Amath 482 Homework 1

FOURIER TRANSFORMS FOR FILTERING NOISY DATA

MICHAEL GABALIS

Abstract: This paper aims to demonstrate the usefulness of the Fourier Transform and averaging techniques to filter through noisy data, specifically highly noisy data. As a case study, we are using a submarine that emits an unknown acoustic frequency that we need to detect. We will attempt to locate the submarine from its acoustic signature using the Fourier Transform.

University of Washington

January 2021

1 Introduction and Overview

When dealing with large amounts of data, whether that be from a radar or a submarine emitting an acoustic frequency, there can be a significant noise component. Figuring out what is causing this noise is needed to figure out appropriate strategies for filtering it out. If we believe the noise is zero-mean in the frequency domain, the techniques used in this paper are highly effective in filtering out the noise. The data used in this submarine case study is a collection of 49 acoustic signatures of a submarine captured at consecutive points in time. Each data point captures the spatial fluctuations in a 64 by 64 by 64 grids at that point in time, while also being complex to capture both the phase and amplitude of the fluctuations. Using the Fourier Transform takes away from some of the information known about time, but in this example, we will use the Gabor Transform to focus on local sections of a signal and limit the loss of temporal information.

In the following figure (figure 1), we are looking for the location of the submarine that is traveling through the system. The goal of the study is to predict the location following the final known location of the submarine. Unfortunately, as seen in the figure (which is an isosurface plot of the original data), there is a large amount of noisy data that we must filter out. This plot connects points of equal value, equal acoustic frequencies in our case. We expect the submarine to consistently create acoustic signatures, but the noise obscures the useful data.

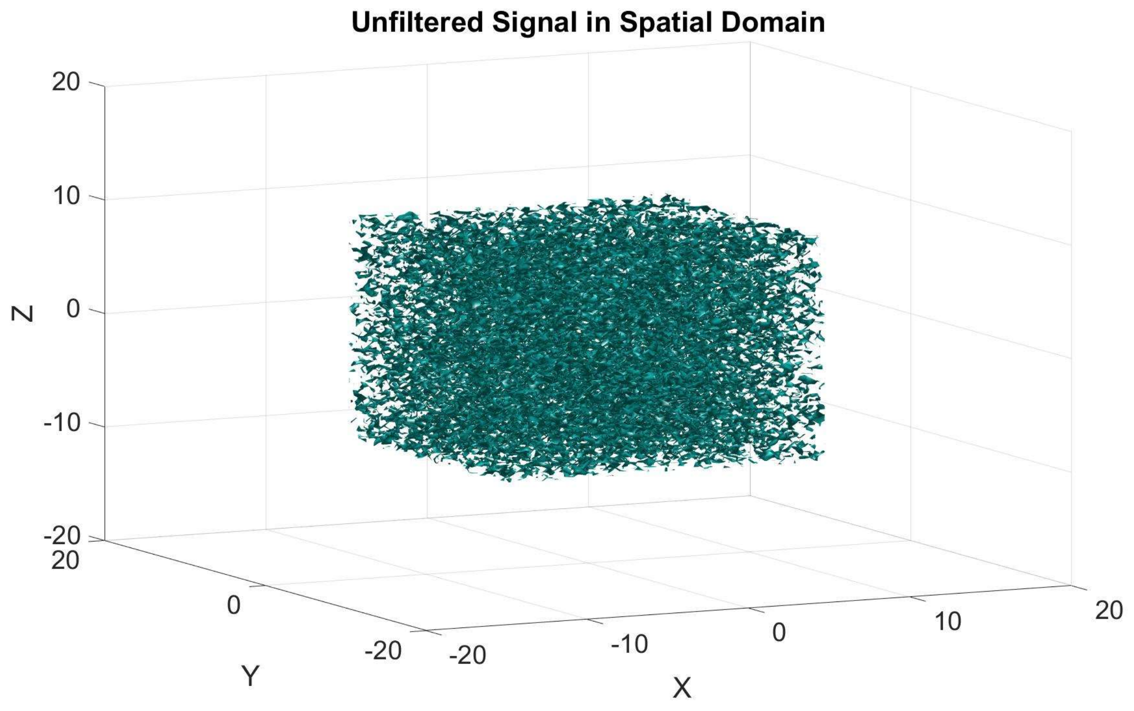

Figure 1: Single Isosurface plot of the initial noisy data.

2 THEORETICAL BACKGROUND

2.1 FOURIER TRANSFORM

A Fast Fourier Transform (FFT) is an algorithm that computes the Discrete Fourier Transform (DFT) of a sequence. In this scenario, the Fourier Transform converts a signal from the spatial domain into the frequency domain and the inverse Fourier Transform converts the signal back into the spatial domain. The DFT is defined as:

$$\mathrm{X_k} = \sum_{\mathrm{n}=0}^{\mathrm{N}-1} \mathrm{x_n} \mathrm{e}^{-2\pi \mathrm{ikn/N}}, \mathrm{k} = 0, \ldots, \mathrm{N}-1 \quad (1)$$

The IFFT Transform is defined as:

$$X_n = \frac{1}{N} \sum_{k=0}^{N-1} x_k e^{2\pi ikn/N}, k = 0, \ldots, N-1 \quad (2)$$

N is some integer 2^n and the Fourier Transform runs at O(Nlog(N)), meaning that it is a very efficient solver, one of the reasons we use it for filtering signals.

2.2 AVERAGING

Let's say our system has zero-mean noise in the frequency domain. That means we have some true signal mixed in with some random noise. Thus, at any given time our signal is the sum of the true signal plus the random noise. However, by Law of Large Numbers, as we increase the sample size we can approximate the signal down to the true signal. Therefore, if we have multiple realizations of the data with similar frequencies, the uncertainty in our signal can be reduced by taking the average in the frequency domain. Doing this, we can approximate the central frequency.

2.3 GAUSSIAN FILTER

As said before, the downside to this technique is that it takes away from some of the spatial and temporal information. When averaging across multiple realizations of data we lose temporal information about the individual realizations of the data. This is especially an issue if the signal is not stationary in space. However, because we narrow down our frequency information to a specific central frequency, we are able to filter the data around this desired frequency. In the frequency domain we can apply the Gaussian Filter function to quickly degrade frequencies away from the frequency of interest. Below is an example of a 1D Gaussian Filter:

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

Applying **equation 3** to the frequency domain, we see that any frequencies (k) away from our central frequency (k_0) quickly approach zero. When we expand this idea to a 3D system we get the following 3D filter:

$$\mathcal{F}(k_x, k_y, k_z) = \exp(-\tau((k_x - k_0)^2 + (k_y - k_0)^2 + (k_z - k_0)^2) \qquad 4$$

In this case, any frequencies that are away from the central frequencies, k_{x0}, k_{y0}, k_{z0} will be filtered out. We can manipulate τ to change the bandwidth or selectivity of the filter.

3 Algorithm Implementation and Development

3.1 Fast Fourier Transform Implementation

The primary tool we using for converting between the spatial and frequency domain will be the FFT and the IFFT. The spatial and spectral resolution of our data is given to be ($L = 10, n = 64$, so we begin the code by defining the spectral and spatial domains. As FFT assumes the domain to be [-π,π], and not [-L,L], we have to rescale our wavenumbers k by $\frac{2\pi}{L}$ and shift them by defining $ks = fftshift(k)$ for convenience. Our signals are in 3D so we have to map these wavenumbers into 3D frequency space, generating K_x, K_y, K_z (Appendix B, code: 7-10).

3.2 Averaging and Indexing

To determine the central frequency, I began by converting each of the 49 realizations of data into the frequency domain using the n-dimensional Fourier Transform and then averaging them across this domain. I also normalized the data for simplicity (Appendix B, code: 17-26). This reduces the variation in the data surrounding the true signal we are looking for. From this, we can find the strongest frequency present by locating the index of the maximum value within our averaged data. We simplify this index into our rescaled wavenumber space to determine the frequency generated by the submarine (Appendix B, code: 28-32). From here we extract the wavenumbers we need from the shifted frequency space: K_x, K_y, K_z. When we did a 3D Fourier

Transform, this generated three separate frequencies for each x,y,z direction which we can obtain by mapping our index into the x,y,z wavenumber domains separately.

3.3 Filtering

Using **equation 4,** we can generate the 3D filter using the central frequencies obtained from earlier in the code. The choice of τ is not clear at first, so I tested several different values to determine when the data was sufficiently filtered, but not filtered out completely. At the end I chose $\tau = 0.05$. Because the central frequencies exist in shifted frequency space, we unshift the filter before applying it to the data (Appendix B, code: 36-40). After this we apply the filter to each realization of the data. For each realization we transform it into frequency space, apply the frequency filter, then shift it back into the spatial domain using the Inverse Fourier Transform. During this loop that goes through each realization, we record the coordinates of the maximum signal in the spatial domain. We assume that this is the location of the submarine that we are looking for, and we later will confirm this by plotting the measurements (Appendix B, code: 49-68).

3.4 Plotting

Inside the same for loop used for each realization of data for finding the coordinates of the submarine, I plotted an Isosurface plot of the measurements. Outside of this loop I used the plot3 command in MATLAB to plot each coordinate of the submarine for each realization (Appendix B, code: 72-84). This will plot the trajectory of the submarine and will tell us the most recent coordinates of the submarine!

4 Computational Results

After averaging the data and finding its maximum, the central frequencies (wavenumbers) were found to be (k_{x0}, k_{y0}, k_{z0})= (5.3407, -6.9115,2.1991). After filtering around these central frequencies, there was a clear and consistent perturbation in the spatial domain across all 49 realizations.

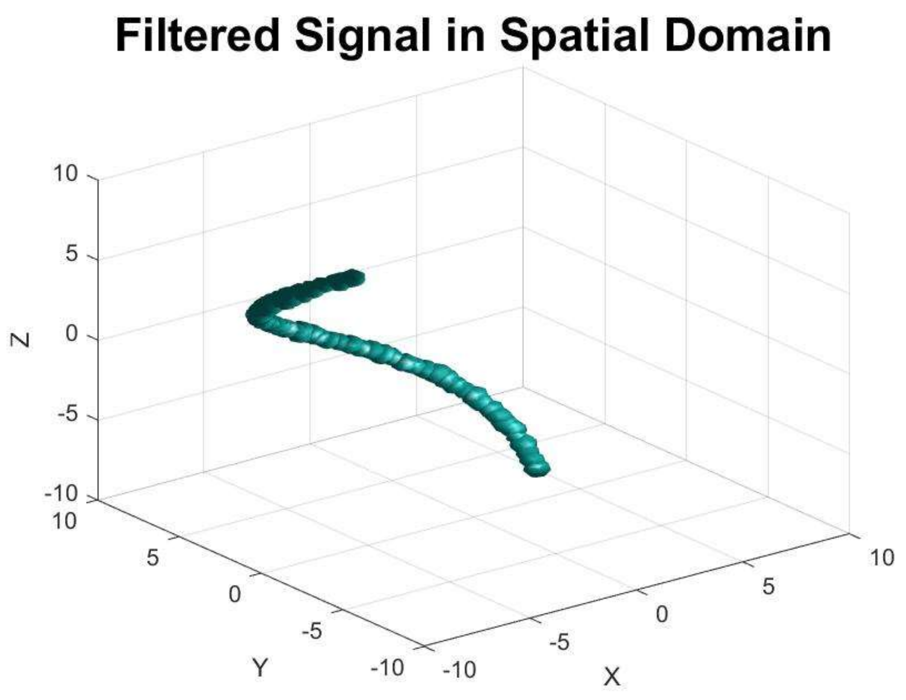

Figure 2: Isosurface plot of post-filtering data in spatial domain

Figure 2 highlights these results and shows that the submarine seems to be following a helical trajectory into the positive z-direction. The path seems smooth and consistent, providing evidence that the results are expected, because a submarine should move along a smooth trajectory. The submarine seems to follow an ellipse in the XY plane and seems to follow sinusoidal trajectories in the x and y direction. In **Figure 3** below, I plotted the coordinates for each realization of the submarine data, or the location of the submarine each time it let out an acoustic signal. The final position of the submarine is approximately (-5.3125, 0.9375, 6.25) given the reference frame we chose. This final location is where we should send the P-8 Poseidon subtracking aircraft and have the P-8 Poseidon follow the trajectory the submarine is going in.

5 Summary and Conclusions

The results show that averaging and filtering techniques with FFT solvers are effective tools for dealing with zero-mean noise in the frequency domain. Averaging is very effective for removing the noise from the data, but it loses some temporal information when multiple realizations of data are combined together. These phenomena can be problematic when the signal is not stationary/varies in time. To counteract this, we can capture the central frequencies of the target signal while averaging, and apply frequency filters to each individual realization of the

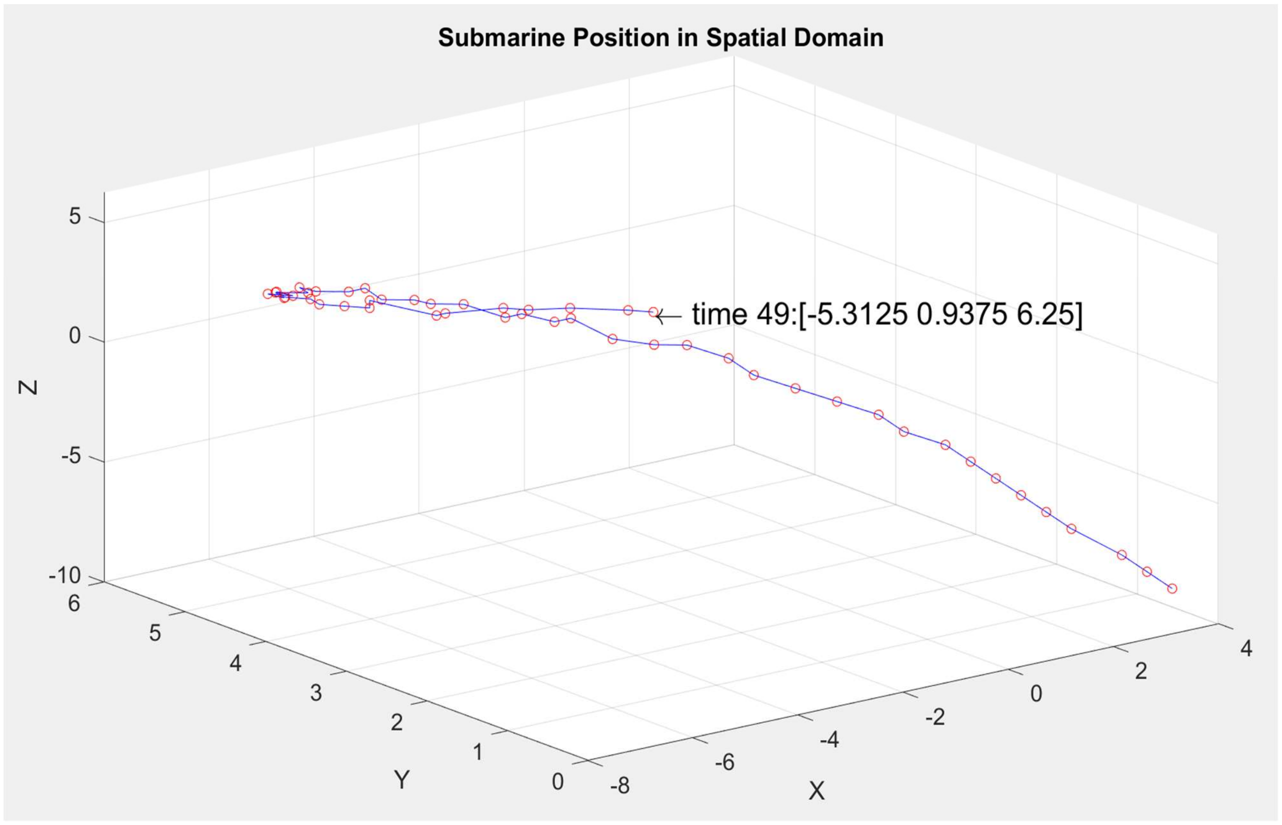

Figure 3:Submarine trajectory through the spatial domain. Begins in the bottom right and follows a helical path in the positive z-direction.

data in order to regain some of the temporal information or limit the loss of it. As mentioned earlier in the paper, the Fast Fourier Transform doesn't preserve any time information because it transforms the entire range of data. However, we can preserve this information by separating the data realizations during filtering. This allows us to track the submarine over time and locate its final location! These conclusive results showcase that the Fourier Transform, filtering techniques, and averaging techniques can be used to dissect meaningful conclusions from highly noisy data.

6 Appendix A

Below is the following MATLAB functions I used and a brief description of their functions:

[X,Y,Z]=meshgrid(x,y,z): Creates a 3D grid of coordinates defined by the X,Y,Z vectors. I used this to define axes to plot the isosurfaces in the space and frequency domains.

abs(x): Returns the absolute value of every element of 'x' or complex magnitude if its complex. I used this function to normalize the data.

Fftn(x): Performs a multidimensional Fast Fourier Transform on x, an N-D array. I used this on the submarine acoustic data to transform it into the frequency domain.

Fftshift(x): Rearranges the contents by placing the zero-frequency component in the center of the array. If x is a matrix, shifts the first and third quadrant and the second and fourth quadrants. I used this to change our transformed data for proper visualization.

Ifftn(x): Performs the multidimensional inverse of the Fast Fourier Transform. I used this after multiplying the centralized filter on the Fourier-transformed data to change it back into the spatial domain.

Ifftshift(x): Inverse of the fftshift function. Used this on the filter to undo the fftshift.

Isosurface(X,Y,Z,V,isovalue): Computes and plots the isosurface from volume data V at the isovalue. I used this to visualize the submarine acoustic signals at different times in the spatial and frequency domain.

Linspace(x1,x2,n): Creates a vector of n evenly spaced points between x1 to x2. I usd this to define the axes to plot the isosurface.

Max(A): Returns the maximum value of the vector A. Used this for normalizing the data.

plot3(X1,Y1,Z1): Displays a 3D plot of a set of datapoints through the inputs (X1,Y1,Z1). I used this to visualize the trajectory of the submarine, show the final coordinate of the submarine, and make predictions on its future location.

reshape(A,'size'): Reshapes the array A into the size defined. I used this to store the data as a 4D 49x64x64x64 matrix and to access each row of this 4D matrix while plotting.

7 APPENDIX B

MATLAB Code

```
clear all; close all; clc

load('subdata.mat')
%% Initial Conditions
L=10; %Spatial Domain
n=64; %Fourier Domain
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);
t = zeros(n,n,n);
%% Setting up unflitered signal to be averaged

%we have 49 samples of the time data
%use these different realizations of the data to
%average out zero mean noise
for j = 1:49
    sub(:,:,:)=reshape(subdata(:,j),n,n,n);
    t = t +fftn(sub);
end
%normalize data and shift in the frequency domain
ave = abs(fftshift(t))/49; %averaging unfiltered signal
%find max to normalize
shape = reshape(ave,1,n^3);
M = max(abs(shape));
ave = ave/M;
```

```
%find frequency index where max occurs
index = find(ave == 1); %Finding the max to filter off of
%central frequencies
kx0 = Kx(index);
ky0 = Ky(index);
kz0 = Kz(index);

%% Filter of Data

tau = 0.05; % filtering bandwidth
filter = exp(-tau * ((Kx - kx0).^2 +  (Ky - ky0).^2 + (Kz - kz0).^2));

% unshift filter as it is currently centered at zero.
filtershift = ifftshift(filter);


% apply this filter to frequency domain at each timestep
%to find submarine at each step
coords = zeros(49, 3);

figure(2)

for j=1:49 % for each realization of data
   sub(:,:,:)=reshape(subdata(:,j),n,n,n);
   subt = fftn(sub);
   subft = subt.*filtershift;
   subf = ifftn(subft);

   [M, I] = max(reshape(abs(subf), 1,n^3));
   coords(j, :) = [X(I), Y(I), Z(I)];

   isosurface(X,Y,Z, abs(subf), 0.6)
   axis([-10 10 -10 10 -10 10]),grid on, drawnow
   hold on;
   title('Filtered Signal in Spatial Domain', 'fontsize' , 20)
   xlabel('X');
   ylabel('Y');
   zlabel('Z');
   if j == 49
       saveas(gcf,strcat('submarine_isoplot.jpg'))
   end
end


%% Plotting the coordinates of the submarine
final_submarine_coordinate_xyz = coords(end, :);

figure (3)
plot3(coords(:, 1), coords(:,2), coords(:, 3), 'ro'), grid on;
hold on;
plot3(coords(:, 1), coords(:,2), coords(:, 3), 'b')
hold on;
txt = strcat('\leftarrow time 49: ', mat2str(final_submarine_coordinate_xyz));
text(final_submarine_coordinate_xyz(1), final_submarine_coordinate_xyz(2),
final_submarine_coordinate_xyz(3), txt, 'fontsize', 20);
xlabel('X');
ylabel('Y');
zlabel('Z');
title('Submarine Position in Spatial Domain');
saveas(gcf, strcat('submarine_position.jpg'));
```