Submarine Detection Using Noisy Acoustic Data

Jiajun Bao

Abstract

This project solves a submarine radar detection problem, using noisy acoustic data obtained over a 24-hour period in half-hour increments to determine the moving submarine's location and path and identify the acoustic admissions of the submarine.

1 Introduction and Overview

This project aims to hunt for a submarine in the Puget Sound using noisy acoustic data. As we know, when the noise of the data we obtained is large, the signal is difficult to detect. Therefore, the goal is to try to filter out the noise from the noisy acoustic data to better detect the signal. In order to get closer to the true signal, the two main approaches are averaging over realizations to try to cancel the noise from each realization and using a spectral filter to try to remove undesired frequencies and noise. In this project, the frequency signature generated by the submarine will be determined first through averaging of the spectrum. Then I will filter the data around the center frequency to denoise the data and provide the path and location of the submarine to P-8 Poseidon subtracking aircraft.

2 Theoretical Background

2.1 Fourier Transform and Discrete Fourier Transform

Fourier Transform is widely used in applications like signal processing. In time series data, the function will be a function of time, and the Fourier transform will be a function of frequency, but the Fourier Transform assumes an infinite domain, so in this case, we will use Discrete Fourier Transform. In MATLAB, it has a built-in function that computes Discrete Fourier Transform called The Fast Fourier Transform (FFT), which is an algorithm with total complexity $\mathcal{O}(N \log N)$, which does the Discrete Fourier Transform ($\mathcal{O}(N^2)$) faster.

2.2 Averaging in Frequency Domain

The noise of this acoustic data is white noise, which is the noise that affects all frequencies the same. Averaging over many realizations will canceling our the noise from each realization and thus help us getting close to the true signal. The averaging process is done in the frequency domain instead of the time domain. This is because in the time domain, the incoming signal would remain the same shape, but the location of the spike will change. In the frequency domain, the same signal shifted in the time domain will look the same in frequency domain (if considering $|\hat{u}|$).

2.3 Noise Filtering

The filtering out method filter around the frequency and help removing undesired frequencies and white noise. In this case, the center frequency generated by the submarine can be found by averaging the spectrum. Mathematically, a filter is a function which we will multiply the signal in the frequency domain. We choose to use Gaussian function in the form of: $F(k) = e^{\tau(k-k_0)^2}$, with τ determining the width of the filter and constant k_0 determining the center of the filter.

3 Algorithm Implementation and Development

- Import data as the 262144 x 49 (space by time) matrix from **subdata.mat**
- Define length of computational spatial domain, then discretize the spatial domain.
- Rescale frequency as the function FFT makes the assumption that the domain is $[-\pi, \pi]$. Here the domain is [-L,L], the space needs to be rescaled back to $[-\pi, \pi]$ by multiplying $(2 * \pi/(2 * L))$. Then use *fftshift* to shift zero-frequency component to center of spectrum.
- Construct 3-D coordinates using *meshgrid*.
- Reshape the data into 3D using *reshape*, and transform the data from time space to frequency space by using *fftn*. Plot the raw signals in frequency space before averaging using *isosurface*.
- Average over 49 realizations in frequency spaces by adding each realization together, taking the absolute value of the sum, and then divide it by 49.
- Find the frequency signature (center frequency) using *max* function and convert linear indices to subscripts using *ind2sub*. Then utilize the subscripts indexes to find the exact location of peak frequency in K_x, K_y, and K_z.
- Normalized the signals by dividing the biggest value of the frequency obtained from above *max* function, then plot the averaged signals using *isosurface*.
- Define the filter around the center frequency and apply the filter to the signal in the frequency space.
- Inverse the filtered signal in frequency space back to time space using *ifftn*, then determine the path of the submarine using *max* function and convert linear indices to subscripts using *ind2sub*.
- Then utilize the subscripts indexes to find the coordinates of the location and plot the path of the submarine in time space using *plot3*.

4 Computational Results

Based on the above algorithms in section 3, (Please refer to Appendix B for detailed codes in MATLAB), the central frequency is $[kx, ky, kz] = [5.3407, -6.9115, 2.1991]$. The isosurface visualization (isovalue 0.4) of the raw signals in the frequency space before averaging is plotted in Figure 1. The isosurface visualization (isovalue 0.5) of the averaged signals in the frequency space is plotted in Figure 2.

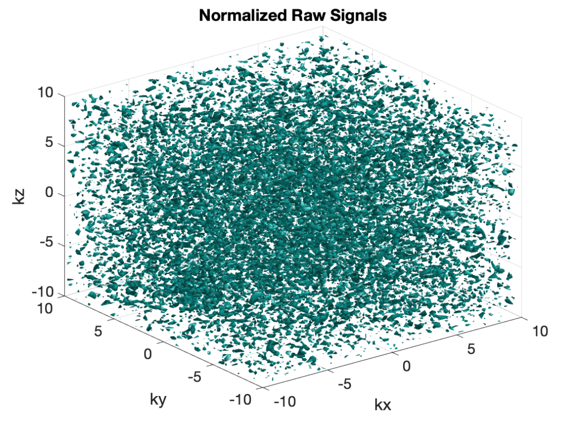


Figure 1: Normalized Raw Signals

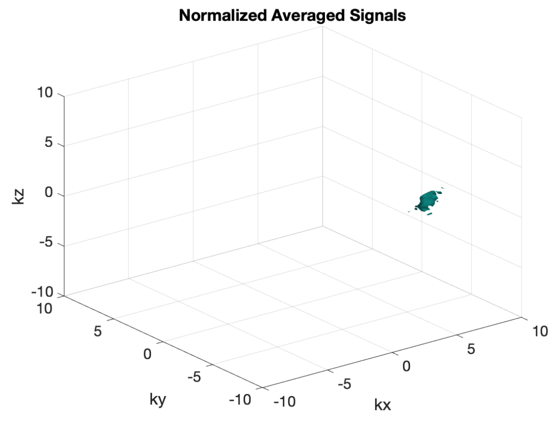
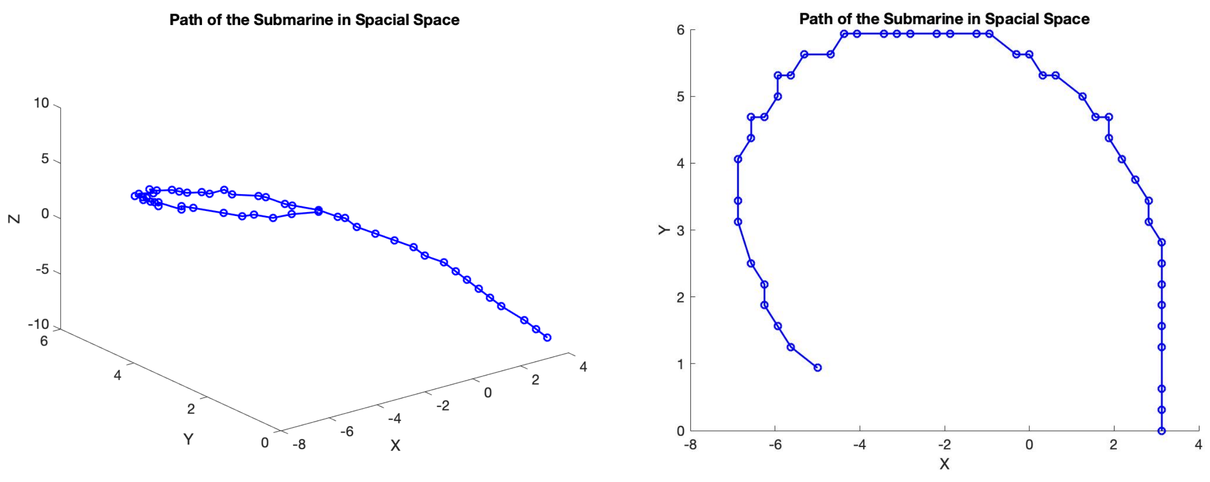


Figure 2: Normalized Raw Signals



(a) 3D View (b) 2D View

Figure 3: Path of the Submarine in Spacial Space

After filtering the data around the center frequency determined above, we are able to denoise the data and determine the path of the submarine. The location and path of the submarine is plotted in Figure 3. Figure 3(a) is the plot in 3D view. Figure 3(b) is the plot in 2D view.

As the P-8 Poseidon subtracking aircraft only requires the x and y coordinates, the following table 1 provides the x and y coordinates with time step in half-hour increments over a 24-hour period for the subtracking aircraft to follow the submarine.

5 Summary and Conclusions

In this problem, to determine the location and path of the moving submarine, we first average over realizations in frequency space and found the center frequency generated by the submarine, which is $[kx, ky, kz] = [5.3407, -6.9115, 2.1991]$. Once the frequency signature generated by the submarine is known, we can apply the spectral filter around the determined center frequency successfully helped us denoise the data and find the path of the submarine, which is illustrated in the figure 3. With the determined location and path of the submarine, we are able to provide the P-8 Poseidon subtracking aircraft with the x and y coordinates to follow the submarine, which is listed in the table 1.

Time Step	1	2	3	4	5	6	7	8	9	10	11	12
X-Coordinate	3.1250	3.1250	3.1250	3.1250	3.1250	3.1250	3.1250	3.1250	3.1250	2.8125	2.8125	2.5000
Y-Coordinate	0	0.3125	0.6250	1.2500	1.5625	1.8750	2.1875	2.5000	2.8125	3.1250	3.4375	3.7500

Time Step	13	14	15	16	17	18	19	20	21	22	23	24
X-Coordinate	2.1875	1.8750	1.8750	1.5625	1.2500	0.6250	0.3125	0	-0.3125	-0.9375	-1.2500	-1.8750
Y-Coordinate	4.0625	4.3750	4.6875	4.6875	5.000	5.3125	5.3125	5.6250	5.6250	5.9375	5.9375	5.9375

Time Step	25	26	27	28	29	30	31	32	33	34	35	36
X-Coordinate	-2.1875	-2.8125	-3.1250	-3.4375	-4.0625	-4.3750	-4.6875	-5.3125	-5.6250	-5.9375	-5.9375	-6.2500
Y-Coordinate	5.9375	5.9375	5.9375	5.9375	5.9375	5.9375	5.6250	5.6250	5.3125	5.3125	5.000	4.6875

Time Step	37	38	39	40	41	42	43	44	45	46	47	48	49
X-Coordinate	-6.5625	-6.5625	-6.8750	-6.8750	-6.8750	-6.8750	-6.8750	-6.5625	-6.2500	-6.2500	-5.9375	-5.6250	-5.000
Y-Coordinate	4.6875	4.3750	4.0625	4.0625	3.4375	3.4375	3.1250	2.5000	2.1875	1.8750	1.5625	1.2500	0.9375

Table 1: X and Y coordinates of the submarine

Appendix A MATLAB Functions

- `y = linspace(x1,x2,n)` generates n points. The spacing between the points is $(x2 - x1)/(n - 1)$..
- `[X,Y,Z] = meshgrid(x,y,z)` returns 3-D grid coordinates defined by the vectors x, y, and z. The grid represented by X, Y, and Z has size length(y)-by-length(x)-by-length(z).
- `Y = fftshift(X)` shiftt zero-frequency component to center of spectrum
- `Y = fftn(X)` returns the multidimensional Fourier transform of an N-D array using a fast Fourier transform algorithm. The N-D transform is equivalent to computing the 1-D transform along each dimension of X.
- `X = ifftn(Y)` returns the multidimensional discrete inverse Fourier transform of an N-D array using a fast Fourier transform algorithm. The N-D inverse transform is equivalent to computing the 1-D inverse transform along each dimension of Y.
- `B = reshape(A,sz1,...,szN)` reshapes A into a sz1-by-...-by-szN array where sz1,...,szN indicates the size of each dimension.
- `[I1,I2,...,In] = ind2sub(sz,ind)` returns n arrays I1,I2,...,In containing the equivalent multidimensional subscripts corresponding to the linear indices ind for a multidimensional array of size sz. Here sz is a vector with n elements that specifies the size of each array dimension.

Appendix B MATLAB Code

```
% Clean workspace
clear all; close all; clc

load subdata.mat % Imports the data as the 262144x49 (space by time) matrix called subdata 5
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);
[Kx,Ky,Kz]=meshgrid(ks,ks,ks);

for j=1:49
    Un(:,:,:)=reshape(subdata(:,j),n,n,n);
    M = max(abs(Un),[],'all');
    close all, isosurface(X,Y,Z,abs(Un)/M,0.7)
    axis([-20 20 -20 20 -20 20]), grid on, drawnow
    pause(1)
end

% plot raw signals
for j=1:49
    un_raw(:,:,:)=reshape(subdata(:,j),n,n,n);
    unf_raw = fftn(un_raw);
end
Max = max(abs(unf_raw),[],'all');
figure
isosurface(Kx,Ky,Kz, abs(unf_raw./Max), 0.4)
axis([-10 10 -10 10 -10 10]), grid on, drawnow
xlabel('kx')
```

```
ylabel('ky')
zlabel('kz')
title ('Normalized Raw Signals')
set(gca,'FontSize',14)
print(gcf,'-dpng','fig1.png')

% average over realizations in frequency spaces
u_ave = zeros(n,n,n);
for j=1:49
    Un(:,:,:)=reshape(subdata(:,j),n,n,n);
    unf = fftn(Un);
    u_ave = u_ave + unf;
end
u_ave = abs(fftshift(u_ave))./49;

% find the frequency signature (center frequency)
[M,I] = max(u_ave(:));
[I1,I2,I3] = ind2sub(size(u_ave),I);

% plot averaged signals
figure
isosurface(Kx,Ky,Kz, abs(u_ave)./M, 0.5)
axis([-10 10 -10 10 -10 10]), grid on, drawnow
xlabel('kx')
ylabel('ky')
zlabel('kz')
title ('Normalized Averaged Signals')
set(gca,'FontSize',14)
print(gcf,'-dpng','fig2.png')

%location of peak frequency
peak_x = Kx(I1,I2,I3);
peak_y = Ky(I1,I2,I3);
peak_z = Kz(I1,I2,I3);


% Define the filter
tau = 0.5;
filter = exp(-tau*((Kx-peak_x).^2 + (Ky-peak_y).^2 + (Kz-peak_z).^2));

X_path = zeros(49,1);
Y_path = zeros(49,1);
Z_path = zeros(49,1);

for j = 1:49
    un = reshape(subdata(:,j),n,n,n); % Noisy signal in time space
    utn = fftn(un); % Noisy signal in frequency space
    utn = fftshift(utn);
    unft = filter .* utn; % Apply the filter to the signal in frequency space
    unf = ifftn(unft); % Inverse back to time space
    [M1,Ind] = max(unf(:));
    [X_x,Y_y,Z_z] = ind2sub(size(unf),Ind);
    X_path(j,1) = X(X_x, Y_y, Z_z);
    Y_path(j,1) = Y(X_x, Y_y, Z_z);
    Z_path(j,1) = Z(X_x, Y_y, Z_z);
end

% plot the path of the submarine in time space
figure
plot3(X_path, Y_path, Z_path,'-o','Color','b','LineWidth', 1.5)
title('Path of the Submarine in Spacial Space','FontSize', 30)
xlabel('X','FontSize', 12)
ylabel('Y','FontSize', 12)
zlabel('Z','FontSize', 12)
set(gca,'FontSize',12)
print(gcf,'-dpng','fig3.png')
```

Appendix C References

1. Wikimedia Foundation. (2021, June 19). Rössler attractor. Wikipedia. Retrieved December 15, 2021, from https://en.wikipedia.org/wiki/R%C3%B6ssler_attractor

2. Wikimedia Foundation. (2021, December 13). Lorenz System. Wikipedia. Retrieved December 15, 2021, from https://en.wikipedia.org/wiki/Lorenz_system

3. AMATH581 notes(Prof J. Nathan Kutz)