AMATH 482 A1: A Submarine Problem

Rishabh Verma

January 27, 2021

Abstract

This paper contains the analysis of 3-D spatial-domain data where each point indicates the acoustic pressure measured at the corresponding coordinate in the Puget Sound. There exists a mysterious submarine emitting an unknown frequency through the waters. Using digital signal processing techniques, I identify this submarine's sonic signature and determine its path through the Puget Sound. These mechanisms can be easily re-applied in the future to locate any other submarine of the same class at any time.

1 Introduction and Overview

A submarine of unknown class is suspected to be in the Puget Sound. There exists audio equipment which can record the acoustic pressure across an evenly spaced 64x64x64 lattice within the Puget Sound. This lattice exists within a 20x20x20 region of unknown physical distance units (pdu's). The actual distance unit does not matter because the goal is to identify the submarine's acoustic signature and trajectory; this can all be given in terms of pdu.

Our data includes 49 snapshots of Puget Sound acoustics taken over a 24-hour span at half-hour increments in time. Each snapshot contains the submarine's acoustic signature as well as a considerable amount of noise. If the noise can be removed, the acoustic signature should look like a wavefront where the strongest displacement is observed by the location of the submarine, and the wavefront's amplitude decays with increasing distance from this maximum. Thus, the filtered data point with maximum amplitude will make a good estimate for the location of the submarine.

1.1 Deliverables

To re-iterate, the expected items to be computed are

- the spatial frequency emitted by the submarine
- the trajectory of the submarine

In order to obtain these, the first item requires denoising the spectrum of the data. The second item requires denoising the data itself in the spatial domain. The methods used for this will rely on applying the Discrete Fourier Transform (DFT) and its inverse.

2 Theoretical Background

2.1 Spatial acoustic data

An acoustic waveform is commonly understood as a sinusoidal displacement function in terms of time. However, an acoustic waveform can also be understood as a sinusoidal displacement function in terms of space. Suppose there exists a point source at $x = 0$ which emits a signal of wavelength 4, and that this signal does not decay with respect to distance. At some instant in time, the reading of acoustic sensors at $x = 1, 2, 3, 4$ might read

$$\begin{bmatrix} x_1 & x_2 & x_3 & x_4 \end{bmatrix} = \begin{bmatrix} 1 & 0 & -1 & 0 \end{bmatrix} \tag{1}$$

in which the RHS sensor values, though dependent on the instantaneous phase of the signal source, indicate a single oscillation. Realistically, a point source of a 3-dimensional acoustic pressure wave will emit a signal whose strength is inversely proportional to the square of the distance. For our purposes, we only need to understand that the signal's amplitude decays with respect to distance from the point source.

2.1.1 An analogy

The notion of recording acoustic pressure but working with spatial data can be disorienting. For this, I offer an analogy. Suppose a researcher by a pond has a device which can freeze time. She walks on the dock, drops a stone into the water, freezes time, and writes down the vertical displacement of the water across a horizontal grid of points. This is an instantaneous 2-dimensional measurement of a longitudinal pressure signal across a lattice of points, and is analogous to our submarine data, though with one dimension lesser. The period of the measured signal is exactly the wavelength of the pressure wave in water.

2.2 Analyzing the Fourier spectrum of spatial data

2.2.1 Output of the Discrete Fourier Transform

The submarine emits a signal which should ideally appear as a periodic spherical wavefront. If an oscillation has fixed period x pdu, it will have fixed spatial frequency $1/x$ pdu^{-1}.

Because the data divides a 20x20x20 physical region into a 64x64x64 lattice, the physical coordinate corresponding to a value at index (x^*, y^*, z^*) can be obtained with a linear interpolation from -10 to 10 with 64 points. This is given by dropping the final point off of MATLAB's `linspace(-20,20,65)`. Note that if `linspace(-20,20,64)` is used instead, then the physical locations will not be centered at zero.

The k-th frequency bin outputted by the DFT corresponds to a wave which is periodic on the domain and includes k oscillations. Such a wave has period D/k.

Consider a wave $\sin(2\pi f x)$ with frequency f and spatial parameter x. Its period is $\frac{1}{f}$, and its natural frequency is $\omega = 2\pi f$.

In the DFT, all oscillations must be periodic by integer order. The k-th bin contains a wave with k oscillations over a domain of width $D = 20$, and so it has period $\frac{D}{k} = \frac{20}{k}$. This means it has frequency $\frac{k}{20}$ and natural frequency $2\pi\frac{k}{20}$. Unless specified otherwise, all frequencies moving forward in this paper are natural frequencies.

It is important to note that the DFT output of bin k is equal to the DFT output of bin $k + N$ where $N = 64$. What this means is that signals with natural frequency

$$\frac{2\pi k}{20}, \frac{2\pi(k+64)}{20}, ..., \frac{2\pi(k+64j)}{20} \forall j \in \mathbb{Z} \tag{2}$$

are indistinguishable. This phenomenon is known as aliasing.

For future applications, the physical wavelength λ of the submarine's signal can be calculated by

$$\lambda = (\frac{\omega}{2\pi})^{-1} \tag{3}$$

where ω is the calculated natural frequency as reported in this paper.

2.2.2 Removing the noise from frequency data

The sensors are able to pick up this signal, but they are also susceptible to noise. Let us suppose that this noise affects each frequency bin with zero mean. Note that this is a weaker assumption than white noise, which includes the assumption that each frequency bin is affected by the same random variable.

Suppose a frequency bin k is subject to zero-mean noise given by the random variable X_i with $E[X_i] = 0$. Suppose there are 49 measurements made with this frequency bin, corresponding to 49 independent random variables X_i for $i = 1, ..., 49$. Then

$$E[\Sigma_{i=1}^{49} X_i] = \Sigma_{i=1}^{49} E[X_i] = 0 \tag{4}$$

If each snapshot contains the signal and noise, what can I expect from computing the spectra of each snapshot and recording the average? The noise will have an expected value of zero. The signal will have a constant-value which remains unaffected by averaging. This is the foundation of our method for denoising the frequency spectrum of the data.

2.2.3 Removing the noise from spatial data

Once the method in section 2.2.3 is applied to denoise the frequency data and the method in section 2.2.2 is applied to identify the signal emitted by the submarine, I can use this information to identify the submarine's trajectory. Given a snapshot, I can compute the DFT. Since the frequency is known, I can filter around that frequency by multiplying each frequency bin by a Gaussian filter centered at that frequency. After filtering, I can compute the inverse DFT and expect the signal emitted by the submarine to be isolated. By the method in section 2.1, the point with the highest amplitude can be approximated as the submarine's current location.

3 Algorithm Implementation and Development

Algorithm 1: Identify submarine's acoustic signature

for each snapshot **do**
  Compute the 3-D Fourier Transform and store in running sum
end for
return the location of the maximum value in the sum of 3-D Fourier Transforms as $(\hat{x}, \hat{y}, \hat{z})$

Algorithm 2: Locate submarine emitting signal with spatial frequency $(\hat{x}, \hat{y}, \hat{z})$

Create a 3-D Gaussian filter centered around $(\hat{x}, \hat{y}, \hat{z})$.
for each snapshot **do**
  Compute the 3-D Fourier Transform
  Apply the 3-D Gaussian filter
  Compute the 3-D Inverse Fourier Transform
  Identify and store the coordinates of the maximum value as the submarine location
end for
return the submarine coordinates in each snapshot

All algorithms require that the data is loaded and reshaped as 49 snapshots of 64x64x64 spatial-domain sensor data.

The 3-D DFT and 3-D inverse DFT are implemented with MATLAB's `fftn(), ifftn()`, which apply Cooley-Tukey's method of Fast Fourier Transform to a signal of arbitrary dimension. See Appendix A for more details.

4 Computational Results

The output of Algorithm 1 is that the data has a strong acoustic signature in the bins corresponding to

$$\begin{bmatrix} \hat{x} \\ \hat{y} \\ \hat{z} \end{bmatrix} = \begin{bmatrix} -6.9115 \\ 5.3407 \\ 2.1991 \end{bmatrix} \tag{5}$$

This is then inputted into algorithm 2 to denoise the signal and identify the submarine's coordinates. These coordinates are plotted in Figure 1. The x-y value of each coordinate is included in Table 1 so that the submarine can be tracked from a horizontal agent.

5 Summary and Conclusions

5.1 Identifying the submarine's signal

Through averaging the 49 spectrums, I was able to cancel most of the noise and isolate the frequency emitted by the submarine. Realistically, negative frequencies are not an actual thing, so I can take the absolute value of these bins to identify the emitted frequency. In accordance with section 2.2.2, these are natural frequencies and include a multiplicative constant of 2π.

The identified frequencies are also subject to aliasing, and so by Equation 2 a frequency $\hat{x}$ is indistinguishable from $\hat{x} + j\frac{2\pi(64)}{20}$ for all integers j.

In order to obtain physical information about the submarine's signal, we can take the natural frequency $\hat{x}$ and compute the physical wavelength with Equation 3. With aliasing taken into account, Equation 6 holds for some integers i, j, k.

$$\begin{bmatrix} \lambda_x \\ \lambda_y \\ \lambda_z \end{bmatrix} = \begin{bmatrix} 2\pi\big(|\hat{x}| + i\frac{2\pi(64)}{20}\big)^{-1} \\ 2\pi\big(|\hat{y}| + j\frac{2\pi(64)}{20}\big)^{-1} \\ 2\pi\big(|\hat{z}| + k\frac{2\pi(64)}{20}\big)^{-1} \end{bmatrix} \tag{6}$$

If our sensor lattice has high enough spatial resolution that there is no aliasing occurring, then this equation can be solved with $i = j = k = 0$ to get the wavelengths of the submarine's signal in unit pdu

$$\begin{bmatrix} \lambda_x \\ \lambda_y \\ \lambda_z \end{bmatrix} = \begin{bmatrix} 0.9091 \\ 1.1765 \\ 2.8571 \end{bmatrix} \tag{7}$$

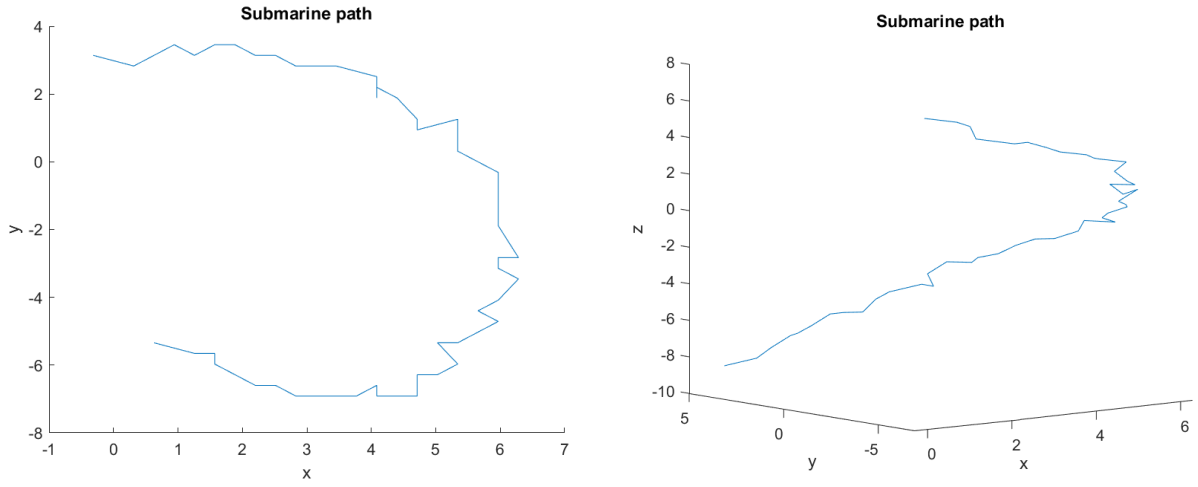
Table 1: Horizontal location of submarine, for an airplane or a seaship to track

x	-0.31	0.31	0.94	1.25	1.57	1.88	2.19	2.51	2.82	3.14	3.45	4.08	4.08
y	3.14	2.82	3.45	3.14	3.45	3.45	3.14	3.14	2.82	2.82	2.82	2.51	1.88
time (hr)	0	0.5	1	1.5	2	2.5	3	3.5	4	4.5	5	5.5	6

x	4.08	4.39	4.71	4.71	5.34	5.34	5.65	5.96	5.96	5.96	5.96	6.28
y	2.19	1.88	1.25	0.94	1.25	0.31	0	-0.31	-0.94	-1.57	-1.88	-2.82
time (hr)	6.5	7	7.5	8	8.5	9	9.5	10	10.5	11	11.5	12

x	5.96	5.96	6.28	5.96	5.65	5.96	5.34	5.02	5.34	5.02	4.71	4.71
y	-2.82	-3.14	-3.45	-4.08	-4.39	-4.71	-5.34	-5.34	-5.96	-6.28	-6.28	-6.59
time (hr)	12.5	13	13.5	14	14.5	15	15.5	16	16.5	17	17.5	18

x	4.71	4.08	4.08	3.76	3.14	2.82	2.51	2.19	1.57	1.57	1.25	0.62
y	-6.91	-6.91	-6.59	-6.91	-6.91	-6.91	-6.59	-6.59	-5.96	-5.65	-5.65	-5.34
time (hr)	18.5	19	19.5	20	20.5	21	21.5	22	22.5	23	23.5	24



(a) Aerial view

(b) 3-D view

Figure 1: Estimated submarine trajectory in Puget Sound over 48 hours.

5.2 Identifying the submarine's trajectory

Once the signal's nature is known, a submarine of this class can be located at anytime by capturing a snapshot of data and applying algorithm 2 to that snapshot. During future encounters with a submarine of this class, this method can come in handy to readily locate it. This method can also form the foundation of a radar system designed to detect such a submarine.

Appendix A MATLAB Functions

- `y = linspace(x1,x2,n)` returns a row vector of `n` evenly spaced points between `x1` and `x2`.
- `[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.
- `ut = fftn(u)` Implements the N-Dimensional Fast Fourier Transform. When passed a 3-dimensional matrix `u`, it applies the 1-D transform along each of the 3 dimensions. Each 1-D transform outputs first the nonnegative bins in increasing order, then the negative bins in increasing order.
- `u = ifftn(ut)` Implements the N-Dimensional Inverse Fast Fourier Transform. Just like the previous bullet, when passed a 3-dimensional matrix `ut`, it applies the 1-D inverse transform along each of the 3 dimensions, outputting first the nonnegative bins in increasing order, then the negative bins in increasing order.
- `us = fftshift(ut)` Rearranges the output of `ut = ffshift(u)` so that the bins are all sorted increasing order.

Appendix B MATLAB Code

```
%% Clean workspace
clear all; close all; clc;
load ../data/subdata.mat;

%%
L = 10;  % spatial domain
```

```
n = 64;  % Fourier modes (data size)
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);  % spatial coordinate grid
[Kx,Ky,Kz]=meshgrid(ks,ks,ks);  % spatial frequency-domain grid

%% Let's average our data in the frequency domain

realize = 49; % number of realizations

aveUt = zeros(n,n,n);
for j = 1:realize
    Un(:,:,:)=reshape(subdata(:,j),n,n,n);
    Ut = fftshift(fftn(Un));
    aveUt = aveUt + Ut;
end

absUt = abs(aveUt)/ max(abs(aveUt), [], 'all');

close all, isosurface(Kx,Ky,Kz,absUt,0.30)
axis([-20 20 -20 20 -20 20]), grid on, drawnow

%% I see a 3-D signal being emitted. Let's identify it.
[M,I] = max(aveUt, [], 'all', 'linear');
dim = [n n n];
[x0,y0,z0] = ind2sub(dim,I);

% Our signal has the frequencies:
ks(x0)
ks(y0)
ks(z0)
% In units of m^-1

%% We now want a 3-D Gaussian filter centered at frequencies x,y,z
% After filtering, we can do:
%     signal -> FFT -> filter -> IFFT
% and the submarine should be visible in each sample.

% Build the filter
filter = ones(n,n,n);
tau = 0.4;
fx = @(x) exp(-tau*(x-x0)^2);
fy = @(y) exp(-tau*(y-y0)^2);
fz = @(z) exp(-tau*(z-z0)^2);
fxyz = @(x,y,z) fx(x)*fy(y)*fz(z);
for x = 1:n
    for y = 1:n
        for z = 1:n
            filter(x,y,z) = fxyz(x,y,z);
        end
    end
end
```

```
% signal -> FFT -> filter -> IFFT
subdata_f = zeros(49, 64,64,64);  % filtered data (still in spatial domain)
for j = 1:realize
    signal = reshape(subdata(:,j),n,n,n);
    signal = fftn(signal);
    signal = signal .* fftshift(filter);
    signal = ifftn(signal);
    signal = abs(signal) / max(abs(signal), [], 'all');
    subdata_f(j,:,:,:) = signal;
end

%% View a Slideshow
frame_duration = 0.5;  % # of seconds each isosurface appears for
iso_strength = 0.8;
for j=1:realize
    frame(:,:,:) = subdata_f(j,:,:,:);
    frame = abs(frame)/max(abs(frame), [], 'all');

    close all, isosurface(X,Y,Z,frame, iso_strength)
    axis([-20 20 -20 20 -20 20]), grid on, drawnow
    pause(frame_duration)
end

%% View One frame
j=43;  % the frame index
frame(:,:,:) = subdata_f(j,:,:,:);
frame = abs(frame)/max(abs(frame), [], 'all');

close all, isosurface(X,Y,Z,frame, iso_strength)
axis([-20 20 -20 20 -20 20]), grid on, drawnow

%% Get and plot the path of the submarine.
% subdata_f has size 49,64,64,64
path = zeros(49,3);
for j = 1:realize
    [M,I] = max(subdata_f(j,:,:,:), [], 'all', 'linear');
    dim = [64 64 64];
    [xj,yj,zj] = ind2sub(dim, I);
    path(j,:) = [ks(xj), ks(yj), ks(zj)];
end

plot3(path(:,1), path(:,2), path(:,3))
xlabel('x')
ylabel('y')
zlabel('z')
title("Submarine path");
```