HW1: Finding Submarines

Jessica Jones
Jan. 28, 2022

Abstract

Using noisy, broad spectrum recording of acoustics data obtained over 24 hours in half-hour increments, our goal is to locate a submarine in the Puget Sound. This submarine uses new technology that emits an unknown acoustic frequency that needs to be detected. Its location and path need to be determined.

1. Introduction and Overview

Fourier transforms can be done very quickly using a Fast Fourier Transform (FFT) algorithm. This algorithm is commonly used to compute the discrete Fourier transform of a sequence in terms of sines and cosines. However, FFT algorithms work on any function on a finite window, not just oscillatory functions, and gives frequency representations of data. FFTs are also less computationally expensive and are used to denoise data (parsing out important frequency information while ignoring undesirable frequency). Certain concepts (like lowpass filters) are easier to visualize in one domain over another.

To implement an FFT and to scale between spatial and frequency domains in this problem, we defined our spatial domain and Fourier modes to determine sampling space via MATLAB. Through averaging of the Fourier transform over 49 data points in time, as well as visual inspection, we found the distinct frequency signature (center frequency) generated by the submarine.

We also designed and implemented a filter, in this case a Gaussian, to extract this frequency signature to denoise the data and determine the path of the submarine. Ultimately, to deploy a sub-tracking aircraft to keep an eye on the submarine in the future, we determined and plotted the x, y coordinates of the submarine during the 24-hour period.

2. Theoretical Background

2.1 Fourier Transforms

Fourier transform decomposes a signal in time into its frequency components. Fourier transforms assume periodicity on the interval [-L, L]: you will get edge effects if you violate this

assumption. [-L, L) should have discretized 2^n points (although MATLAB will do this for you). In terms of sine and cosine, this equation can be represented as such:

$$f(x) = a_0 + \sum_{n=1}^{\infty} \left(a_n \cos\frac{n\pi x}{L} + b_n \sin\frac{n\pi x}{L} \right) \tag{1}$$

And when solved:

$$a_n = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \cos(nx)\, dx \tag{2}$$

$$b_n = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \sin(nx)\, dx \tag{3}$$

a_1 gives the projection of f(x) onto cos1x, and so on for each orthogonal direction. The definition of a Fourier transform is defined as:

$$F(k) = \frac{1}{2\pi} \int_{-\infty}^{\infty} e^{-ikx} f(x) dx \tag{4}$$

with the inverse being:

$$f(x) = \frac{1}{2\pi} \int_{-\infty}^{\infty} e^{ikx} F(k) dk \tag{5}$$

This transforms *f* from *x* space to *k* space, where *k* is frequency.

2.2 Fast Fourier Transform Algorithms

FFT algorithms are useful for reducing runtime of a regular Fourier transform. They are defined as:

$$F\,(k) = \sum_{k=1}^{N} f(n) e^{\frac{-i2\pi(k-1)(n-1)}{N}} \tag{6}$$

with inverse being:

$$f\,(n) = \frac{1}{N} \sum_{k=1}^{N} F(n) e^{\frac{i2\pi(k-1)(n-1)}{N}} \tag{7}$$

2.3 Time Averaging and Gaussian Filtering

Noise can be modeled to have a normal distributed random variable with zero mean and infinite standard deviation. If you average many measurements with different white noise, the white noise will cancel out, but if the signal is moving, time domain averaging will obscure the signal - the frequency domain is much more stable. For denoising and signal detection via filtering, we used a Gaussian filter to attenuate the frequencies away from the center frequency. Our filter is defined as:

$$f(k) = e^{-\tau(k-k_0)^2} \tag{8}$$

3. Algorithm Implementation and Development

3.1 FFT Algorithm Implementation

FFTs in MATLAB require certain instantiated variables before you can utilize its built-in fft() function.

1. To make a time series to construct a signal, define L (the signal will go from -L to L and the number of points (n)
2. Define n to be the number of Fourier modes you have represented. Our submarine data file contains a matrix with 49 columns of data corresponding to the measurements of acoustic pressure taken over 24 hours.
3. FFTs are scaled from 0 to 2pi in n increments (where n is the number of points in the time domain) – so we rescaled from -L to L (where L is the pre-defined frequency window)

We constructed a grid in 3D in normal space with meshgrid() function, and reshaped each row of data into 3D. meshgrid() adds dimensionality to make a grid – this allows you to make a pair of time points (T) by measurements (S) matrices and a pair of frequency points (K) by measurements (S) matrices - a signal (U) can then be constructed based on T and K.

The measurements themselves are 3D, so we will reshape() to change the vector into a 64 x 64 x 64 grid to make this data uniformed at each time point.

3.2 Filtering Implementation

Its possible to pull out unknown frequencies by averaging in the frequency domain, then you can construct a filter based around that frequency, allowing you to get a meaningful signal even if you don't know *a priori* what frequency you're looking for. The Matlab command fft() gives a series of n complex numbers that are not easily visually displayed – we use fftshift() to plot the frequencies in order.

For filtering, we first found the signature frequency. We used a 3D Gaussian function with signature frequency as a center and used it to project the entire row of data. The bandwidth

was set at 0.02 arbitrarily. We saw that a smaller value will give a larger bandwidth. We multiplied the Fourier transformed signal and the Fourier transformed filter in the frequency domain, then took the inverse Fourier transform using ifftn() to get the filtered signal in the time domain. Finally, using the plot3() function and the 3 position vectors we tracked the movement and final position of the submarine.

4. Computational Results

We found the signature frequency of the submarine to be *kx = 55, ky=6, kz= 26*. Using this signature frequency, I was able to filter the signal and obtain a plot of the movement of the submarine over time. To locate the submarine, an acoustic wave should be focused on *x = 5:625; y = 2.65; z = 8.43.*

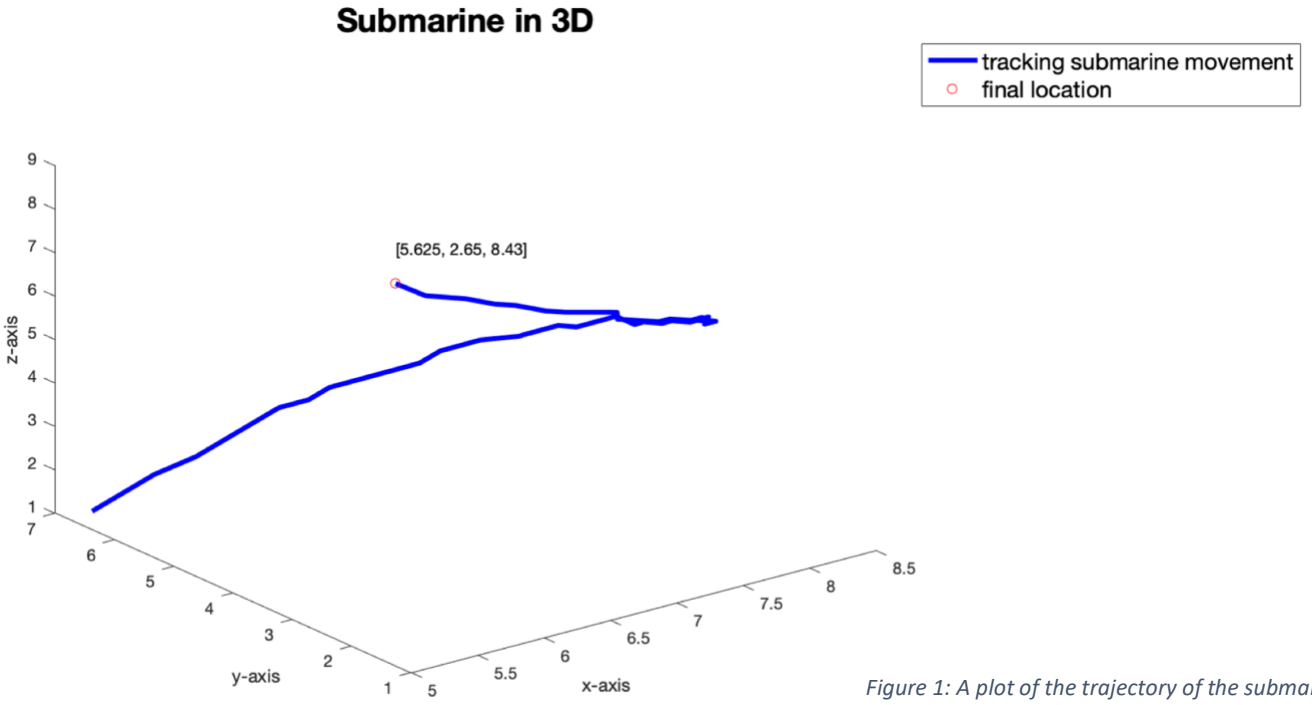

Figure 1: A plot of the trajectory of the submarine plotted with the Gaussian filter applied

5. Summary and Conclusions

Through averaging of the Fourier transform and visual inspection, we determined the frequency signature (center frequency) generated by the submarine to be [55, 6, 26]. We designed and implemented a Gaussian Filter to extract this frequency signature in order to denoise the data and determine the path of the submarine. We determined and plotted the x,y, z coordinates of the submarine during the 24 hour period and found the final position of the submarine to be:

$$[x, y, z] = [5.65, 2.65, 8.43]$$

located in the Puget Sound.

6. Appendix: MATLAB Functions

- `load:` Used to upload raw ultrasound data.
- `linespace(X, Y, n):` generate n points between X and Y.
- `meshgrid():` creates a grid with grid vectors in desired dimension(s).
- `reshape():` Changes ultrasound data into 49 separate 64 by 64 by 64 arrays.
- `fftn():` fast Fourier transform function. transforms spatial domain to the frequency domain.
- `fftshift():` rearranges a Fourier transform X by shifting the zero frequency component to the center of the array.
- `max():` Finds the max value in an array. Used to find the max amplitude among the frequency signals.
- `abs():` returns the absolute value of the input value.
- `ind2sub():` Finds the indices of the max value.
- `ifftn():` inverse fast Fourier transform function. Transforms frequency domain back to the spatial domain.
- `plot3():` Generates a plot in 3 dimensions of path and location of submarine