AMATH 482 Homework 1

Sandy Yang

January 24, 2020

Abstract

In this problem set, we explore one use case where signal averaging and filtering can be used to extract information from a data signal with noise. The Fourier transforms are used to convert a signal from the time domain to the frequency domain, where it is averaged over time filtered with a Gaussian function to get rid of noises. Targeting the predominant remaining frequency, we are able to track the submarine's path through a noisy field, determining its location at a given point in time.

1 Introduction and Overview

Suppose we are hunting for a submarine in the Puget Sound with a new submarine technology that emits an unknown acoustic frequency. The data is obtained over a 24-hour period in half-hour increments with the use of a broad spectrum recording of acoustics. Unfortunately, the submarine is moving, so its location and path need to be determined.

Since we are tracking the submarine, we know that the noise is generated from the random movements of fluid and we can assume their corresponding frequencies in the acoustics data will also be random. Therefore, by averaging the frequency-domain data, all random frequencies should be canceled out, leaving only the frequencies generated by the submarine. An filter can then be applied to the data, centered around this target frequency, to further isolate the desired signal. Then, the frequency data can be converted back into the time domain to get a much clearer view of the spatial movements of just the submarine.

2 Theoretical Background

We will rely on two technique in order to track the submarine through the noisy data : the Fourier Transform and Gaussian filtering.

2.1 The Fourier Transform

The Fourier Transform is used to turn a time signal into its frequency components. Given a function f(x), it integrates the function multiplied by Euler's formula over an infinite domain, producing a function of frequency F(k).

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

On the other hand, the Inverse Fourier Transform converts signals from the frequency domain back into the time domain:

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

Since we can move between the time and frequency domains, we can do the analysis in whatever space is most convenient, then return to the other space to get our results. Computationally, we perform Fourier Transforms using the Fast Fourier Transform (FFT) algorithm. These operations assume certain conditions exist. One is that the it operates over a discrete interval of -2π to 2π, meaning we must scale data from its original domain to fit this. Periodic boundary conditions are also assumed, and so we must set up our boundaries as such, even if our problem is not periodic. Another assumption is that the signal is described in $n = 2^j$ points. This is due to the recursive nature of FFT which continually divides the signal in half.

2.2 Gaussian Filter

One of the methods to remove noise from a signal data is to remove data not near a desired frequency. By multiplying the frequency-domain signal by some other function that is near zero for all values except our desired frequency, we can attenuate our signal, leaving only relevant data. One of the common functions that is used to demolish noises is the Gaussian function:

$$e^{-\tau(K-k_0)^2} \tag{3}$$

A Gaussian is centered around a given frequency k0 where it has a normalized value of 1, and the other value decays to 0 when moving away from the center. The rate of attenuation is controlled by variable $-\tau$. The simplicity and smoothness of the Gaussian function is why it's used to filter frequency domain signals.

3 Algorithm Implementation and Development

The method tracking the location of the submarine is broken down into consecutive steps, described below. This particular implementation is done in MATLAB.

3.1 Setup

Our problem is defined over the x, y, z intervals -10 to 10 in the time domain, which we divide up into a 3D grid 64 points across along every axis, allowing us to use FFT.

```
L = 10; % spatial domain
n = 64; % Fourier modes
x2 = linspace(-L,L,n+1); x = x2(1:n); y =x; z = x;
```

We then transform our time domain interval into the frequency domain interval by scaling by $\frac{2\pi}{2L}$, giving us the needed interval of -2π to 2π.

```
k = (2*pi/(2*L))*[0:(n/2 - 1) -n/2:-1];
ks = fftshift(k);
```

Then, we used meshgrid to build up the matrix.

```
[X,Y,Z]=meshgrid(x,y,z);
[Kx,Ky,Kz]=meshgrid(ks,ks,ks);
```

3.2 Average the Spectrum

After setting up the axis, we want to work on signals. To determine the center frequency for the filter, we must first average through the signals to remove the noise. By transforming the data into the frequency domain with fftn for each time step and each dimension, we get a all the frequencies of our signal across time. Since some of these frequencies is random, averaging the spectrum over time can make all the random data less intense, leaving only the frequencies corresponding to the submarine's path. We can also normalize our data get everything on a scale of 0 to 1.

```
Uave = zeros(n,n,n);
for j=1:49
    Utn(:,:,:) = fftn(reshape(subdata(:,j),n,n,n));
    Uave = Uave + Utn;
end
Uave = fftshift(Uave)./49;
```

3.3 Determine the Center Frequency

With the averaged spectral frequencies, we then have to decide where the target frequency is. The frequencies most associated with the submarine's location will make up the largest portion of the time-averaged signal, so we can simply find the maximum value and its indices.

```
[val,idx] = max(Uave(:));
[a,b,c] = ind2sub(size(Uave),idx);

x0 = Kx(a,b,c);
y0 = Ky(a,b,c);
z0 = Kz(a,b,c);

sprintf('x0: %s, y0: %d, z0: %f',x0, y0, z0) % print out result
'x0: 5.340708e+00, y0: -6.911504e+00, z0: 2.199115'
```

The central frequency is located at:

$$(x, y, z) = (5.340708, -6.911504, 2.199115) \tag{4}$$

3.4 Filter the Data

Now, with the position of the target frequency in Fourier space, we can construct a 3D Gaussian filter to center around that frequency.

```
tau = -.5;
filter=exp(tau*(((Kx-x0).^2)+((Ky-y0).^2)+((Kz-z0).^2)));
```

By multiplying this function by the Fourier-transformed data at every point in time, we can then filter out all the noise generated from random variation. The important thing to note is that we must use fftshift to un-swap the first and second halve of the domains for our data, as the Gaussian filter is constructed as if there was no swapping. Then, we can convert the filtered data back to the time domain using Inverse Fast Fourier transform (ifftn). And finally, with the newly de-noised spatial data, we can track the movement of the submarine by looking for the location of the highest intensity signal at each time step, and saving them to a list of coordinates.

```
A = [];
B = [];
C = [];

for j = 1:49
    Un = fftn(reshape(subdata(:,j),n,n,n));
    Unt = fftshift(Un);
    Unft = filter.*Unt;
    Unf = ifftn(Unft);

    [val,idx] = max(Unf(:));
    [b,a,c] = ind2sub(size(Unf),idx);
    A(j) = a;
    B(j) = b;
    C(j) = c;
end
```

4 Computational Results

By using Fourier Transforms, Gaussian filtering, and time-averaging to extract relevant data from a noisy data set, we were able to determine coordinate of the submarine and plot the trajectory out.

```
plot3(x(A),y(B),z(C), 'Linewidth', 2)

xlabel('x')
ylabel('y')
zlabel('z')
title('Submarine Movement Trajectory')
x49 = x(A(end));
y49 = y(B(end));
z49 = z(C(end));

hold on
plot3(x49, y49, z49, 'r*')
```

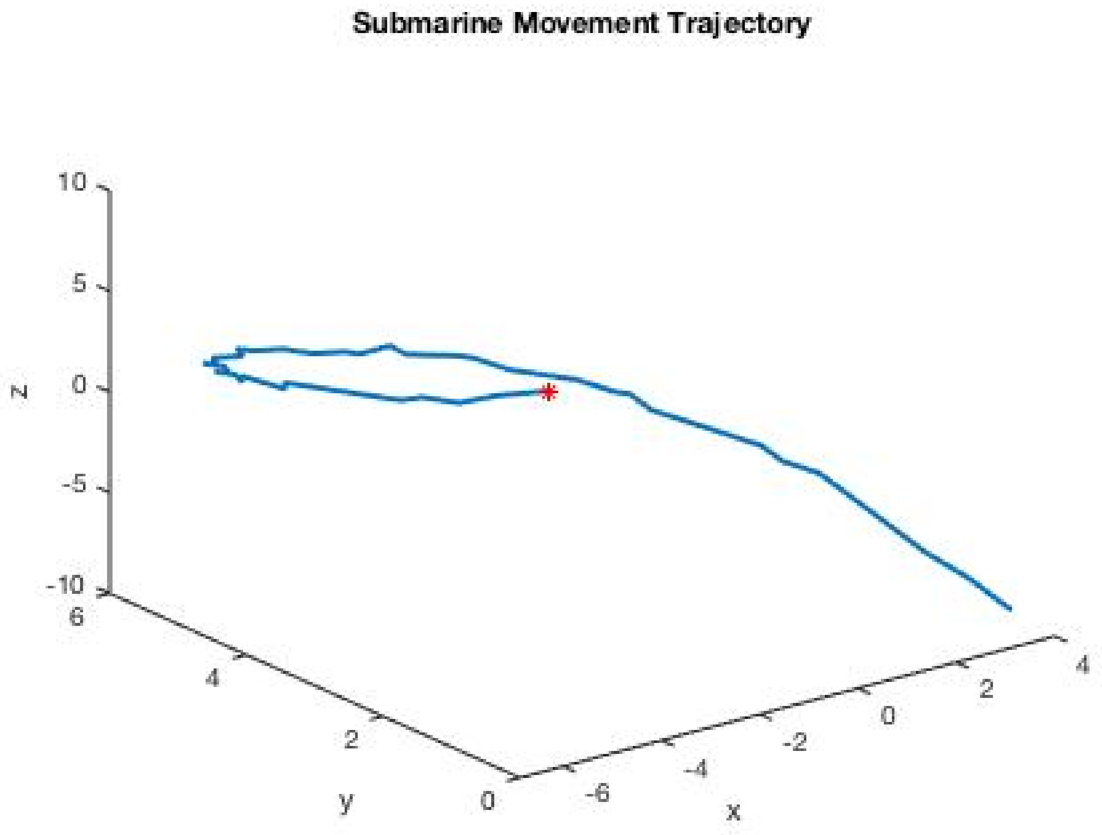

Figure 1: This graph show the submarine location through out time

5 Summary and Conclusions

In conclusion, we should send our P-8 Poseidon subtracking aircraft to the following coordinates:

Table 1: x-y Coordinate of the Submarine.

x-y coordinate	
x	y
3.125	0
3.125	0.3125
3.125	0.625
3.125	1.25
3.125	1.5625
3.125	1.875
3.125	2.1875
3.125	2.5
3.125	2.8125
2.8125	3.125
2.8125	3.4375
2.5	3.75
2.1875	4.0625
1.875	4.375
1.875	4.6875
1.5625	4.6875
1.25	5
0.625	5.3125
0.3125	5.3125
0	5.625
-0.3125	5.625
-0.9375	5.9375
-1.25	5.9375
-1.875	5.9375
-2.1875	5.9375
-2.8125	5.9375
-3.125	5.9375
-3.4375	5.9375
-4.0625	5.9375
-4.375	5.9375
-4.6875	5.625
-5.3125	5.625
-5.625	5.3125
-5.9375	5.3125
-5.9375	5
-6.25	4.6875
-6.5625	4.6875
-6.5625	4.375
-6.875	4.0625
-6.875	4.0625
-6.875	3.4375
-6.875	3.4375
-6.875	3.125
-6.5625	2.5
-6.25	2.1875
-6.25	1.875
-5.9375	1.5625
-5.625	1.25

Continuation of Table 1	
Something	something else
-5	0.9375
End of Table	

6 Appendix A - Function Descriptions

- `fftn` Discrete Fast Fourier transform in n-dimensions. We need to use this since our data is in 3 dimension. If use only fft, the transformation will only be applied to x-coordinate. Same thing for ifftn.
- `ifftn` Inverse Discrete Fast Fourier transform in n-dimensions
- `fftshift` Swap first and second halves of domain, as is done in FFT.