AMATH 482 Homework 1

lewisxy

January 24, 2020

Abstract

In this paper, we demonstrated the possibility of tracking a submarine from synthetic noisy acoustic signals with Fast Fourier Transforms, white noise removal by averaging, and frequency domain filtering.

1 Introduction and Overview

This paper introduces a method of tracking a submarine from an acoustic recording over time. Due to complex recording situations, the recorded signals are usually too noisy to work with directly. We uses averaging method in frequency domain to reduce the effective noise level in order to find the frequency signature of the submarine. Then using a Gaussian filter centering at the signature frequency of submarine, we can remove the noise from each individual measurement and locate the submarine in spatial domain.

2 Theoretical Background

2.1 Discrete Fourier Transform

The core mathematical tool in our methods is Discrete Fourier Transform (DFT), which allows us to find the frequency composition of an discrete signal. Mathematically, Discrete Fourier Transform (2) is the discrete version of the more generalized Fourier Transform (1).

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

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

With N samples in a discrete signal, Discrete Fourier Transform will provide N frequency component with $k = -N/2, -N/2+1, ..., -1, 0, 1, ..., N/2-1$. Note: Because of the periodic nature of Fourier Transform, k can also be $0, 1, ..., N-1$. We need to scale the frequency by $2\pi/2L$ since Fourier Transform assumes 2π periodic signal within domain $[-L, L]$. In this paper, Discrete Fourier Transform is performed using MATLAB, which implements it with Fast Fourier Transform (FFT) algorithm. This allows us to compute DFT efficiently on a computer. In this paper, we are performing DFT on a 3-dimensional signal. In general, we can perform DFT on arbitary signal by performing one dimensional DFT on each dimension.

2.2 White Noise

White noise is a noise that present on all frequencies. For each frequency, the noise level is a normal random variable with mean of 0 and some variance 1. Let $\hat{s_k}$ be the original signal in frequency domain, A be the noise level, the noisy signal in frequency domain $\hat{x_k}$ can be modeled as the following.

$$\hat{x_k} = \hat{s_k} + AX \text{ for all } k, \quad X \sim N(0, 1) \tag{3}$$

Suppose noise at each frequency is independent, $\text{Var}(AX) = A^2$.

A common way to reduce the effective noise level is to measure a signal repeatedly and take the average of those signals. Let $\bar{\hat{x_k}}$ be the average of n measurements.

$$\bar{\hat{x_k}} = \frac{1}{n}\sum_{i=1}^{n}\hat{x_{ki}} = \frac{1}{n}\sum_{i=1}^{n}(\hat{s_{ki}} + AX_i) = \hat{s_k} + A\frac{1}{n}\sum_{i=1}^{n}X_i \tag{4}$$

Suppose the noise in each measurement is independent, $\mathrm{Var}(A\frac{1}{n}\sum_{i=1}^{n}X_i) = \frac{A^2}{n}$. Therefore, the effective noise level is reduced by a factor of $\sqrt{n}$.

3 Algorithm Implementation and Development

In this section, we describes the different steps to take in order to track the submarine. Implementations are provided in the form of MATLAB code in Appendix B, explanations of selected MATLAB functions are provided in Appendix A.

3.1 Frequency Signature

The presence of white noise in the data suggests we can reduce the noise level by averaging the signals. To compute the average, we apply fast Fourier transform to each measurement and take the average of them. We will be able to observe a spike (a particular frequency with magnitude significant larger than other frequencies) in the averaged signal, and that is the frequency signature of the submarine.

3.2 Filtering

With the frequency signature, we can analyze the data for each measurement individually in spatial domain with the help of filters. We can construct Gaussian filter as the following.

$$F(\hat{x}) = e^{-\tau||\hat{x}-\hat{x_0}||_2^2} \tag{5}$$

$\hat{x}$ is a particular data point in frequency domain, and $\hat{x_0}$ is the frequency signature. By adjusting parameter τ, we can change the width of the filter, a large τ will keep a narrow frequency range. Since we want to analyze the data in spatial domain, to apply the filter, we need to first transform the data into frequency domain, multiply the Filter with the transformed data, and transform back.

3.3 Tracking

After filtering, we removed most of the noise from the signal. Because the position of submarine where the signal comes from, in spatial domain, we can track the position of submarine by finding the position with the largest signal magnitude.

4 Computational Results

From the data description, each measurement is done in spatial domain of $[-10, 10]$ for all 3 dimensions with an resolution of 64 data point per dimension (a particular signal is a $64 \times 64 \times 64$ cube). We have 49 measurements in total.

The frequency signature for the submarine is **(5.3407, -6.9115, 2.1991)**. The path of submarine is given by figure 1 and 2. The position of the submarine at each measurement is provided in table 1.

5 Summary and Conclusions

In this paper, we demonstrated the possibility of tracking a submarine from synthetic noisy acoustic signals with Fast Fourier Transforms, white noise removal by averaging, and frequency domain filtering. These techniques allows us to analyze signal efficiently, and has a wide range of applications. In the real-world, although the noise composition and the designated frequency signature will be much more complicated, the technique used in this paper can still be applied to better understand the underlying signal.

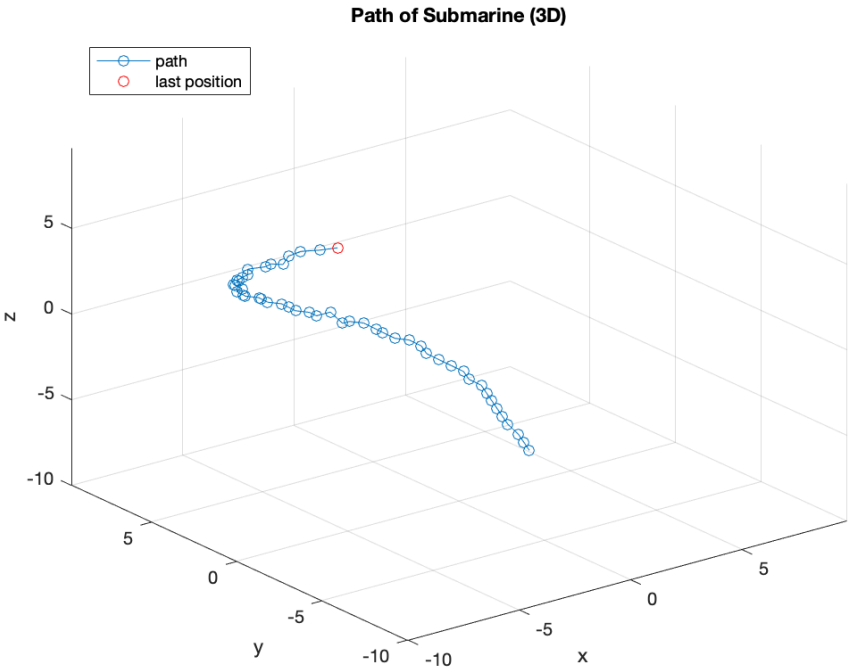

Figure 1: Path of Submarine (3D)

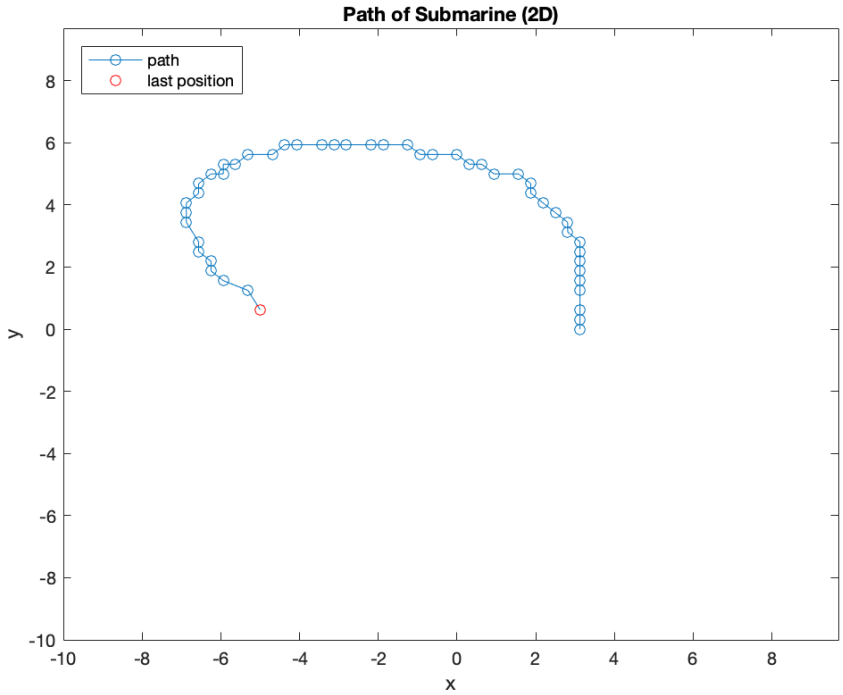

Figure 2: Path of Submarine (2D)

Time	Position (x, y)	Depth (z)
1	(3.1250, 0.0000)	-8.1250
2	(3.1250, 0.3125)	-7.8125
3	(3.1250, 0.6250)	-7.5000
4	(3.1250, 1.2500)	-7.1875
5	(3.1250, 1.5625)	-6.8750
6	(3.1250, 1.8750)	-6.5625
7	(3.1250, 2.1875)	-6.2500
8	(3.1250, 2.5000)	-5.9375
9	(3.1250, 2.8125)	-5.6250
10	(2.8125, 3.1250)	-5.3125
11	(2.8125, 3.4375)	-5.0000
12	(2.5000, 3.7500)	-4.6875
13	(2.1875, 4.0625)	-4.3750
14	(1.8750, 4.3750)	-4.0625
15	(1.8750, 4.6875)	-3.7500
16	(1.5625, 5.0000)	-3.4375
17	(0.9375, 5.0000)	-3.1250
18	(0.6250, 5.3125)	-2.8125
19	(0.3125, 5.3125)	-2.5000
20	(0.0000, 5.6250)	-2.1875
21	(-0.6250, 5.6250)	-1.8750
22	(-0.9375, 5.6250)	-1.8750
23	(-1.2500, 5.9375)	-1.2500
24	(-1.8750, 5.9375)	-1.2500
25	(-2.1875, 5.9375)	-0.9375
26	(-2.8125, 5.9375)	-0.6250
27	(-3.1250, 5.9375)	-0.3125
28	(-3.4375, 5.9375)	0.0000
29	(-4.0625, 5.9375)	0.3125
30	(-4.3750, 5.9375)	0.6250
31	(-4.6875, 5.6250)	0.9375
32	(-5.3125, 5.6250)	1.2500
33	(-5.6250, 5.3125)	1.5625
34	(-5.9375, 5.3125)	1.8750
35	(-5.9375, 5.0000)	2.1875
36	(-6.2500, 5.0000)	2.5000
37	(-6.5625, 4.6875)	2.8125
38	(-6.5625, 4.3750)	3.1250
39	(-6.8750, 4.0625)	3.4375
40	(-6.8750, 3.7500)	3.7500
41	(-6.8750, 3.4375)	4.0625
42	(-6.8750, 3.4375)	4.3750
43	(-6.5625, 2.8125)	4.6875
44	(-6.5625, 2.5000)	5.0000
45	(-6.2500, 2.1875)	5.0000
46	(-6.2500, 1.8750)	5.6250
47	(-5.9375, 1.5625)	5.9375
48	(-5.3125, 1.2500)	5.9375
49	(-5.0000, 0.6250)	6.2500

Table 1: Submarine positions

Appendix A MATLAB Functions

- `fftn(x)`: perform n-dimensional Fast Fourier Transform (FFT) on the input. n is the dimension of input.
- `fftshift(x)`: shift the output from `fft` in all dimensions so that frequencies are arranged in increasing order.
- `arr(:)`: flatten an array of higher dimensions to 1-dimension.
- `[i1, ..., in] = ind2sub(sz, idxs)`: convert index in 1-diemension (`idxs`) format to index in dimensions defined by `sz`.

Appendix B MATLAB Code

```
%% Clean workspace
clear all; close all; clc

% Imports the data as the 262144x49 (space by time) matrix called subdata
load('subdata.mat');

%% Setup constants
L = 10; % spatial domain
n = 64; % Fourier modes
ntime = 49; % number of time instance
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); % grid for space
[Kx,Ky,Kz] = meshgrid(ks,ks,ks); % grid for frequency

%% Step 1: locate the frequency signature by averaging in frequency space
ut_sum = complex(zeros(n, n, n));
for j = 1:ntime
    uj(:,:,:) = reshape(subdata(:,j), n, n, n);
    utj = fftn(uj);
    ut_sum = ut_sum + utj;
end
ut_avg = ut_sum / ntime;
utm_avg = abs(ut_avg); % signal magnitude in frequency space

tmp_ = fftshift(utm_avg);
[mv, idx] = max(tmp_(:));
[i1, i2, i3] = ind2sub([n, n, n], idx);
% Note: meshgrid treat horizontal axis as x and vertical axis as y
% this is different from index notation (row, column)
% to keep consistency with meshgrid, we swap x and y from index
fx = ks(i2); fy = ks(i1); fz = ks(i3); % actual center frequency
fprintf('frequency signature: (%.4f, %.4f, %.4f)\n', fx, fy, fz);

%% Step 2: make a gaussian filter using the frequency signature we found
% the larger tau the faster filter decays
tau = 0.1;
F = exp(-tau*((Kx - fx).^2 + (Ky - fy).^2 + (Kz - fz).^2));

%% Step 3: track the submarine by applying filter in each timer instance
```

```
mvs = zeros(ntime, 1);
idxs = zeros(ntime, 1);
for j = 1:ntime
    uj(:,:,:) = reshape(subdata(:,j), n, n, n);
    utj = fftshift(fftn(uj));
    utfj = F.*utj; % apply filter
    ufj = ifftn(ifftshift(utfj));
    tmp_ = abs(ufj);
    [mv, idx] = max(tmp_(:));
    mvs(j) = mv;
    idxs(j) = idx; % record the position at each measurement
end

% compute the path
% note: we swap x and y in index notation to be consistent with meshgrid
[iy, ix, iz] = ind2sub([n, n, n], idxs);

%% plot the path of submarine 3D
figure;
plot3(x(ix), y(iy), z(iz), '-o');
hold on;
plot3(x(ix(end)), y(iy(end)), z(iz(end)), 'o', 'Color', 'red');
axis([x(1) x(end) y(1) y(end) z(1) z(end)]);
grid on;
xlabel('x');
ylabel('y');
zlabel('z');
legend('path', 'last position', 'Location','northwest');
title('Path of Submarine (3D)');

%% plot the path of submarine 2D
figure;
plot(x(ix), y(iy), '-o');
hold on;
plot(x(ix(end)), y(iy(end)), 'o', 'Color', 'red');
axis([x(1) x(end) y(1) y(end)]);
xlabel('x');
ylabel('y');
legend('path', 'last position', 'Location','northwest');
title('Path of Submarine (2D)');

%% print positions
for j = 1:ntime
    fprintf('%d: position: (%.4f, %.4f, %.4f)\n', j, x(ix(j)), y(iy(j)), z(iz(j)));
end
```

Listing 1: Complete MATLAB code