A Submarine Problem

Paul Abboud

***Abstract*—Noisy acoustic data of a submarine in Puget Sound was recorded over a 24 hour period. The location and trajectory of the submarine need to be determined, but the acoustic frequency emitted by the submarine is unknown. Through the use of signal averaging and filtering of frequency domain data, we will produce denoised data to identify the correct frequency. Reverting the remaining frequency data to the time domain will allow us to find the position of the submarine.**

I. Introduction

There is a submarine in the Puget Sound that needs to be located. Data of the submarine was obtained over a 24-hour period in half-hour increments. The new submarine technology emits an unknown acoustic frequency that needs to be detected. The data contains excess noise since the submarine is moving. This prevents us from identifying the correct acoustic frequency and determining the location and trajectory of the submarine.

In order to resolve the issue of cluttered data, we need to determine the central frequency and remove unnecessary noise around it. Assuming the movements of the submarine are random, we can find the center frequency by averaging the 49 transformed data sets. Applying a Gaussian Filter around the center frequency will denoise the data and then reversing the Fourier transformed data back into the time domain will help produce the path of the submarine. We will plot the path and direct the P-8 Poseidon subtracking aircraft to finally locate the submarine.

II. Theoretical Background

To understand the issue at hand, the following will discuss the key ideas behind extracting the target acoustic frequency. The concepts implemented are Fourier transforms, signal averaging and Gaussian Filtering.

A. Fourier Transforms

The first essential aspect of processing the provided signal data is applying the Fourier transform. The Fourier transform is denoted by $\hat{f}(k)$, taking in the function $f(x)$.

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

The Fourier transform in Eq.(1) inputs a function of space or time, x, and transforms it into a function of frequencies, k. To revert the transform from a frequency domain to a space or time domain, we can apply the inverse Fourier transform defined as $f(x)$.

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

After processing the data, to produce the position and trajectory of the submarine we will apply an inverse Fourier transform. The methods illustrated above are useful when the function is continuous, in this case we have a discrete set of data sampled at equally-spaced points. Instead, we will utilize the Discrete Fourier Transform (DFT) defined as a sequence of numbers, $\hat{x}_k$.

$$\hat{x}_k = \frac{1}{N} \sum_{n=0}^{N-1} x_n e^{\frac{2\pi i k n}{N}} \tag{3}$$

Where N is the number of data points and values of k are the frequencies present. In the DFT, we can mistake signals for other signals due $-k$ and $k+N$ being equivalent in the algorithm. This is known as aliasing and to avoid it we can instead use frequencies from

$$k = \frac{-N}{2}, \frac{-N}{2}+1, ..., 1, 0, 1, 2, ..., \frac{N}{2}-1 \tag{4}$$

We will implement the Fast Fourier Transform (FFT), a faster and less computationally expensive version of DFT, into Matlab. The total complexity of DFT is $\mathcal{O}(N^2)$ which is highly expensive when N is large. However, the FFT is $\mathcal{O}(n \log n)$ which is a huge improvement when dealing with large data sets like those in signal processing. The FFT algorithm assumes 2π periodic signals over a discrete interval. Our data is not on the interval $[-2\pi, 2\pi]$ so we will have to re-scale the frequencies by $\frac{2\pi}{L}$ where L is the length of computational domain. After denoising the signal, can then compute the Inverse Fast Fourier Transform (IFFT) to transform the data back to a space domain.

B. Signal Averaging

Assuming that the additional noise is random we can model it as a normally distributed random variable with zero mean and unit variance. Averaging over many frequency sets, the noise will sum to zero and this produces the central frequency. Signal averaging is a useful tool when combined with Gaussian Filtering around the central frequency. Utilizing signal averaging and the following concept will yield denoised and readable data.

C. Gaussian Filtering

After computing the central frequency we can filter noise around a desired frequency by applying a Gaussian Filter. The Gaussian is centered around the frequency k_0 and minimizes values around the central frequency by multiplying the filter and frequency data. Data apart from the target frequency are minimized by the filter. The filter we will implement is

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

where τ determines the width of the filter. Applying the Gaussian Filter described in Eq.(5) will be sufficient in this application, however, filter design and fine-tuning parameters play a significant role in optimizing filtration.

III. Algorithm Implementation

Firstly, we need to establish the problem space to process signals. We can establish the space domain by using

```
x2 = linspace(-L, L, n+1);
x = x2(1:n);
y = x; z = x;
```

Then we can establish the frequency domain first by multiplying $\frac{2\pi}{L}$ and the frequency set in Eq.(4) to re-scale k and then shifting k for FFT. Finally, we can create the grid space of both domains.

```
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);
```

Then we can reshape the data that was emitted from the submarine to match the domain of the grid space and finally FFT the data to yield noisy frequency data.

```
for j = 1:49
    Un(:,:,:) = reshape(subdata(:,j), n, n, n);
end
```

IV. Computational Results

Plotting the isosurface of the noisy data produced the following figure which describes the problem set after applying signal averaging but before Gaussian Filtering.

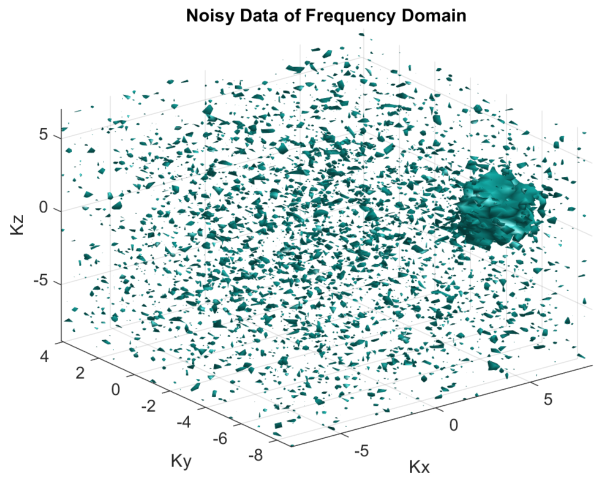

Fig. 1. Submarine Acoustic Data

V. Summary of Results

From Fig. 1 we can see that much of noise present in the data was reduced and a central frequency stands out allowing us to later apply Gaussian Filtration to further remove excess noise from the data.

Appendix A

MATLAB Functions

fftn: Using the Fast Fourier Transform algorithm, returns the Fourier transformed function.

fftshift: Rearranges the Fourier transformed data by shifting the zero frequency to the center.

isosurface: Produces an isosurface from volume data at a specified isosurface value.

meshgrid: Returns a 3-D grid based on 3 vector inputs.

linspace: Creates a discrete interval from a specified interval and number of points.

Appendix B

MATLAB Code

```
clear all; close all; clc;

load subdata.mat

L = 10;
n = 64;

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);

a = zeros(n, n, n);

for j = 1:49
    Un(:,:,:) = reshape(subdata(:,j), n, n, n);
    a = a + fftn(Un);
end
```