AMATH 482 Homework 1

Sophia Jannetty

January 27, 2021

Abstract

There is a Submarine in Puget Sound that I must locate using a series of broad spectrum acoustics recordings taken every half hour over a 24 hour period. Here, I use the Discrete Fourier Transform as a tool to analyze and process these data. I average the data in the frequency domain to find the central frequencies the submarine is emitting in each spacial dimension and then construct and apply a 3D Gaussian filter to eliminate irrelevant frequencies from the data. Plotting the filtered data reveals the path of the submarine. This document outlines the theoretical basis for this signal processing and provides details pertaining to my use of MATLAB's Fast Fourier Transform implementation to solve this submarine tracking problem. My intended audience is my graders and my future self.

1 Introduction and Overview

The Fourier Transform is a coordinate transform that takes a function and quantifies the magnitudes of its constituent frequencies. This is a useful tool in signal processing. In this exercise, I have artificial data representing noisy broad spectrum acoustics recordings. I am told that the noise in the data is white, which means its average magnitude over all instances of data collection goes towards zero as the number of data collection instances increases. I average the magnitudes of frequencies present in my data and take the maximum frequency in each dimension to be the central frequency emitted by the submarine in each dimension. I then multiply my Fourier transformed data by a 3D Gaussian filter that is centered at the central frequency in each dimension. This minimizes the magnitudes of frequencies in my data that I know are not relevant to my submarine tracking task while maximizing the magnitudes of the frequencies the submarine is emitting. I then inverse Fourier Transform my data and record the location of the maximum amplitude signal in the space domain at each time point. I take this point to be the location of the submarine.

2 Theoretical Background

2.1 Fourier Analysis

2.1.1 The Fourier Series

The Fourier series is an infinite sum of cosines and sines of increasing frequency[1]. Any 2-π periodic piecewise smooth function $f(x)$ can be written as:

$$f(x) = \frac{a_0}{2} + \sum_{k=1}^{\infty} (a_k \cos kx + b_k \sin kx) \quad x \in (-\pi, \pi]. \tag{1}$$

The value of k (termed the wave number) gives the frequency of the sin or cosine waves present in the function. The coefficients a_k and b_k give the magnitude of each frequency present in the function. The constant $\frac{a_0}{2}$ shifts the series to the correct y position. The coefficients a_k and b_k can also be thought of as the coordinates obtained from projecting the function onto the cosine and sine basis $\{\cos kx, \sin kx\}_{x=0}^{\infty}$[1]. This means that each coefficient is the inner product of $f(x)$ and either $\sin kx$ or $\cos kx$ as seen in Equations 2 and 3:

$$a_k = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \cos(kx) dx \tag{2}$$

$$b_k = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \sin(kx) dx \tag{3}$$

By using Euler's formula $e^{ikx} = \cos(kx) + i\sin(kx)$ to concisely represent the sines and cosines and by scaling the 2-π domain to a $2L$ domain, the formula for the Fourier series of a function with a $2L$ periodic domain is given by formula 4 with corresponding Fourier coefficients given by formula 5:

$$.f(x) = \sum_{-\infty}^{\infty} \left(c_n e^{in\pi x/L} \right) \quad x \in [-L, L]. \tag{4}$$

$$c_n = \frac{1}{2L} \int_{-L}^{L} f(x) e^{-in\pi x/L} dx \tag{5}$$

Notably, though the function $f(x)$ is real, the coefficients will be complex numbers with the real part of the number corresponding to the even components of the function and the imaginary part of the number corresponding to the odd components of the function[2].

2.1.2 The Fourier Transform

The Fourier transform takes a function of time or space that spans the entire line $x \in [-\infty, \infty]$ and converts it into a function of frequency. It is an integral transform that essentially finds the limit of a Fourier series as the domain length $2L$ goes to infinity[1]. This allows for a Fourier series-like representation of continuous functions with an infinite domain. The formula to convert a function from the time/space domain into the spectral domain is given by formula 6 and the inverse formula to take a function from the spectral domain back into the time/space domain is given by formula 7:

$$.\hat{f}(k) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(x) e^{-1kx} dx \tag{6}$$

$$f(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \hat{f}(k) e^{ikx} dk \tag{7}$$

2.1.3 The Discrete and Fast Fourier Transform

The Discrete Fourier Transform (DFT) takes a vector of N data points (which can be obtained by sampling a function at equally-spaced intervals or can be amplitude measurements of signals at discrete points as in the case of this assignment) and returns a vector given by formula 8

$$\hat{x}_k = \frac{1}{N} \sum_{n=0}^{N-1} x_n e^{\frac{2\pi ikn}{N}} \tag{8}$$

This tests for the presence of the specified set of frequencies given by $k = 0, 1, 2, ..., N-1$. By capping the maximum wave number at $N-1$, the discrete Fourier transform is taking into account the fact that it is unable to detect frequencies that are so high that oscillations would happen between our sampled points.

The Fast Fourier transform (FFT) is a Discrete Fourier Transform algorithm that runs in $O(n\log(n))$ operations instead of $O(n^2)$ operations [2]. The FFT requires that the number of wavenumbers must be a power of two, but the MATLAB implementation will pad the number of wavenumbers you provide to the nearest power of two. When applying FFT to a multi-dimensional dataset, the algorithm FFTs in each dimension and produces one Fourier coefficient for each possible combination of wavenumbers across dimensions.

2.2 Averaging Data to Eliminate White Noise and Identify Central Frequencies

White noise was added to the submarine data at each data collection time point. Because this noise is drawn from a Gaussian distribution with a mean of 0, the average value of noise for each Fourier coefficient over all of the time points goes towards zero as the number of data collection instances increases [2]. By averaging the results of the discrete Fourier transform of the signal at each time point, the coefficients of wavenumbers that are present due to noise get smaller while the coefficients of frequencies that are emitted from the submarine do not. Therefore I can find the central frequencies emitted by the submarine in each dimension by averaging all of the Fourier transformed data at each time point and finding the frequencies with the largest magnitude in each dimension.

2.3 Filtering

2.3.1 Why and How to Filter

We filter data when there are frequencies present that are irrelevant to our frequencies of interest. Once I use averaging to identify the central frequencies of the submarine, I want to remove all frequencies that are not at or near the central frequencies. I consider values near the central frequencies because I do not know that I definitely found the exact central frequencies by averaging. However I assume the values I found were close.

Filtering must be done in the frequency domain. The convolution theorem states that multiplication in the frequency domain results in convolution in the space domain [2]. Some function must be defined that maximizes frequencies at and around the central frequencies while minimizing or deleting all other frequencies. The Fourier transformed data at each time point is multiplied by this function and then inverse transformed back to the space domain. When back in the space domain, all data that was not at the central frequencies will be removed, leaving only data relevant to the submarine

2.3.2 Gaussian Filter Design

I constructed a Gaussian function in each dimension shifted so the mean is at the central frequency of each dimension. When defining the Gaussian one must specify the mean and the variance. The specified variance will determine the width of the filter. A larger variance means a wider filter and more frequencies around the central frequencies will be left maximized. A smaller variance means a smaller filter and more frequencies will be filtered out.

3 Algorithm Implementation and Development

3.1 Setup and Assumptions

For corresponding code, please refer to Appendix B lines 1-58. I assume the signal emitted by submarine is static and is emanating from a single point. Raw data are imported into MATLAB. The starter code and dimensions of the incoming dataset gave the spatial resolution and spectral resolution of the submarine tracking equipment. All values are unitless but the spatial domain is 10 (which was given in the starter code) and the spectral resolution (or number of Fourier modes) is 64 (which was given in the starter code and corresponds to the length of each dimension's axis in my data at each time point). With this information, I create a linearly spaced vector of 65 points spanning from -10 to 10. I then eliminate the last point because the fast Fourier transform assumes the function underlying my data is periodic so the first point would be the same as the last. This leaves me a vector v of 64 linearly distributed points spanning the spatial domain of my data. I use this vector to create three 3D matrices, one for each spacial dimension represented in my data. The dimensions of each of these matrices is a v by v by v. I also construct a corresponding coordinate system that allows me to plot my data in the spacial domain.

Next I define my frequency components. I define a vector k of wavenumbers (or frequencies). I scale my domain down to 2-pi and define my wavenumbers to count from 0 to 31 and then from -32 to -1 on order to account for the assumed 2-pi domain of MATLAB's FFT function and to mimic the order in which data is returned from MATLAB's FFT function. I then create three 3D matrices, one for each spacial dimension represented in my data. The dimensions of each of these matrices is k by k by k and each cell stores a different wavenumber. When I run MATLAB's FFT shift function, it will return a k by k by k matrix of Fourier coefficients. The wavenumber each of those coefficients corresponds to is stored in the analogous location in my frequency component 3-D matrices. The corresponding wavenumber in the x dimension will be at the corresponding location in the x wavenumber matrix, the corresponding wavenumber in the y dimension will be at the corresponding location in the y wavenumber matrix, and the corresponding wavenumber in the z dimension will be at the corresponding location in the z wavenumber matrix. I also construct a corresponding coordinate system that allows me to plot my data in the frequency domain.

As a first step I plotted the unfiltered data (Figure 1.A) and saw extraneous signals in my plot unrelated to my submarine. I also plotted the maximum amplitude point at each time point in the unfiltered data (Figure 1.B). The path of the submarine is visible, but is not as smooth as expected implying the presence of noise.

3.2 Averaging to Find the Central Frequencies

For corresponding code, please refer to Appendix B lines 59-89. I fast Fourier transformed my data at each time point and added it to a 64x64x64 matrix, obtaining a sum of the Fourier coefficients for each possible wavenumber in each dimension at each time point. I then divide each sum by 49 (the number of data collection instances) to get an average coefficient for each wavenumber in each dimension. I then find the maximum of the absolute value of each coefficient. I want to find the coefficient with the maximum magnitude regardless of the sign of the imaginary component. Comparing the absolute values of the coefficients ensures I am selecting the coefficient with the largest magnitude regardless of the sign of the imaginary part of the number. The location of the maximum coefficient in the output matrix from MATLAB's FFT function corresponds to the location of the central wavenumber in my frequency component 3-D matrices for each dimension. I find the central frequencies for each dimension by going to the corresponding location in each of my three wavenumber matrices (see Appendix B lines 79-86).

3.3 Filter Construction and Application

For corresponding code, please refer to Appendix B lines 90-152. I create a 3D Gaussian in the frequency domain using the wavenumber matrices (see Appendix B lines 94-95). In each dimension the Gaussian is shifted so that it is centered on each respective dimension's central frequency. I experimented with filter widths (variance values) and selected a width by visually assessing the smoothness of the submarine path that resulted from each filter application (data not shown). I selected a filter width of .5 because it yielded a satisfactorily smooth path.

To apply the filter, I fast Fourier transformed my data at each time point and multiplied the resulting matrix by my 3d Gaussian filter. I then inverse Fourier transformed the data back into the spacial domain (see Appendix B lines 115-130). I recorded the location in the matrix of the maximum amplitude as the location of the submarine at that time point.

4 Computational Results

The center frequencies determined from averaging were 5.340707511102648 in the X dimension, -6.911503837897545 in the Y dimension, and 2.199114857512855 in the Z dimension. The location of the submarine at each time point found post filtering can be seen in Figure 1. The exact coordinates of each location at each time point can be seen in Table 1. If directing a plane to the submarine's last known location, I would send the plane to to location corresponding to the -5.0000 X coordinate and the 0.9375 Y coordinate.

Time	00:00	00:30	01:00	01:30	02:00	02:30	03:00	03:30	04:00	04:30
X	3.1250	3.1250	3.1250	3.1250	3.1250	3.1250	3.1250	3.1250	3.1250	2.8125
Y	0.0000	0.3125	0.6250	1.2500	1.5625	1.8750	2.1875	2.5000	2.8125	3.1250
Z	-8.1250	-7.8125	-7.5000	-7.1875	-6.8750	-6.5625	-6.2500	-5.9375	-5.6250	-5.3125
Time	05:00	05:30	06:00	06:30	07:00	07:30	08:00	08:30	09:00	09:30
X	2.8125	2.5000	2.1875	1.8750	1.8750	1.5625	1.2500	0.6250	0.3125	0.0000
Y	3.4375	3.7500	4.0625	4.3750	4.6875	4.6875	5.0000	5.3125	5.3125	5.6250
Z	-5.0000	-4.6875	-4.3750	-4.0625	-3.7500	-3.4375	-3.1250	-2.8125	-2.5000	-2.1875
Time	10:00	10:30	11:00	11:30	12:00	12:30	13:00	13:30	14:00	14:30
X	-0.3125	-0.9375	-1.2500	-1.8750	-2.1875	-2.8125	-3.1250	-3.4375	-4.0625	-4.3750
Y	5.6250	5.9375	5.9375	5.9375	5.9375	5.9375	5.9375	5.9375	5.9375	5.9375
Z	-1.8750	-1.8750	-1.2500	-1.2500	-0.9375	-0.6250	-0.3125	0.0000	0.3125	0.6250
Time	15:00	15:30	16:00	16:30	17:00	17:30	18:00	18:30	19:00	19:30
X	-4.6875	-5.3125	-5.6250	-5.9375	-5.9375	-6.2500	-6.5625	-6.5625	-6.8750	-6.8750
Y	5.6250	5.6250	5.3125	5.3125	5.0000	4.6875	4.6875	4.3750	4.0625	4.0625
Z	0.9375	1.2500	1.5625	1.8750	2.1875	2.5000	2.8125	3.1250	3.4375	3.7500
Time	20:00	20:30	21:00	21:30	22:00	22:30	23:00	23:30	00:00	
X	-6.8750	-6.8750	-6.8750	-6.5625	-6.2500	-6.2500	-5.9375	-5.6250	-5.0000	
Y	3.4375	3.4375	3.1250	2.5000	2.1875	1.8750	1.5625	1.2500	0.9375	
Z	4.0625	4.3750	4.6875	5.0000	5.0000	5.6250	5.6250	6.2500	6.5625	

Table 1: Submarine X, Y, and Z Coordinates at each Data Collection Time Point

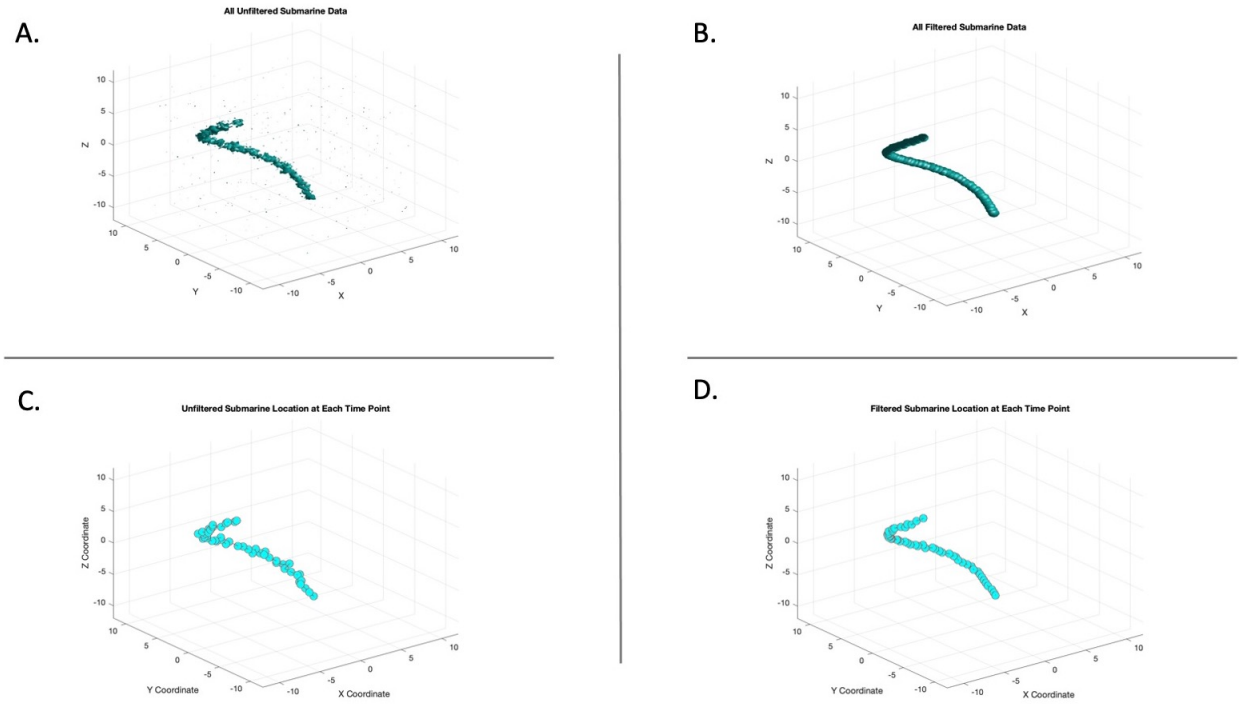

Figure 1: Path of the Submarine Before (A., C.,) and After (B., D.,) filtering. Time 0 is the point with the lowest Z value. A. and B. plot all data before and after filtering. C. and D. plot the maximum magnitude

5 Summary and Conclusions

Here I use the Discrete Fourier Transform for signal analysis and processing. I differentiate central frequencies from white noise in noisy broad spectrum acoustics data by averaging the data in the frequency domain. I then eliminate frequencies that resulted from the presence of white noise in my data by applying a Gaussian filter in the frequency domain, yielding a data set depicting a smooth submarine path.

My entire analysis depends on the assumptions that this submarine has constant central frequencies in each dimension, that the noise in my data is white, and that my environment around the submarine remains unchanged. If the central frequencies changed over time I am not sure I would have been able to detect them from the noise when averaging. If I somehow knew the submarine's central frequencies at each time point I would have to move my Gaussian to be centered at the correct central frequencies for each time point. If my noise was not white or if the environment around my submarine changed over time (for example if another submarine was passing through emitting different central frequencies), I would not be able to detect the central frequencies of my submarine averaging.

The Fourier transform is a powerful tool for signal analysis, but it has limitations. The Fourier transform quantifies the magnitudes of a function's constituent frequencies over the entire time frame of the function. It can not tell you which frequencies happened when. This was not a problem for this artificial circumstance because I analyzed data from discrete moments in time and considered the function underlying my data at each time point to be distinct. Therefore I could track whether a frequency was present at each time point by looking for it in each time point's Fourier transform.

References

[1] Steven L. Brunton and Jose Nathan Kutz. *Data-driven science and Engineering: machine learning, dynamical systems, and control*. Cambridge University Press, 2019.

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

Appendix A MATLAB Functions

Table 2: Notable Matlab Functions

`y = linspace(x1,x2,n)`	returns a row vector of `n` evenly spaced points between `x1` and `x2`
`[X,Y,Z] = meshgrid(x,y,z)`	returns 3-D grid coordinates defined by vectors `x`, `y`, and `z`. The grid is represented by the coordinates `X`, `Y`, and `Z` and has size `length(y)` by `length(x)` by `length(z)`. Used for constructing my 3D coordinate systems for plotting data in the time domain and frequency domain, and for creating wavenumber matrices that allow me to find the wavenumbers that correspond to the coefficients returned by the `fftn` function.
`ks = fftshift(k)`	used to reorder wavenumbers to match output structure of `fftx()`, which returns wavenumbers from 0 to $\frac{n}{2}-1$, and then $\frac{n}{2}$ to -1. Allows for wavenumber axis to be continuous from $\frac{-n}{2}$ to $\frac{n}{2}$ while plotting.
`Un(:,:,:) = reshape(subdata(:,j),n,n,n)`	reshapes my data into `n` by `n` by `n` matrix `Un`. Used in `for` loop that iterates through each time point
`[Mn, idx] = max(abs(Un(:)))`	returns max point `Mn` in matrix `Un` and the index `idx` of its location in `Un`.
`[yind, xind, zind] = ind2sub(size(Un), idx)`	returns the `y`, `x`, and `z` coordinates of the cell at index `idx` in matrix `Un`.
`Utn(:,:,:) = fftn(Un)`	MATLAB's FFT implementation for 3+ dimensional data. Returns matrix `Utn` of Fourier coefficients that describe the function underlying the data in matrix `Un`. Wavenumbers are ordered as described in `ks = fftshift(k)` above. Each coefficient's wavenumber can be found in at that coefficient's index in the wavenumber matrix seen in Appendix B lines 83-86
`unf = ifftn(Utn)`	MATLAB's inverse FFT implementation for 3+ dimensional data. Takes the matrix `Utn` of Fourier coefficients and returns matrix `unf` of data returned to the spacial domain.
`isosurface(X,Y,Z,Un,isovalue)`	plots and connects all points equal to `isovalue` in Un matrix on 3D surface with dimensions specified by `X, Y, Z`

Appendix B MATLAB Code

```
%% Preparation and setup
%clean workspace
clear all; close all; clc
load subdata.mat %Imports data as 262144x49 (space by time) matrix

L = 10; %spatial domain (spatial resolution)
n = 64; %Fourier modes (spectral resolution)
x2 = linspace(-L,L,n+1); %x is from -10 to 10, generates lineraly spaced
% vector of n+1 points from -10 to 10
x = x2(1:n); %periodic boundary means first point is same as last
y =x; z = x;

%scale my data's domain to 2 pi periodic, define fourier wave numbers
k = (2*pi/(2*L))*[0:(n/2 - 1) -n/2:-1];
ks = fftshift(k); %used for plotting in frequency domain

[X, Y, Z] = meshgrid(x,y,z); %time domain meshgrid, 3 3dimensional vectors
[Kx, Ky, Kz] = meshgrid(ks,ks,ks); %frequency domain meshgrid fftshifted
[UKx, UKy, UKz] = meshgrid(k,k,k); %frequency domain meshgrid not shifted

%% Initial Data Investigation
%Create vectors for storing indicies of max magnitude point at each time
unfiltered_xvec = zeros(49,1);
unfiltered_yvec = zeros(49,1);
unfiltered_zvec = zeros(49,1);
for j=1:49 %for data at each time point (slice)
    %takes data from time point, reshape into 64x64x64 matrix
    Un(:,:,:) = reshape(subdata(:,j),n,n,n);

    %find point with max magnitude at this time point
    [Mn, idx]=max(abs(Un(:)));
    %Get coordinates of this spatial domain max point
    % Matlab is column major, so int2sub returns y then x then z
    [yind, xind, zind] = ind2sub(size(Un), idx);
    unfiltered_xvec(j) = X(xind, xind, xind);
    unfiltered_yvec(j) = Y(yind, yind, yind);
    unfiltered_zvec(j) = Z(zind, zind, zind);

    %Plot all Unfiltered Data
    figure(1)
    isosurface(X,Y,Z,abs(Un)/Mn,0.7)
    title('All Unfiltered Submarine Data')
    xlabel('X')
    ylabel('Y')
    zlabel('Z')
    axis([-12 12 -12 12 -12 12]), grid on, drawnow
end

%Plot Max Point at each time point
figure(2)
plot3(unfiltered_xvec(),unfiltered_yvec(),unfiltered_zvec(), '-o',...
    'Color','r','MarkerSize',10,'MarkerFaceColor','cyan')
```

```
title('Unfiltered Submarine Location at Each Time Point')
xlabel('X Coordinate')
ylabel('Y Coordinate')
zlabel('Z Coordinate')
axis([-12 12 -12 12 -12 12]), grid on, drawnow

%% Averaging to find central frequencies in each dimension
Uave = zeros(64, 64, 64); %matrix for summing coefficients at each time
for j=1:49
    Un(:,:,:) = reshape(subdata(:,j),n,n,n);
    Utn(:,:,:) = fftn(Un);% noisy data in frquency domain
    Uave = Uave + Utn; % add frequencies for each time point
end
Uave = Uave/49; % find average coefficient of each wavenumber
Mave=max(abs(Uave),[],'all'); %max coefficient

%Plot Max Fourier Coefficient Point- Coordinates of this point are central
% frequencies in each dimension
figure(3)
%scale to max value so all magnitures between 0 and 1
isosurface(Kx,Ky,Kz, fftshift(abs(Uave))/Mave, .9)
title('Maximum Amplitude Point in Frequency Domain')
xlabel('X')
ylabel('Y')
zlabel('Z')

% maxvalue and its index for the averaged signal over all time slices
[mxv, idx] = max((abs(Uave(:))));
% Retrieve indeces for each cartesian dimension; yields index in each
%dimension's 3D k array corresponding to primary K number in that dimension
[firstind, secondind, thirdind] = ind2sub(size(Uave), idx);
xfreq = UKx(firstind, secondind, thirdind);
yfreq = UKy(firstind, secondind, thirdind);
zfreq = UKz(firstind, secondind, thirdind);



%% Make Filter
%a = width of filter, 1/a is variance of gausian filter:
% smaller a = wider, bigger = narrower,
a = .5;
filter = (exp(-a* ((UKx-xfreq).^2))).* (exp(-a* ((UKy-yfreq).^2))).* ...
    (exp(-a* ((UKz-zfreq).^2)));

%Plot filter in frequency domain to make sure it is in the correct place
plotfilter = fftshift(filter);
figure(4)
isosurface(Kx,Ky,Kz,abs(plotfilter),.7)
title('Gaussian Filter Size and Location')
xlabel('X')
ylabel('Y')
zlabel('Z')


```

```
%% Apply filter

%Create vectors for storing indicies of max magnitude point at each time
%in spacial domain post filtering
xvec = zeros(49,1);
yvec = zeros(49,1);
zvec = zeros(49,1);

for j=1:49 %for data at each time point (slice)
    Un(:,:,:) = reshape(subdata(:,j),n,n,n);

    Utn(:,:,:) = fftn(Un);%noisy data in frquency domain
    clean = Utn.*filter; %apply filter to data at this time point

    unf = ifftn(clean); %Move clean data back into time domain

    %find point with max magniture in filtered time domain at this time point
    [Mn, idx]=max(abs(unf(:)));
    % Get coordinates of this spatial domain max point (coordinates of sub)
    % Matlab is column major, so int2sub returns y then x then z
    [yind, xind, zind] = ind2sub(size(Un), idx);
    xvec(j) = X(xind, xind, xind);
    yvec(j) = Y(yind, yind, yind);
    zvec(j) = Z(zind, zind, zind);

    %Plot of all Filtered Submarine Data in Space Domain
    figure(5)
    isosurface(X,Y,Z,abs(unf)/Mn,0.9)
    title('All Filtered Submarine Data')
    xlabel('X')
    ylabel('Y')
    zlabel('Z')
    axis([-12 12 -12 12 -12 12]), grid on, drawnow
end

%% Plot Submarine Course
%Plots of max value at each time point
figure(6)
plot3(xvec(),yvec(),zvec(), '-o','Color','r','MarkerSize',10,...
    'MarkerFaceColor','cyan')
title('Filtered Submarine Location at Each Time Point')
xlabel('X Coordinate')
ylabel('Y Coordinate')
zlabel('Z Coordinate')
axis([-12 12 -12 12 -12 12]), grid on, drawnow

figure(7)
plot(xvec(),yvec(),'-o','Color','r','MarkerSize',10,...
    'MarkerFaceColor','cyan')
title('X and Y Submarine Location at Each Time Point')
xlabel('X Coordinate')
ylabel('Y Coordinate')
axis([-12 12 -12 12 -12 12]), grid on, drawnow

```

```
%Create character array of time labels
start = fix(now);
increment = 30/(60*24);% 30-Minute Increments
time_vct = 0:increment:(1);% Vector
time_str = datestr(start + time_vct, 'HH:MM:SS');% Time Strings
time_label_array = [time_str(1:49,:)];
time_label_array;

figure(8)
plot(xvec(),zvec(),'-o','Color','r','MarkerSize',10,'MarkerFaceColor',...
    'cyan')
title('X and Z Submarine Location at Each Time Point')
xlabel('X Coordinate')
ylabel('Z Coordinate')
%Apply time labels to each data point
%dx = 0.5; dz = 0.1; % displacement so the text isn't over data points
%text(xvec()+dx, zvec()+dz, c);
axis([-12 12 -12 12 -12 12]), grid on, drawnow
```