AMATH 582 Homework 1

Qiangqiang Guo

Due January 27, 2021

1 Introduction

There is a submarine in the Puget Sound. It is a new submarine technology that emits an unknown acoustic frequency. Using a broad spectrum recording of acoustics, data is obtained over a 24-hour period in half-hour increments. Unfortunately, the submarine is moving, so its location and path need to be determined. This project aims to identify the acoustic admissions of this new class of submarine, locate the submarine and find its trajectory using the acoustic signature. In addition, a future trajectory point is predicted so that the sub-tracking aircraft could track it.

The submarine is moving in the time and spatial domains, making it hard to track directly. Considering that the frequency that the submarine emits is fixed, we should look into the frequency domain. However, the acoustic frequency that the submarine emits is unknown and there is noise (assuming it is random noise) in the acoustic data. We will first eliminate the noise by averaging the data in frequency domain. In this way, the random noise should diminish and the frequency generated by the submarine should show up. Once we found the center frequency, we can filter the data around the center frequency to further denoise the data and determine the path of the submarine. Finally, we should predict the future trajectory of the submarine so that the sub-tracking aircraft can track it.

2 Theoretical Background

Based on above discussions, there are three key tasks in this project:

1. Average the frequency domain data to denoise the data and find the center frequency.
2. Filter the data around the center frequency to further denoise the data and determine the path of the submarine.
3. Predict the trajectory of the submarine.

For the first task, we use Fourier transform to transform the temporal-spatial acoustic data into frequency domain. The Fourier transform and its inverse are defined as

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

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

In MATLAB, we use the Fast Fourier transform (FFT) to perform the forward and backward Fourier transforms.

For the second task, once we have found the center frequency k_0, we can use a Gaussian filter to filter to further denoise the data:

$$\mathcal{F}(k) = e^{-\tau(k-k_0)^2} \tag{3}$$

where τ is the bandwidth of the filter.

For the third task, we can use a curve fitting method (e.g., polynomial fitting) to predict the future trajectory of the submarine, assuming that the submarine will not change direction and speed dramatically.

3 Algorithm Development and Implementation

3.1 Setup

This problem is defined in a 3-dimension domain. The spatial domain is $[-10, 10]$ and the Fourier modes is 64, which means we divided each of the 3 axis into 64 discrete points. Since FFT assumes 2π periodic signals, we should re-scale the wavenumbers by $\frac{2\pi}{2L}$. Due to the same reason (periodic signals), the first and last points should be the same, therefore, only the first 64 Fourier modes should be considered. In addition, we can shift the original wavenumbers (i.e., $k = (2\pi/(2L)) * [0 : (n/2-1) - n/2 : -1]$) back to its mathematically correct positions.

3.2 Denoise the data and find the center frequency

We first reshape each column of the acoustic data (i.e., the data at one time slot) to $64 \times 64 \times 64$ and transform it to frequency domain. Then, we average the frequency domain data for all time slots (i.e., 49 time slots). Assume that the noises are random, averaging the spectrum data should reduce the noise and make the frequency of the submarine clearer. Finally, we normalize the average frequency data on $[0, 1]$. After eliminating the random noise (at least to some extend), the frequency generated by the submarine should dominate the average frequency data. So we can select the frequency with the maximum amplitude as the center frequency.

3.3 Filter the data and find the path of the submarine

We use a Gaussian filter (i.e., Equation 3) to further denoise the data. The bandwidth is set to 0.2 and k_0 is the center frequency found above. We iterate over all time slots, for each time slot, we do the following: 1) reshape the data of each column to $64 \times 64 \times 64$; 2) transform it to frequency domain by FFT; 3) apply the Gaussian filter; 4) transform the filtered frequency data back to spatial domain; 5) find the position of the submarine by selecting the position with maximum amplitude; 6) record the position data for current time slot.

3.4 Predict the trajectory of the submarine

To predict the trajectory in the near future, we assume that the submarine hasn't noticed our detection thus it will not change direction and speed dramatically. We use a polynomial fitting method to predict the trajectory. The last 10 hours' data (i.e., 20 points of (x, y)) are used to do the fitting.

3.5 Overall algorithm

In summary, the algorithm of this whole project is shown in Algorithm 1.

Algorithm 1 Hunt the submarine

Import data from `subdata.mat`
Setup the parameters: spatial domain, number of Fourier modes, wavenumbers, etc.
for $j = 1 : 49$ **do**
  Reshape the jth column of `subdata.mat` to $64 * 64 * 64$
  Use FFT to transform the reshaped data to frequency domain
  Sum up frequency data
end for
Take average and normalize the frequency domain data
Find the center frequency

Create the Gaussian filter based on the center frequency
for $j = 1 : 49$ **do**
  Reshape the jth column of `subdata.mat` to $64 * 64 * 64$
  Use FFT to transform the reshaped data to frequency domain

Filter the transformed and shifted data by the Gaussian filter
Use IFFT to inverse the frequency domain data to time-spatial domain
Find the center spatial position
Record the position at time j
end for
Plot the trajectory points
Use IFFT to inverse the frequency domain data to time-spatial domain

Prepare the fitting data
Use polyfit to fit the $x - y$ data points
Predict the trajectory points using the fitted parameters
Plot results

---

4 Computation results

First, we plot the trajectory of the submarine using plot3 in Figure 1.

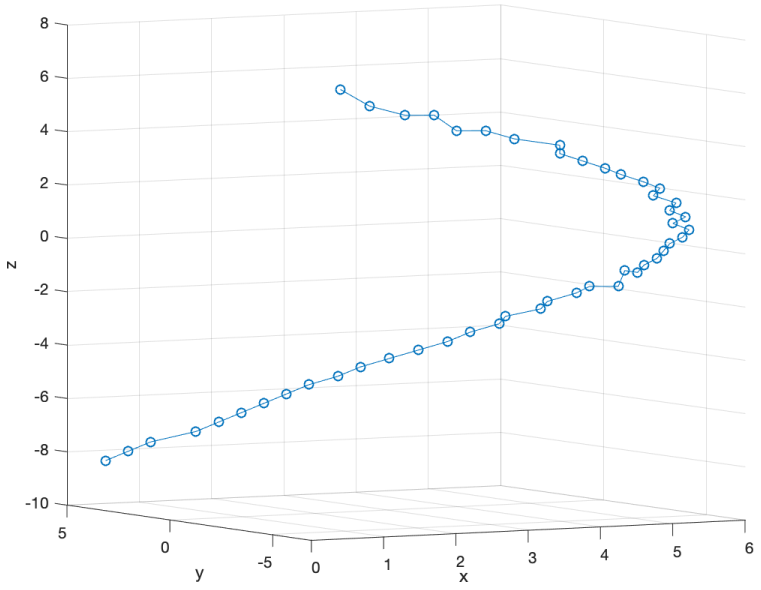

Figure 1: Trajectory of the submarine

Then, we plot the trajectory points of the submarine in the $(x - y)$ plate in Figure 2.

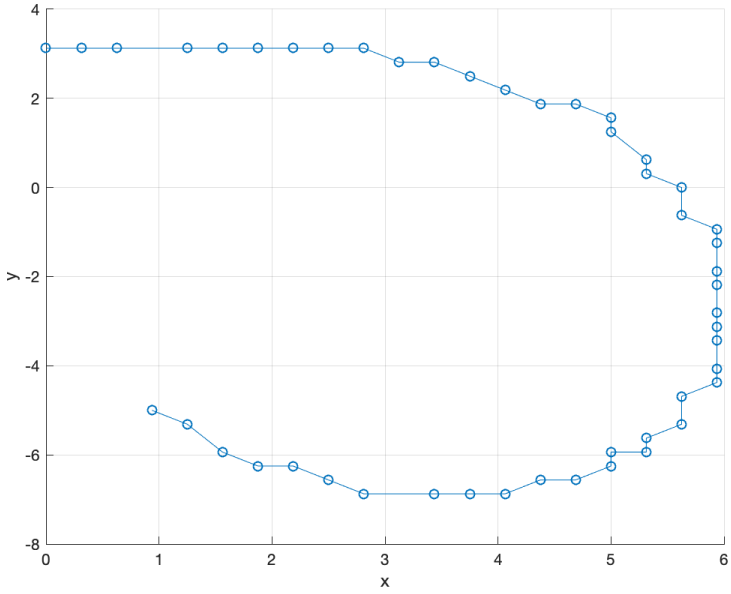

Figure 2: Trajectory of the submarine in x-y plate

Finally, the predicted trajectory is shown in Figure 3

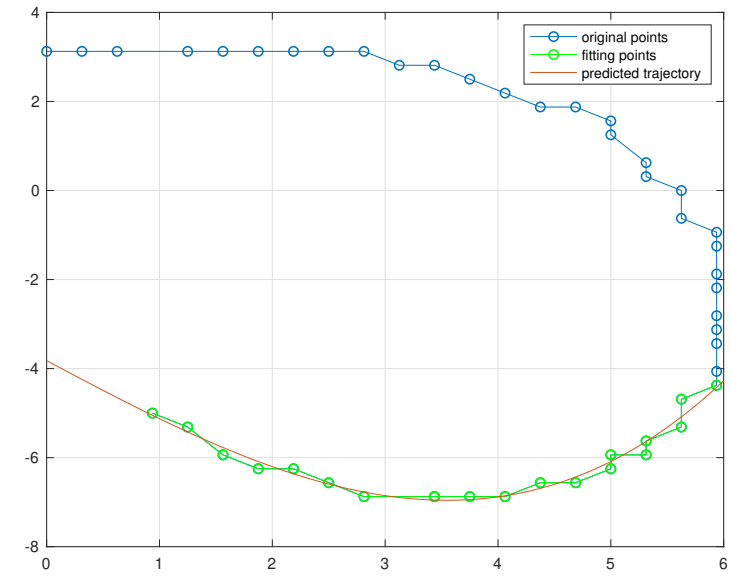

Figure 3: Predicted trajactory of the submarine

The sub-tracking aircraft should be sent to the last point $(0.9375, -5)$ and follow the predicted trajectory to find the submarine.

5 Summary and Conclusions

In this project, we designed an algorithm to process the acoustic data produced by an unknown submarine. We used Fourier transform to capture the data characteristics in frequency domain, and averaged the frequency data over all time slots to reduce the random noise and found the center frequency of the submarine. Based on the center frequency, we designed a Gaussian filter to further denoise the data and find the position of the submarine at each time slot. Together, those positions constitute the trajectory of the submarine. In addition, we used a polynomial fitting method to predict the trajectory of the submarine in a near future by using the trajectory points in recent 10 hours. Now we have sent out the P-8 Orion sub-tracking aircraft. Hope we can hunt the submarine down!

Appendix A GitHub repository

`https://github.com/Guoqq17/UW-AMATH-582`

Appendix B 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.
- `B = reshape(A,sz1,...,szN)` reshapes `A` into a `sz1-by-...-by-szN` array where `sz1,...,szN` indicates the size of each dimension.
- `Y = fftn(X)` returns the multidimensional Fourier transform of an N-D array using a fast Fourier transform algorithm.
- `[row,col] = ind2sub(sz,ind)` returns the arrays row and col containing the equivalent row and column subscripts corresponding to the linear indices ind for a matrix of size `sz`. Here `sz` is a vector with two elements, where `sz(1)` specifies the number of rows and `sz(2)` specifies the number of columns.
- `X = ifftn(Y)` returns the multidimensional discrete inverse Fourier transform of an N-D array using a fast Fourier transform algorithm. The N-D inverse transform is equivalent to computing the 1-D inverse transform along each dimension of `Y`. The output `X` is the same size as `Y`.
- `p = polyfit(x,y,n)` returns the coefficients for a polynomial `p(x)` of degree n that is a best fit (in a least-squares sense) for the data in `y`. The coefficients in `p` are in descending powers, and the length of `p` is `n+1`.
- `y = polyval(p,x)` evaluates the polynomial `p` at each point in `x`.

Appendix C MATLAB Code

Add your MATLAB code here. This section will not be included in your page limit of six pages.

```
%% Set up
% clean workspace
clear all; close all; clc

% load data
load('subdata.mat');

L = 10; % spatial domain
n = 64; % Fourier modes
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);
[Kx,Ky,Kz]=meshgrid(ks,ks,ks);

% visulize the data
for j = 1:49
    Un(:,:,:)=reshape(subdata(:,j),n,n,n);
    M = max(abs(Un),[],'all');
    close all, isosurface(X,Y,Z,abs(Un)/M, 0.7)
    axis([-20 20 -20 20 -20 20]), grid on, drawnow
    pause(1)
end

%% Task 1
close all; clc
ave = zeros(n,n,n);
for j = 1:49
    % reshape
    un(:,:,:)=reshape(subdata(:,j),n,n,n);
    % fft
    utn = fftn(un);
    % sum frequency data
    ave = ave + utn;
end
% average frequency data
ave = abs(fftshift(ave))/49;
% normalize
ave = ave/max(max(max(ave)));

% find the center frequency
[val, ind] = max(ave(:));
[a,b,c] = ind2sub(size(ave),ind);
c_x = Kx(a,b,c);
c_y = Ky(a,b,c);
c_z = Kz(a,b,c);
```

```
%% Task 2
filter = exp(-0.2 * ((Kx - c_x).^2 + (Ky - c_y).^2 + (Kz - c_z).^2));
pos_sub = zeros(3, 49);
for j = 1:49
    un(:,:,:)=reshape(subdata(:,j),n,n,n);
    utn = fftn(un);
    utnf = fftshift(utn) .* filter;
    unf = ifftn(utnf);

    [val, ind] = max(abs(unf(:)));
    [a,b,c] = ind2sub(size(unf),ind);
    pos_sub(1,j) = a;
    pos_sub(2,j) = b;
    pos_sub(3,j) = c;
end

figure(1)
plot3(x(pos_sub(1,:)), y(pos_sub(2,:)), z(pos_sub(3,:)), '-o')
grid on
xlabel('x')
ylabel('y')
zlabel('z')

%% Task 3
x_pre = x(pos_sub(1, 30: 49));
y_pre = y(pos_sub(2, 30: 49));
p_pre = polyfit(x_pre, y_pre, 3);

x_new = linspace(0, 6, 60);
y_new = polyval(p_pre, x_new);
figure(2)
plot(x(pos_sub(1,:)), y(pos_sub(2,:)), '-o')
hold on
plot(x_pre, y_pre, 'g-o')
plot(x_new, y_new)
legend('original points', 'fitting points', 'predicted trajectory')
grid on
```