AMATH 582 Homework 1: Denoising Stationary Signal Using FFT

Hongda Li

January 18, 2021

Abstract

The paper seeks to explain the process of using FFT as a way to denoise static signal. Especially using the assumption of the presence of white nosie in the data, we can leverage the statistical feature of normal distribution to reduce the amount of noise in the frquency domain using multiple realizations. To demonstrate it, we were given the hypothetical scenario of searching for a submarine and synthetic data to work with, and the goal is to recover the full path of the submarine.

1 Introduction and Overview

We are interested in locating a submaring in the Puget Sound area given some noisy acoustic data. The assignment is designed to test the skill of filtering out noise in stationary waves, with the reward of figuring out the full path that the submarine takes.

The data given is a series of cube of data, collection every 30 mins during a 24 hours period. The data is in the format of complex double and the magnitude of each of the data point represents the presence and absence of the submarine.

2 Theoretical Background

2.1 Fourier Series/Transform, and Frequency Domain

As we learned from our textbook [**kutz˙2013**], Fourier introduced the concept of representing a given function $f(x)$ by a trigonometric series of sines and cosines:

$$f(x) = \frac{a_0}{2} + \sum_{i=1}^{\infty} (a_n \cos nx + b_n \sin nx) \quad x \in (-\pi, \pi]. \tag{1}$$

And the coefficients of the sin and cos is up to our interests because it transform the signal with additional useful mathematical properties. The continuation of the Fourier Series is Fourier Transform:

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

And the inverse Fourier Transform is:

$$f(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \exp(ikx) F(k) dk \tag{3}$$

The Fourier Transform extend the domain for the coefficients of sin and cos to the real domain.

For computational reasons, signal is discretized into 2^n complex floats and then a discrete Fourier Transform is performed on the signal. The algorithm is relatively efficient with $\mathcal{O}(N\log(N))$ complexity where N is the length of the signal.

However, the Fourier Transform above is applicable for signal in $[-\pi, \pi]$ range, but in actual application, the signal locates in $[-L, L]$ or $[\frac{-L}{2}, \frac{L}{2}]$ (depends on convention) range. The the additional caveats that, the FFT algorithm shifts the domain in its implementation.

Therefore, for any given signal in the domain $[-L/2, L/2]$, the k should rescale to $\frac{2\pi k}{L}$, and when the signal has been discretized: $k_i = \frac{2\pi i}{L}$ where $-\frac{N}{2} \leq i < \frac{N}{2}$. Therefore, our frequency domain goes from $\frac{-\pi N}{L}$ to $\frac{\pi N}{L}$.

2.2 Denoising in the Frequency Domain

The idea here is to observe the same signal over different time fames (call realization), and using the fact that the expected value of white noise in the frequency domain is zero. This idea is Further Discussed in the data science textbook[**kutz˙2013˙pg316**], but the gist of it is summarized below:

Let ϵ_i to be a white noise signal sampled with discretization of N, then:

$$\Re(\widehat{\epsilon_i}) \sim \text{Normal}(0, \sigma) \wedge \Im(\widehat{\epsilon_i}) \sim \text{Normal}(0, \sigma) \implies \mathbb{E}(\widehat{\epsilon_i}) = 0 \tag{4}$$

And this means that, summing over the same stationary signal over different time frame will reduce the magnitude of white noise in the frequencies domain because the real and imaginary part of the white noise has an expectation value of zero.

2.3 Filtering and Reconstruction of the Submarine Path

Because the signal is stationary, therefore the frequency component is unchanging, and in this particular case, the submarine emits a particlar frequency the is unknown.

Therefore after the denoising process, we seek to find the unknown frequencies by simply looking for the frequency component that has maximum magnitude.

After the identification of the signal, we use a Gaussian filter to boost this particular frequencies in the original signal. The Gaussian filter is given as:

$$\exp\left(\mu_x(x - x_i)^2 + \mu_y(y - y_0)^2 + \mu_z(y - y_0)^2\right) \tag{5}$$

Where x_0, y_0, z_0 is the frequencies we try to boost (Should be the signal that the submarine is emitting) and the multiplier μ_x, μ_y, μ_z determines the window size of the Gaussian filter.

After the desired signal emitted by the submarine is identified, using the gaussian filter and applied it to the signal in the fourier domain will allow us to listen to this particular frequency. A similar idea but for 1D signal is discussed in the textbook[**kutz˙2013˙pg312**] , here we extend the same idea into 3D.

because white noise is normal random, there will be significant amount of phase cancellation. Therefore, if we inverse Fourier Transform the filtered signal, looking for the signal that is peaking at each time frame will give us the approximate location of the submarine.

3 Algorithm Implementation and Development

The process of identifying the path requires the following procedures

1. Establish basic parameters such as the spatial and Frequency domain.
2. Taking the average on the frequencies domain for all realization and identify the submarine frequency.
3. Use the frequency to construct a Gaussian Filter, apply the filter to the data and them reconstruct the signal.
4. Identify the path of the submarine from the reconstructed signal.

3.1 Signal Averaging and Frequencies Identification

For the first part, we need to establish the vectors that represents the spatial and frequencies domain.

Algorithm 1: Setup Spatial and Frequency Domain

```
1: Input: Subdata
2: Reshape Subdat into $49 \times 64 \times 64 \times 64$ sequence of data cubes
3: Initialize: L = 10
4: Initialize: ks, to be the frequency domain vector shifted
5: Initlize: X, Y, Z to be 3d meshgrid of equally spaced 64 points (excludes endpoints) on $[-L, L]$
6: Initliaze: kx, ky, kz, to be 3d meshgrid using ks
```

On line 4 of Algorithm 1, the frequency domain vector is given by:

$$\text{fftshift}\left(\frac{2\pi}{2L}\left[\frac{-n}{2} : \left(\frac{n}{2} - 1\right)\right]\right) \tag{6}$$

3.2 Averaging and Identify Unknown Frequencies

Algorithm 2: Identifying Unknown Frequencies

```
1: Precondition: Algorithm 1 has been run
2: Average := zeros
3: for TimeFrame T in Subdata do
4:     Perform FFT on T, with shifts
5:     Average += T
6: end for
7: Average /= Total Number of Realizations
8: Let T be the the magnitude
9: Let T be normalized
10: Identify indices that has element in T which are larger than a given threshold
11: Let the list of indices be: $\mathbb{I}$
```

Take note that, we have selected frequencies with a magnitude that is larger than a certain threshold. This choice is made for the purpose of relaxing the constraint, giving us the possibility see just how much higher is the magnitude of the peaking frequencies compare to all the others.

3.3 Applying Filter and Reconstruction

Algorithm 3: Filtering, Reconstructing Path

```
1: Precondition: Algorithm 2 has been run
2: Use $\mathbb{I}$ and kx, ky, kz, to find a list of coordinates
3: Take the average of the above list of coordinate, it's an estimation of the unknown frequencies, name it P
4: Construct a guassian filter using P, name it GF
5: Initlize Filtered: to store the signal after frequencies has been filtered
6: for each time frame: $F$ in Signal do
7:     Apply shifted FFT on $F$
8:     Apply GF on $F$
9:     Apply FFT shifts on $F$
10:    Apply Inverse FFT on $F$
11:    Store to: Filtered
12: end for
```

Algorithm 3 produces a series of tensors representing the likelihood of the presence of the submarine in the spatial domain for each realization.

3.4 Tracing the Path

Algorithm 4: Tracing the Submarine Path

```
1: Input: "Filtered" from Algorithm 3
2: Initlized: Results, to store a list of 3d coordinates for all timeframes
3: for Timeframe T in Filtered do
4:     Take the magnitude for T, normalized it, and figure out the index with the maximum magnitude
5:     Find the coordinates using the index and X, Y, Z meshgrid
6:     Append it to "Results"
7: end for
8: Visualize Results
```

4 Computational Results

The distribution of the magnitude of the averaged signal in the Frequencies Domain is the most convincing evidence that, a stationary signal exists in the frequencies domain.

As a common knowledge, the distribution of the modulus of a complex number is the Rayleigh Distribution, a Chi Square distribution with DF = 2. Therefore, we expect low level confidence of observing magnitude on the tail of the distribution. Anything contrary observation will yield existence of a stationary signal, intuitively. This is showed in Figure 1, the first plot is fitting the Rayleigh Distribution, however there are a fair amount of frequencies that is higher than the threshold 0.7.

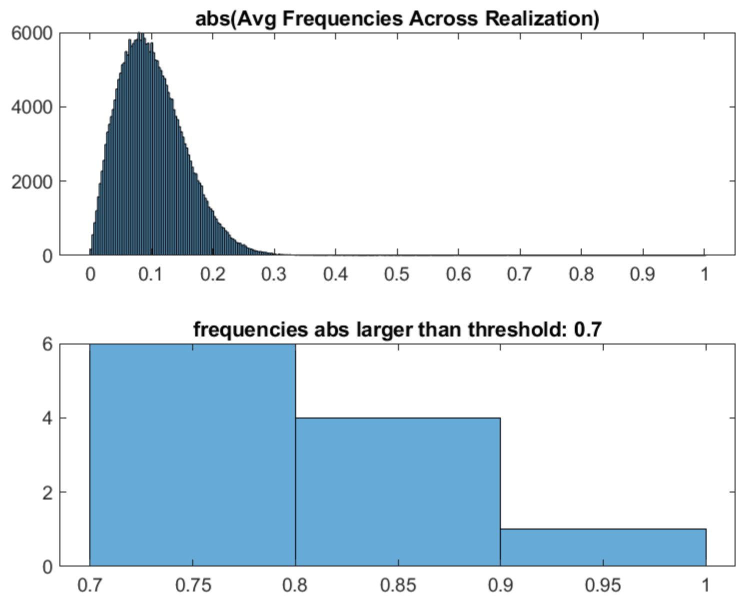

Figure 1: Distribution of the magnitude of averaged frequencies

If the frequencies that is peaking indeed is the frequency that the submarine is emitting, then it will be in a narrow range. Which is indeed the case and the frequencies is plotted in Figure 2, and the red dot is the location of frequency with maximum magnitude, and all other frequencies that is above the threshold close to each other.

Next, we use the approximate location of the frequency, which in the box $[4.6, 5.4] \times [-7.2, -6.6] \times [1.6, 2.2]$, and that is the approximately where the Gaussian Filter is covering. The filter is plotted in Figure 3.

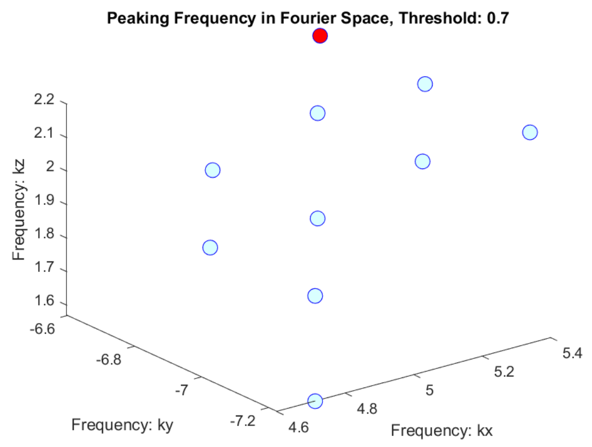

Figure 2: Distribution of the magnitude of averaged frequencies

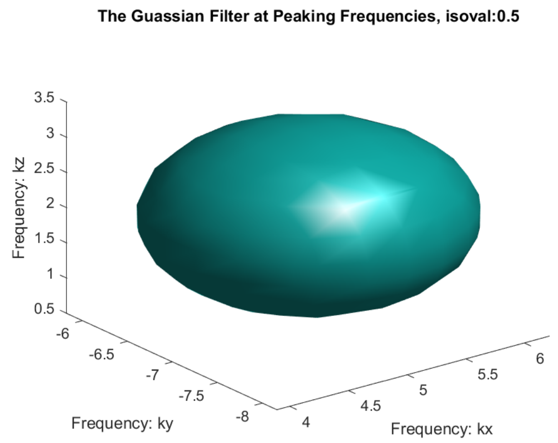

Figure 3: The Gaussian Filter

Finally, the filter is applied in Algorithm 4, and after selecting indices of all peaking signal in the Spatial domain, we were able to recover the full path of the submarine, which is showed in: figure 4. The red star is the final location of the submarine, therefore, we should send the P-8 Orio sub-tracking aircraft to the point $[-5, 0.9375, 6.5625]$.

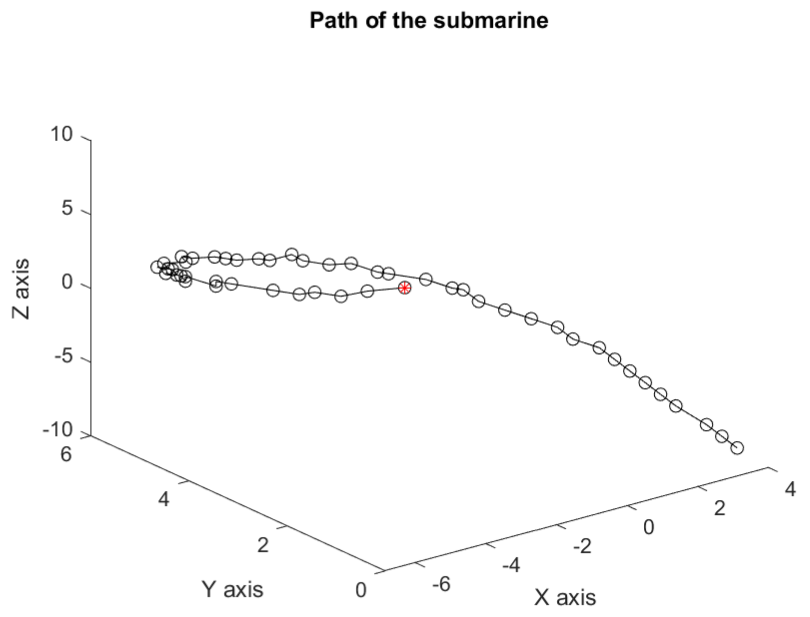

Figure 4: Path of the Submarine

5 Summary and Conclusions

The purpose of the homework is to understand the theoretical background for stationary signal processing, and apply them for this example, while extending some of the ideas in class, mainly, solving the problem in a higher dimension. As we can see, for this example, the method off FFT is effective for the noise filtering, however, more careful analysis should be involved for determining what kind of threshold to place using advanced statistical method, in this discussion however it's entirely empirical and it just happens to work because we have the assumption that it's a homework for a class and it should work out as expected.

Appendix A MATLAB Functions

1. ReconstructAllRealization.m
   Function accepts a shifted series of signal in the Fourier space, and it will reconstruct the signal use Inverse Fourier Transform.

2. MaxFrequency.m
   Given a all frames of realization in the frequencies domain, this function figure out the indices of the frequencies over a certain threshold.

3. IndexToFrequencies.m
   Given a set of indices, it will use the meshgrid to figure out the coordinates in the frequency domain.

4. GetPeakingSignal.m
   Given the signal in the spatial domain, it normalize and return the indices of the points that has the maximum magnitude, this is for figuring out the path of the submarine.

5. FFTAllRealization.m
   For all realizations in

6. BestGaussianFilter.m
   Given the list of indices representing the frequencies that is over a given threshold, it will return a Gaussian Filter, which is a 3D tensor having the same size as each realization in the spatial domain.

7. HW1Demo.m
   This is the main file that should be run to execute all 4 algorithms described in this paper and it will produce all the plots.

```
function [Reconstructed] = ReconstructAllRealization(filtered)
    % Given a series of data cube in the fourierspace that is: shifted, filtered.
    % reconstruct that data using inverse fourier transform.
    % Filtered:
    %   Tensor of the size: [49, 64, 64, 64]
    Reconstructed = zeros(size(filtered));

    for II = 1: size(filtered, 1)
        Reconstructed(II, :, :, :) = ifftn(ifftshift(filtered(II, :, :, :)));
    end
end
```

```
function [MaxFrequencies, Transformed] = MaxFrequency(params, Data, threshold)
   % Maxfreqencies:
   %   Index that gives the frequencies with the strongest
   %   signal.
   % Transformed:
   %   The normalized matrix, denoised,
   %   in fourier domain and shifted.

   n  = params.n;
   Avg = zeros(n, n, n);
   for II = 1: size(Data, 2)
       Cube(:, :, :) = reshape(Data(:, II), [n, n, n]);
       Cube(:, :, :) = fftn(Cube);
       Avg = Avg + Cube;
   end
   Avg = fftshift(Avg);
   Avg = abs(Avg)/max(abs(Avg), [], "all");
   MaxFrequencies = find(Avg >= threshold);
   Transformed = Avg;
end
```

```
function [Freqxyz] = IndexToFrequencies(params, index)
    Kx = params.Kx; Ky = params.Ky; Kz = params.Kz;
    Freqxyz = [Kx(index), Ky(index), Kz(index)];
end
```

```
function [Indices, Points] = GetPeakingSignal(params, signal)
  % Given cubes of signal, in the shape of [49, 64, 64, 64], the function
  % will take out the peaking signal, the indices and it's position

  Indices = zeros(1, size(signal, 1));
  Points = zeros(3, size(signal, 1));

  for II = 1: size(signal, 1)
    Cube = signal(II, :, :, :);
    Cube = abs(Cube)/max(abs(Cube), [], "all");
    Indices(II) = find(Cube == 1);
    Points(:, II) = ...
        [params.X(Indices(II)); params.Y(Indices(II)); params.Z(Indices(II))];
  end

end
```

```matlab
function [FFT, FFTShifted] = FFTAllRealization(dataTensor)
    % Given a tensor in that shape of [p, n, n, n] where it represents
    % a cube data collected with p realization.
    %
    % This function performs FFT transform on the data, with/without shifts, on
    % each of the realization.
    % params:
    %   An instance of ProblemParameters
    % dataTensor:
    %   The submarine Data for the Homework, should be in shape of [1, 64, 64, 64]

    n = size(dataTensor, 1);
    FFT = zeros(size(dataTensor)); FFTShifted = zeros(size(dataTensor));
    for II = 1:n
        FFT(II, :, :, :) = fftn(dataTensor(II, :, :, :));
        FFTShifted(II, :, :, :) = fftshift(FFT(II, :, :, :));
    end
end
```

```
function Filter = BestGaussianFilter(params, peakFrequenciesIndices)
    %   Function figure out the best Guassian filter given the indices of the
    %   frequencies that is peaking in the Fourier Domain.
    % params:
    %   An instance of the class: ProbemParameters.
    % preakFrequenciesIndices:
    %   An array of indices indicating the index of the preaking
    %   frequencies in the frequencies space.
    % Return:
    %   A filter for the freq domain, in the shape of [1, 64, 64, 64]

    Kx = params.Kx;
    Ky = params.Ky;
    Kz = params.Kz;
    for II = 1: length(peakFrequenciesIndices)
        F = peakFrequenciesIndices(II);
        Coords(:,II) = [Kx(F), Ky(F), Kz(F)];
    end
    Center = mean(Coords, 2);
    Variance = [0.5, 0.5, 0.5];
    GaussianFilter = @(x, y, z, vx, vy, vz)...
        exp(-(vx*(x - Center(1)).^2 + vy*(y - Center(2)).^2 + vz*(z - Center(3)).^2));
    Filter = GaussianFilter(params.Kx, params.Ky, params.Kz, Variance(1), Variance(2), Variance(3));
end
```

HW1Demo.m

```
% Files involved:
% 1. MaxFrequencies.m
% 2. IndexToFrequencies.m
% 3. ProblemParameters.m
% 4. FFTAllRealization.m
% 5. BestGuassianFilter.m
% 6. ReconstructAllRealization.m
% 7. IndexToFrequencies.m

close all; clear all; clc;
load subdata;
%% Setting up things
params = ProblemParameters();
Cubes  = zeros([size(subdata, 2), params.n, params.n, params.n]);
for II = 1: size(Cubes, 1)
    Cubes(II, :, :, :) = reshape(subdata(:, II), [params.n, params.n, params.n]);
end

%% Visualizing Peaking Frequencies
params = ProblemParameters();
Threshold = 0.7;
[Freqs, Transformed] = MaxFrequency(params, subdata, Threshold);
% How does the freqiencies magnitude distribute?

Normalized = abs(Transformed)/max(abs(Transformed), [], "all");
figure(1);
subplot(2, 1, 1)
histogram(Normalized); title("abs(Avg Frequencies Across Realization)");
subplot(2, 1, 2)
histogram(Normalized(Normalized >= Threshold));
title(strcat("frequencies abs larger than threshold: ", num2str(Threshold)));
disp(strcat("Index of the preaking frequencies: ", num2str(Freqs)));
disp(num2str(Freqs));
saveas(gcf, "freq-distribution", "png");

%% Where is the peak frequencies?
figure(2);
for II = 1:length(Freqs) - 1
    Freq = IndexToFrequencies(params, Freqs(II));
    plot3(Freq(1), Freq(2), Freq(3), '-o', 'Color', 'b','MarkerSize', 10, 'MarkerFaceColor', '#D9FFFF')
    hold on;
end
 plot3(Freq(1), Freq(2), Freq(3), '-o', 'Color', 'b','MarkerSize', 10, 'MarkerFaceColor', '#FF0000');
title(strcat("Peaking Frequency in Fourier Space, Threshold: ", num2str(Threshold)));
xlabel("Frequency: kx"); ylabel("Frequency: ky"); zlabel("Frequency: kz");
disp("The spread of the peak frequencies are relatively compact inside of the region.");
saveas(gcf, "peaking-freq-position", "png");

%% Getting the Filter
Filter = BestGaussianFilter(params, Freqs);
close gcf;
figure(3);
isosurface(params.Kx, params.Ky, params.Kz, Filter, 0.5);
```

```
title(strcat("The Guassian Filter at Peaking Frequencies, isoval:", num2str(0.5)));
xlabel("Frequency: kx"); ylabel("Frequency: ky"); zlabel("Frequency: kz");
saveas(gcf, "gaussian-filter", "png");

%% Applying the Filter
FilterTensor = Filter;
FilterTensor = reshape(FilterTensor, [1, params.n, params.n, params.n]);
[InFourier, ToFilter] = FFTAllRealization(Cubes);
Filtered = FilterTensor.*ToFilter;
Reconstructed = ReconstructAllRealization(Filtered);
[~, PeakingPoints] = GetPeakingSignal(params, Reconstructed);

%% Visualizing the Paths.
figure(5);
plot3(PeakingPoints(1, :), PeakingPoints(2, :), PeakingPoints(3, :), "ko-");
title("Path of the submarine");
t = linspace(0, 1, size(PeakingPoints, 2));
QueryPoints = linspace(0, 1, 100);
PathX = interpn(t, PeakingPoints(1, :), QueryPoints, "spline");
PathY = interpn(t, PeakingPoints(2, :), QueryPoints, "spline");
PathZ = interpn(t, PeakingPoints(3, :), QueryPoints, "spline");
hold on;
plot3(PathX(end), PathY(end), PathZ(end), "*r");
xlabel("X axis"); ylabel("Y axis"); zlabel("Z axis");
saveas(gcf, "submarine-path", "png");
```