3.4375
8	3.125	2.5	25	-2.5	5.9375	42	-6.875	3.125
9	3.125	2.8125	26	-2.8125	5.9375	43	-6.5625	2.8125
10	2.8125	3.125	27	-3.125	6.25	44	-6.875	2.5
11	2.8125	3.4375	28	-3.4375	5.9375	45	-6.25	2.1875
12	2.5	4.0625	29	-3.75	5.9375	46	-6.25	1.875
13	2.1875	4.0625	30	-4.375	5.625	47	-5.9375	1.5625
14	2.1875	4.375	31	-4.6875	5.625	48	-5.3125	1.25
15	1.875	4.6875	32	-5.3125	5.625	49	-5.3125	0.9375
16	1.5625	5	33	-5.3125	5.3125			
17	0.9375	5	34	-5.625	5.3125			

Table 1: x,y coordinates of the submarine's position at each time

Appendix A MATLAB Functions

This section lists all the important MATLAB Functions used in the process of submarine tracking.

- `linspace(x1,x2,n)`: returns a row vector of evenly spaced `n` points from `x1` to `x2`. This is used when creating real space grid vectors
- `meshgrid(x,y,z)`: returns 3-D grid coordinates based on the elements (coordinates) in vectors `x`,`y`, and `z`.
- `zeros(n)`: returns an empty vector of `n` points.
- `reshape(A,sz1,sz2,sz3)`: reshapes `A` into `sz1-by-sz2-by-sz3` array. Hence, this function lets to increase or decrease the dimension.
- `abs(X)`: returns the absolute value of each element in array `X`.
- `max(X)`: returns the maximum value in array `X`.
- `exp(x)`: returns the value of e^x.
- `[i1, i2, i3] = ind2sub(sz,ind)`: converts linear indices to corresponding multi-dimension subscripts of matrix of size `sz`. $[i1, i2, i3, \cdots, in]$ is an array of subscripts of n-dimensional matrix. This function is used to extract the center frequency and the location of the submarine.
- `fftn(A)`: returns the Discrete Fourier Transformation of n-dimensional data `A`. This uses the Fast Fourier Transform algorithm. Recall that performing the DFT in MATLAB requires 2π periodic signals and gives the Fourier values in the order of $\{\hat{x}_0, \hat{x}_1, \cdots, \hat{x}_{\frac{n}{2}}, \hat{x}_{\frac{-n}{2}+1}, \hat{x}_{\frac{-n}{2}+2}, \cdots, \hat{x}_{-1}\}$.
- `fftshift(A)`: returns the rearranged Fast Fourier transform with zero-frequency component shifted to the center.
- `ifftn(At)`: returns the inverse Discrete Fourier transformation of `n`-Dimensional data `At`. This uses the Fast Fourier Transform algorithm and returns the signal in real space.
- `isosurface(X,Y,Z,V,isovalue)`: Visualizes the isosurface from the data V at isovalue of `isovalue`. `X,Y,Z` represents the coordinate vectors from the domain. `V` represents the values corresponding to each coordinates.
- `plot3(X,Y,Z)`: Plots each coordinate given by `X,Y,Z` in 3D space.

6 MATLAB Code

```
clear all; close all; clc;

load subdata.mat % Imports the data as the 262144x49 (space by time) matrix called subdata

L = 10; % spatial domain
n = 64; % Fourier modes
x2 = linspace(-L,L,n+1); x = x2(1:n); y =x; z = x;
k = (2*pi/(2*L))*[0:(n/2 - 1) -n/2:-1]; ks = fftshift(k);
[X,Y,Z]=meshgrid(x,y,z); % real-space grid vectors
[Kx,Ky,Kz]=meshgrid(ks,ks,ks); % frequency-space grid vectors


% Averaging
ave = zeros(n,n,n);
for j=1:49
  Un(:,:,:)=reshape(subdata(:,j),n,n,n);
  M = max(abs(Un),[],'all');
  Unt = fftn(Un);
  ave = ave + Unt;
end

ave = abs(fftshift(ave))./49; % averaged signal
[max_ave, mave_ind] = max(ave(:));
[mave_x, mave_y, mave_z] = ind2sub(size(ave), mave_ind); % center frequency
                                                         % coordinates

% figure(1) % visualize signal in real-space
isosurface(X,Y,Z,abs(Un),0.4)
axis([-L L -L L -L L]), grid on, drawnow
xlabel('X'), ylabel('Y'), zlabel('Z')
%
figure(2) % visualize averaged signal in frequency-space
isosurface(Kx,Ky,Kz,abs(ave)/max_ave,0.7), grid on
xlabel('Kx'), ylabel('Ky'), zlabel('Kz')


% Gaussian filter
tau = 0.02; % Width of filter
Kxi = Kx(mave_x, mave_y, mave_z); %
Kyi = Ky(mave_x, mave_y, mave_z); % Center of the filter
Kzi = Kz(mave_x, mave_y, mave_z); %

% Define the filter
filter = exp(-tau*(((Kx - Kxi).^2) + ((Ky - Kyi).^2) + ((Kz - Kzi).^2)));

path_X = []; path_Y = []; path_Z = [];
for jj=1:49
    Un(:,:,:)=reshape(subdata(:,jj),n,n,n);
    Unt = fftn(Un);
    Unft = filter .*  fftshift(Unt); % applying filter
    Unf = ifftn(Unft); % real-space data
```

```
    [max_val, max_val_ind] = max(abs(Unf(:)));
    [max_x, max_y, max_z] = ind2sub(size(Unf), max_val_ind);
    % real-space coordinates of submarine position
    path_X(jj) = X(max_x, max_y, max_z);
    path_Y(jj) = Y(max_x, max_y, max_z);
    path_Z(jj) = Z(max_x, max_y, max_z);

    figure(3) % visualize trajectory (3D)
    isosurface(X, Y, Z, abs(Unf)/max_val, 0.8)
    axis([-L, L, -L, L, -L, L]), grid on, drawnow
    xlabel('X'), ylabel('Y'), zlabel('Z')
end
set(patch(isosurface(X, Y, Z, abs(Unf)/max_val, 0.8)), 'FaceColor', 'm', 'EdgeColor', 'none');
final_position = [path_X(49), path_Y(49)] % current position

figure(4) % visualize trajectory
plot3(path_X, path_Y, path_Z, 'Linewidth', 1.5)
axis([-L, L, -L, L, -L, L]), grid on, hold on
plot3(path_X(49), path_Y(49), path_Z(49), 'm.', 'Markersize', 20)
xlabel('X'), ylabel('Y'), zlabel('Z')
```