AMATH 482 Homework 1

Shane Fretwell

January 24, 2020

Abstract

Almost any nautical vessel utilizes active sonar, and thus creates acoustic disturbances that, with enough determination, can be detected and even tracked. In this paper, we explore a theoretical scenario in which we must track a submarine that emits an unknown frequency as it travels through the Puget Sound. At our disposal are acoustic measurements taken every half hour for 24 hours, at each point within a 64x64x64 grid of sensors. With the tools of signal processing, we are able to precisely track the position of the submarine over the course of the 24 hour period.

1 Introduction and Overview

A task such as this entails signal decomposition, cleaning, and localization.

1.1 Decomposition – More Precise than Visual Inspection

Given a single frequency signal, you can tell if a frequency is relatively high or low, but discerning whether a signal is of a specific frequency is imprecise by eye, especially when a signal consists of multiple overlapping frequencies. Signal decomposition removes the guesswork by revealing the amount of each frequency that is present in a signal. In tracking the sub, we will decompose and reconstitute the signal often.

1.2 Cleaning – The Removal of Noise

The provided measurements are saturated with white noise that must be mitigated in order to; one, discern the frequency emitted by the submarine, and two, single out that frequency from the rest. These two instances of cleaning may seem similar from that description, but we will see that they each require very different signal processing methods.

1.3 Localization – Knowing *When*, Not Just *If*

Complete decomposition of a signal gives you complete information of the frequencies within the signal, at the cost of any information about the time each frequency is present. In the context of tracking the submarine, we would know *if* the submarine was detected at a given point in the sensory grid, but not *when*. We need to know when the submarine was at each point in its path to predict its location in the future. To know when, we will decompose the signal in parts and combine the information about each part to get a fuller picture of the when and where of things.

2 Theoretical Background

2.1 Fourier Series

The cornerstone of this analysis is the Fourier Transformation, which, as we discussed in lecture, is the decomposition of intensity over time and/or space into its composite frequencies. Sine and cosine waveforms are used for our purposes because this is the natural waveform of acoustic phenomena. Thus, we have the Fourier series constructed from sines and cosines of a given signal function s(x):

$$\boldsymbol{s_N(x) = \frac{a_0}{2} + \sum_{k=1}^{N} a_k cos\left(\frac{2\pi kx}{P}\right) + b_k sin\left(\frac{2\pi kx}{P}\right)} \tag{1}$$

The coefficients a_k and b_k are given by inner product of the waveforms of frequency k and the signal $s(x)$.

2.2 Frequency Filtering

The fourier series gives us the frequencies present in a signal, but in real-world applications, a measured signal will never be free of noise: frequencies that are both irrelevant and detrimental to signal analysis. Some sources of such noise include radio, heavy equipment, and lightning. If you know what frequency you mean to detect, then you can dampen the rest in the frequency space by multiplying all the frequencies by a filter function centered at the desired frequency. This filter function must be symmetric to avoid introducing bias towards frequencies higher or lower than that which you are hoping to detect, and it must have larger values at its center than further away from center. Gaussian curves are a popular choice due to their easily modifiable center values (b) and widths (a):

$$\boldsymbol{g(k) = e^{-a(k-b)^2}} \tag{2}$$

Thus, a Fourier series filtered by a Gaussian, centered at the b^{th} frequency, and with a width of a is given by:

$$\boldsymbol{s_N(x) = \frac{a_0}{2} + \sum_{k=1}^{N} \left(a_k cos\left(\frac{2\pi kx}{P}\right) + b_k sin\left(\frac{2\pi kx}{P}\right)\right) e^{-a(k-b)^2}} \tag{3}$$

A plot of equation 3 over values of x will give a de-noised version of $s(x)$. $s_N(x)$ will have the appearance of the desired frequency if and only if that frequency was present in the measured signal, making visual inspection possible even in the presence of noise (1).

2.3 Averaging of the Spectrum

In many cases, the frequency of the signal one wishes to detect is unknown. When you know only that there is a signal to be detected and not what the frequency of that signal may be, averaging of the spectrum is a powerful tool. If a measured signal had no noise present, then a Fourier transform would be sufficient to see frequencies present. Thus, if we can remove noise from a signal, then the frequency to detect is apparent. Most noise is sufficiently analogous to white noise, such that treating it as such is reasonable. White noise has the useful property of consisting of a zero-mean uniformly distributed range of frequencies, so the average of the frequency coefficients (a_k and b_k) of white noise signals tends towards a value of zero. In other words, the frequencies of multiple instances of white noise tend to 'cancel out.'

Given multiple signals containing noise approximate to white noise, the sum of the frequency coefficients will tend to lose its noise, leaving only the frequencies that are generated by non-pseudorandom sources (1).

2.4 Gabor Transform

Signal filtering can be used to discern with sufficient accuracy whether a given frequency is present in a signal, but it will not reveal anything about when that frequency occurred. To determine when a frequency was detected within the signal, one can make use of a Gabor transform, which essentially performs the Fourier transform on overlapping windows of time within a signal. This reveals whether a given frequency was present in the signal within a given window of time.

Performing a Gabor transform is done by multiplying a signal by Gaussian filters (like those used in frequency filtering) that are centered at many different times. This removes any frequency information from outside of the time window, and thus when a Fourier transform is performed on this new signal, only frequencies within the time window will be present. The width of the filter will determine the width of the waveforms that are possible to detect, so care must be taken in choosing appropriate width for the problem at hand. A narrow filter may increase the precision of time information gleaned, but it also makes impossible the detection of frequencies wider than the filter itself (1).

3 Algorithm Implementation and Development

3.1 Determining the Submarine's Frequency Signature

Averaging of the spectrum using measurements over time from each point in the grid as an instance of white noise reveals the frequencies that are least likely to be attributed to noise alone.

Algorithm 1: Averaging of the Spectrum

```
for i, j , k ∈ 1:64
    avg_frequencies += frequenciesOf(signalFrom(i, j, k));
end
avg_frequencies = avg_frequencies / 64^3
```

3.2 Filtering by the Possible Signatures

Averaging of the spectrum revealed two possible frequency signatures for the submarine, so we create one filter centered at each frequency and see which filter produces the best results after further processing.

Algorithm 2: Frequency Filtering

```
freqfilter = exp(-b*(ks - a).^2); % Filter of width b centered at a
decomposed = frequenciesOf(signalFrom(i, j, k));
freqfiltered = decomposed*freqfilter;
signalfiltered = recombineToSignal(freqfiltered);
% The greater the max value of signalfiltered, the more likely that frequency a is present in the original signal
```

3.3 Localizing in Time

Utilizing frequency filtering, we determine the presence of the possible frequency signatures at each point in the sensor grid within time frames centered at each half hour, giving us sufficient information about both time and space to track the submarine. Specifically, the submarine's path is tracked as 'hits' wherein the presence of the submarine's frequency signature was detected. This algorithm is a more refined version of Algorithm 2.

Algorithm 3: Frequency Filtering and Time Localization

```
freqfilter1 = exp(-b*(ks + 0.94).^2);
freqfilter2 = exp(-b*(ks + 7.54).^2);
for tau = 1:49
  filter = exp(-a*([1:T] - tau).^2); % Define the time frame with width a
  for t=1:T % Apply the filter to the signal at all points in space
    localized(:,:,:,t) = signal(:,:,:,t) .* filter(t);
  end
  for i, j , k ∈ 1:64
     timeseries = localized(i, j, l, :);
     decomposed = fft(timeseries,n);
     filteredfreq1 = decomposed.*freqfilter1;
     filteredfreq2 = decomposed.*freqfilter2;
     filteredsignal1 = ifft(filteredfreq1);
     filteredsignal2 = ifft(filteredfreq2);
     if (max(filteredsignal1) > hitthreshold), then hits1.append([i,j,k,tau]);
     if (max(filteredsignal2) > hitthreshold), then hits1.append([i,j,k,tau]);
  end
end
```

---

4 Computational Results

4.1 Visual Analysis of the Submarine's Frequency Signature

Before averaging the spectrum, there were too many frequencies that stood out as possibly man-made, as is seen in Figure 1 at left. After averaging, however, only two frequencies stand out as possibly man-made due to their rough adherence to a Gaussian shape. This is shown in Figure 1 at right, with approximate Gaussian curves in blue, centered at the frequencies in question: -7.54 and -0.94.

4.2 Plotting the Hits

Given the data on whether each possible frequency signature was detected at a given point at a given time, we can plot these hits to reveal possible structures that belie non-pseudorandom phenomena, like a submarine moving through space.

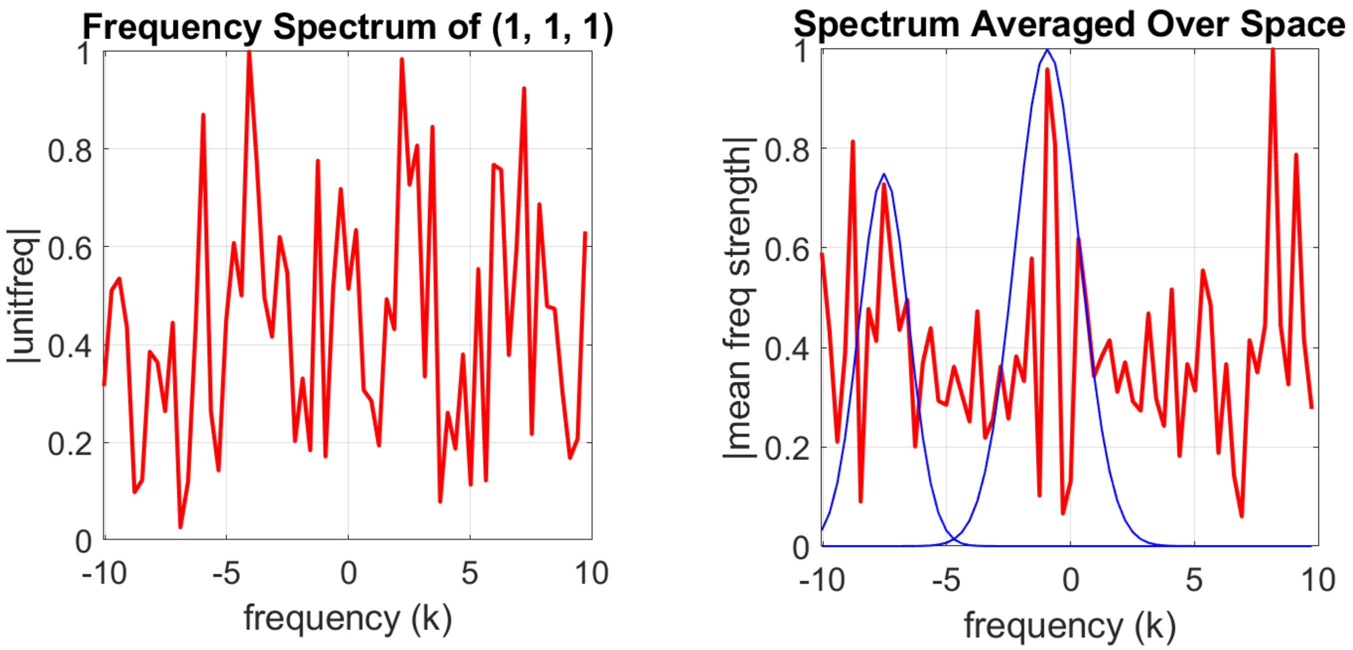

Figure 1: Unprocessed frequency spectrum of the point (1, 1, 1) at left, and the averaged spectrum at right, with added approximations of the sub's frequency signature in blue.

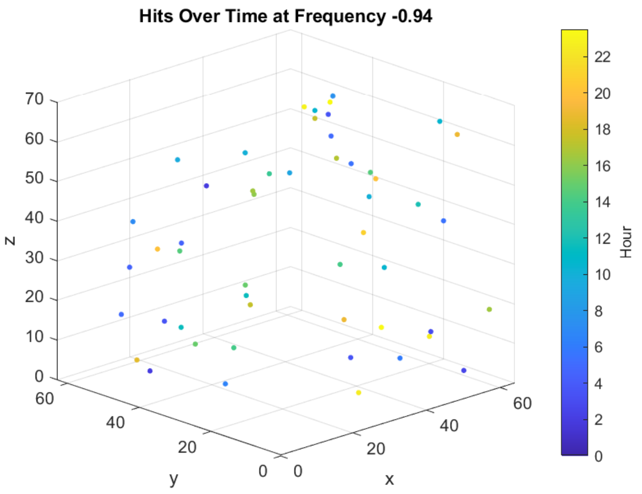

Figure 3: The absence of structure within the hits at Frequency -0.94

4.2.1 Frequency -0.94

Filtering for the frequency -0.94 at all points in time and space did yield a significant number of hits, but as seen in Figure 3, there is no discernable correlation to the hits in space or time. A possible explanation for this is the presence of large marine life in the Puget Sound, which could produce measurable acoustic disturbances such as these. But alas, it is clear that no single source is responsible for the hits present in Figure 3, so frequency -0.94 is not the frequency emitted by the submarine.

4.2.2 Frequency -7.54

The hits at frequency -7.54, however, clearly follow what one would expect from a submarine's acoustic signature (See Figure 4).

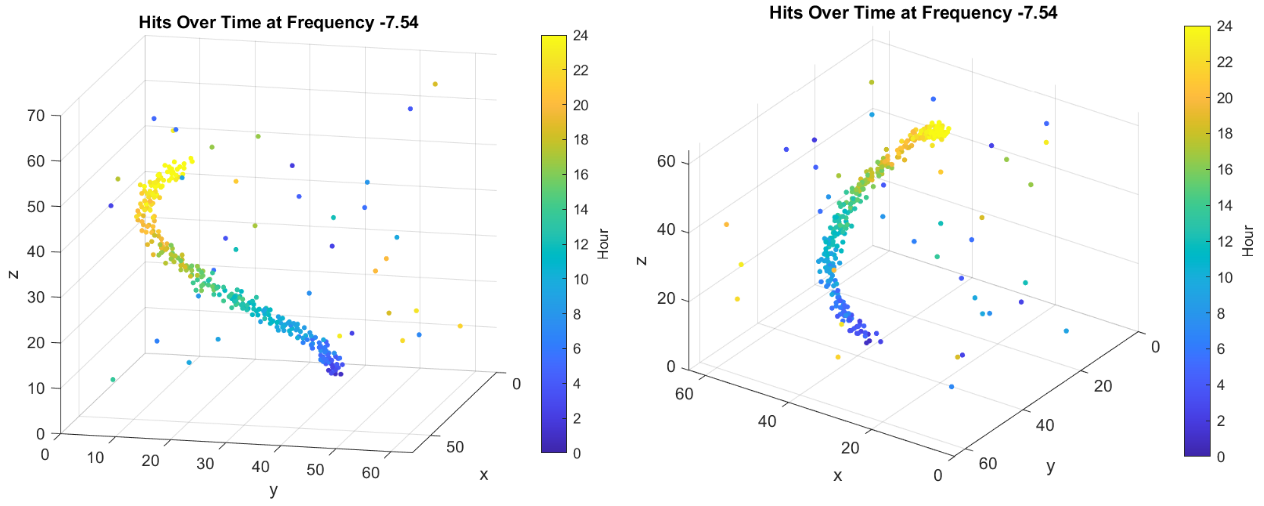

Figure 4: Two viewing angles of the hits at frequency -7.54 colored by time of occurrence.

For the most part, the hits are grouped tightly in space to a structure resembling a helix, and there is a clear progression in time from the beginning of the helix (x≈36, y≈42, z≈9) to the end (x≈38, y≈14, z≈51). The few points outside of this structure can be attributed to noise that was not successfully eliminated. Overall, there is little doubt that there is something in the Puget Sound that is producing this frequency signature as it maneuvers through the water. Given there is only one submarine to be tracked and only one path revealed at frequency -7.54, it is safe to conclude that we have found what we were searching for.

5 Summary and Conclusions

Given acoustic data from the Puget Sound, we were tasked to determine the path of a submarine emitting an unknown frequency signature.

Through averaging of the spectrum, we were able to identify two strong candidates for possible frequency signatures produced by the submarine. By filtering for these two frequencies, we were able to accurately determine the presence of each signature within a given signal. And finally, we were able to modify signal data using a Gabor Transform in order to determine the presence of each signature at different times as well as different points in space. Plotting hits for frequency -7.54 revealed a helical path that is believed to be the path of the submarine.

While it took considerable effort to determine this frequency and the appropriate widths of filters and time windows, we now have everything we need to track this submarine in the future. One needs only to load new data into the MATLAB script *gaboreal.m* to receive a new readout of the submarine's movements over time.

References

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

Appendix A MATLAB Functions

- y = linspace(x1,x2,n) returns a row vector of n evenly spaced points between x1 and x2.
- Un = reshape(data, dim1, …, dimN) returns *data* reshaped into a *dim1*-by-...-by-*dimN* array where *dim1*,...,*dimN* indicates the size of each dimension.
- decomposed = fft(signal) performs a discretized version of the Fourier transform and returns the coefficients of the frequencies present in *signal*
- signal = ifft(decomposed) returns the signal approximated by the sum of decomposed frequencies according to the coefficients in *decomposed*.
- decomposedShifted = fftshift(decomposed) returns the array *decomposed*, previously ordered by positive indices before negative (starting with 0), such that *decomposedShifted* is ordered from least to greatest index.
- decomposed = ifftshift(decomposedShifted) returns the array *decomposedShifted*, previously ordered from least to greatest index, such that *decomposedShifted* is ordered by positive indices before negative (starting with 0).

Appendix B MATLAB Code

main.m – Determining the frequency signature

```
%{
01-07-2021
Shane Fretwell
AMATH 482 Assignment 1, Submarine Tracking
%}

%%

clear; 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
T = 49; % Time points
k = (2*pi/(2*L))*[0:(n/2 - 1) -n/2:-1]; ks = fftshift(k);

Un(:,:,:,:)=reshape(subdata,n,n,n,T);

%% Time-Intensity signals, averaged over realizations from each point in
space
test=reshape(Un(1,1,1,:),[1,T]);
unitfreq=fft(test,n);

figure(1)
plot(ks, fftshift(abs(unitfreq))/max(abs(unitfreq)),'r','Linewidth',2);
set(gca,'Fontsize',16)
title('Frequency Spectrum of (1, 1, 1)')
xlabel('frequency (k)')
ylabel('|unitfreq|')
grid on, axis square

ave = zeros(1, n);
for i=1:n
   for j=1:n
      for k=1:n
         fspace = fft(reshape(Un(i,j,k,:), [1,T]), n);
         ave = ave + fspace;
      end
   end
end
ave = abs(fftshift(ave))./(n^3);

%%
figure(2)
```

```
hold off
plot(ks, ave/max(ave),'r','Linewidth',2);
hold on
plot(ks, exp(-0.3*(ks + 0.94).^2),'b','Linewidth',1);
plot(ks, 0.75*exp(-0.5*(ks + 7.54).^2),'b','Linewidth',1);
set(gca,'Fontsize',16)
title('Spectrum Averaged Over Space')
xlabel('frequency (k)')
ylabel('|mean freq strength|')
grid on, axis square
```

gaboreal.m – Tracking the sub using the known frequency signature

```
clear; close all;

%%
% Clean workspace
clear; 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
T = 49; % Time points
k = (2*pi/(2*L))*[0:(n/2 - 1) -n/2:-1]; ks = fftshift(k);

Un(:,:,:,:)=reshape(subdata,n,n,n,T);

%% Gabor to mark possible hits (freq strength > 0.2)
% Matrix of hit markers at every position and time
hits1 = [];
hits2 = [];

hitthreshold = .46;

b = 0.2; % filter width

freqflt1 = exp(-b*(ks + 0.94).^2);
freqflt2 = exp(-b*(ks + 7.54).^2);

a = .005;
for tau=1:T
    filter = exp(-a*([1:T] - tau).^2); % Define the filter
    % Apply the filter to the signal at all points in space
    UnG=zeros(n,n,n,T);
    for t=1:T
        UnG(:,:,:,t) = Un(:,:,:,t) .* filter(t);
    end
    for i = 1:64
       for j = 1:64
```

```
          for l = 1:64
             timeseries = reshape(UnG(i, j, l, :), [1,T]);
             unitfreq = fft(timeseries,n);
             filteredfreq1 = unitfreq.*freqflt1;
             filteredfreq2 = unitfreq.*freqflt2;

             processed1 = abs(ifft(filteredfreq1, T));
             processed2 = abs(ifft(filteredfreq2, T));

             if max(processed1) > hitthreshold
                hits1(size(hits1, 1) + 1, 1:4) = [i,j,l,tau];
             end
             if max(processed2) > hitthreshold
                hits2(size(hits2, 1) + 1, 1:4) = [i,j,l,tau];
             end
          end
       end
    end
end

%% Plot hits
figure(1)
s1 = size(hits1, 1);
x1 = hits1(:, 1); y1 = hits1(:, 2); z1 = hits1(:, 3);
time1 = hits1(:, 4);
scatter3(x1, y1, z1, 10, time1, 'filled');
title('Hits Over Time at Frequency -0.94')
xlabel('x')
ylabel('y')
zlabel('z')
cb = colorbar('Ticks', 1:4:T,...
         'TickLabels', 0:2:T/2);
cb.Label.String = 'Hour';

figure(2)
s2 = size(hits2, 1);
x2 = hits2(:, 1); y2 = hits2(:, 2); z2 = hits2(:, 3);
time2 = hits2(:, 4);
scatter3(x2, y2, z2, 10, time2, 'filled');
title('Hits Over Time at Frequency -7.54')
xlabel('x')
ylabel('y')
zlabel('z')
cb = colorbar('Ticks', 1:4:T,...
         'TickLabels', 0:2:T/2);
cb.Label.String = 'Hour';
```