AMATH 482 Homework 1

David Warne

January 27, 2021

Abstract

Time-frequency analysis and Fourier transforms are extremely powerful tools that can be used for a multitude of data science related tasks. For the purpose of this assignment, these methods are implemented to mimic the tracking of a submarine in the Puget Sound.

1 Introduction and Overview

In this assignment, I am essentially hunting for a submarine in the Puget Sound using noisy acoustic data. The data that I was provided with spans a 24 hour period, and is split into half-hour increments (resulting in a total of 49 realizations). Using this data, I am tasked with finding the frequency signature of the submarine, as well as the path traveled by it. This is done through the implementation of fourier transforms and time frequency analysis.

2 Theoretical Background

As we learned from our textbook [1], the concept of Fourier series is vastly applicable to practices in data science. The information that is necessary to understand for this particular assignment is the way that a fourier transform behaves, and why it is particularly important for signal processing. The fourier transform helps us move from the time domain to the frequency domain, while the inverse fourier transform helps us move from the frequency domain back to the time domain. Why is this important to know? Great question! Simply put, we cannot compute the necessary calculations in the time domain. The noise that we are trying to get rid of in all of these signal processing problems is only present in the frequency domain, so it would be pointless to try and denoise a signal in the time domain. That being said, it is also important to be aware of the Heisenberg Uncertainty Principle, because the one major drawback of the fourier transform is that we will never be able to acquire perfect data in both the time and frequency domains (we'll always have to settle for a bit of both). Regardless, the fourier transform provides various ways to interpret signal processing data, and that is what will be explored in the rest of this assignment.

Below is the main equation that I use to filter the submarine data around the frequency signature:

$$f(x,y,z) = e^{a*((Kx-x0)^2+(Ky-y0)^2+(Kz-z0)^2)} \tag{1}$$

Where a determines the size of the window, Kx, Ky, and Kz are the possible frequency values, and x0, y0, and z0 are the centered frequencies. This is better known as a Gaussian, and it helps to de-noise the data.

3 Algorithm Implementation and Development

There are essentially two algorithms. One that was used to compute the frequency signature of the submarine and one that was used to compute the path traveled by it. See Algorithm 1 for averaging the data and finding the frequency signature, and see Algorithm 2 for filtering the data and determining the path of the submarine.

Algorithm 1: Averaging the Submarine Data

Import data from `subdata.mat`
Initialize all important variables (i.e. linspace and the frequency domain
for $j = 1 : realizations$ **do**
  Reshape subdata(:, j) into a 64x64x64 matrix
  Fourier transform the reshaped data, and add it to the empty, preset average matrix
end for
fftshift the averaged submarine data
take the absolute value and normalize by realizations
Find the max frequencies from the averaged and normalized matrix (and their indices), then use ind2sub to extract the values corresponding to those indices.
Plug in the extracted values to Kx, Ky, Kz to acquire the centered frequencies

Algorithm 2: Filtering the Data and Finding the Submarine Path

Set up Gaussian 1 with a = 1
implement fftshift on the gaussian, as well as an empty 49x3 matrix to store coordinates for each timestamp
for $j = 1 : realizations$ **do**
  Reshape subdata(:, j) into a 64x64x64 matrix
  Fourier transform the reshaped data, and use element-wise multiplication to apply the filter
  use ifftn to transform back to time domain
  use max() to determine indices of frequencies
  use ind2sub() to determine coordinates in time domain from the found indices
  save coordinates in corresponding path matrix
end for
Use plot3 to plot the path of the submarine

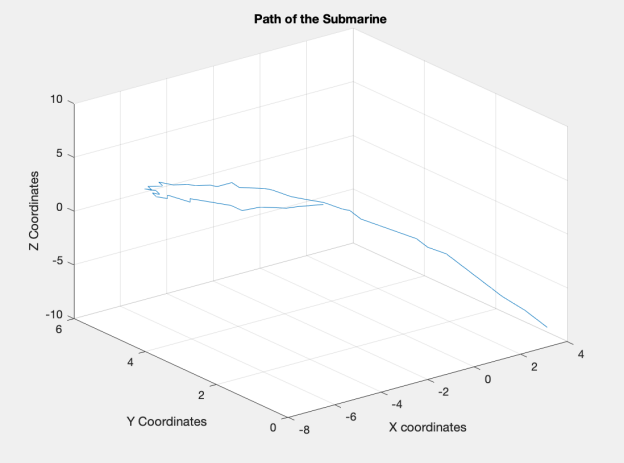

Figure 1: Pictured is the resulting path of the submarine that I was able to recover from the provided data.

4 Computational Results

- Part 1: The frequency signature of the submarine is:(kx,ky,kz) = (5.3407, -6.9115, 2.1991)
- Part 2: See Figure 1 to see the path of the submarine.
- Part 3: I was unable to figure out how to import a table to latex since I'm relatively new to the platform, so instead here are the final coordinates of where the submarine ends its journey: (x,y,z) = (-5.0265, 0.9425, 6.5973)

5 Summary and Conclusions

In conclusion, I think the most important thing to take away from this assignment in particular is how to properly use fourier transforms to declutter signals and determine whether or not a signal is present. Part 1 highlights how the use of an average is so simple yet so effective in helping to find the center frequencies, which is something I can really appreciate. Part 2 highlights how we can compute frequency values in fourier space, then inverse transform back into the time domain to compute spatial coordinates. Part 3 allows us to see the true path of the submarine, and to go from a massive matrix of complex numbers to a detailed path is very satisfying to see indeed!

References

[1] Jose Nathan Kutz. *Data-driven modeling & scientific computation: methods for complex systems & big data*. Oxford University Press, 2013.

Appendix A MATLAB Functions

Here are the main functions that I used in this assignment:

- `[M, I] = max(abs(*data*),[],'all', 'linear')` returns the max absolute values for the given data, as well as the linear indices at which they occur. This is used to de-noise and determine the signal frequencies.
- `[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.

- `[I1,I2,I3,...,In] = ind2sub(SIZ,IND)` returns N subscript arrays I1,I2,..,In containing the equivalent N-D array subscripts equivalent to IND for an array of size SIZ. I utilize this specific function in order to determine spatial coordinates from de-noised frequencies.
- `plot3()` used to plot the path of the submarine.

Appendix B MATLAB Code

Here is the MATLAB code that was used to produce the information in this assignment.

```
% Part 1

% Clean workspace
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
realizations = 49;

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

% create an empty matrix that is 64x64x64 to store the averaged values in
Uave = zeros(n, n, n);

% using the template from lecture 3, we can fourier transform our given
% data for each realization and then add it to our average matrix
for j=1:realizations
    Un(:,:,:)=reshape(subdata(:,j),n,n,n);
    Uave = Uave + fftn(Un);
end

% we have to shift these average values since the k values that we are plotting
% against have already been shifted
Uave = abs(fftshift(Uave))/realizations;

% Averaging reveals the most prominent (centered) frequencies, so we can find those
% values in every direction by using the max function.
[M, I] = max(abs(Uave),[],'all', 'linear');

close all, isosurface(Kx,Ky,Kz,abs(Uave)/M,0.7)
axis([-10 10 -10 10 -10 10]), grid on

% Using the indices of our centered frequencies, we can use the built in
% method ind2sub to acquire their subscript location
[x_val, y_val, z_val] = ind2sub([n n n], I);

% Plugging in the subscript locations of the centered frequencies into each
% dimension Kx, Ky, and Kz will reveal the frequencies we have been
% searching for
x_center = Kx(x_val, y_val, z_val);
y_center = Ky(x_val, y_val, z_val);
z_center = Kz(x_val, y_val, z_val);
```

Listing 1: Code for Part 1

```
%% Part 2 -> Denoising the Data and Plotting the Path

% it will take on the normal form of exp(-a*(t-t).^2), however we are
% working in 3 dimensions so we need to account for all 3 of our center
% values

a = 1; % keep a at 1 so we retain accuracy in both time and frequency domains
g = exp(-a*((Kx-x_center).^2 + (Ky-y_center).^2 + (Kz - z_center).^2));

% because our filter is being applied to our submarine data, and the
% submarine data is already in fourier space, we need to shift the filter
% data accordingly
gt = fftshift(g);

% Initialize a matrix that will store an x y and z coordinate for each
% timestamp
path = zeros(49,3);
for i = 1:49
    Un(:,:,:)=reshape(subdata(:,i),n,n,n);
    Unt = fftn(Un);
    filtered = gt.*Unt;

    time_domain = ifftn(filtered);

    % repeat process of finding indices for centered frequencies
    % We do this in order to get coordinates for each timestamp
    [M, I] = max(abs(time_domain), [], 'all', 'linear');
    [xcoordinate, ycoordinate, zcoordinate] = ind2sub([n n n], I);
    path(i, 1) = Kx(xcoordinate, ycoordinate, zcoordinate);
    path(i, 2) = Ky(xcoordinate, ycoordinate, zcoordinate);
    path(i, 3) = Kz(xcoordinate, ycoordinate, zcoordinate);
end

plot3(path(:,1), path(:,2),path(:,3)), grid on
xlabel('X coordinates')
ylabel('Y Coordinates')
zlabel('Z Coordinates')
title('Path of the Submarine')
```

Listing 2: Code for Part 2

```
%% Part 3:

% We want to reveal the x and y coordinates to follow the submarine:
x_coor = path(:, 1);
y_coor = path(:, 2);

coordinates = table(x_coor,y_coor)
```

Listing 3: Code for Part 3