AMATH 482 HW 1

Fourier Transforms, Averaging, and Filtering with Noisy Data

Zachary McNulty
zmcnulty, ID: 1636402

Abstract: This paper aims to demonstrate the usefulness of Fourier transforms and traditional averaging and filtering techniques within the frequency domain. Specifically, we aim to show how meaningful information can be extracted from highly noisy data. As a case study, we utilize these tools to denoise ultrasound data collected from the local veterinarian in order to locate a foreign object within one of their patient's digestive tracts.

University of Washington
January 2019

1 Introduction and Overview

It is often the case that the data we are forced to work with has a significant noise component. Understanding the source of this noise is key to finding appropriate strategies for effectively filtering it out. For example, if we have reason to believe our noise is zero-mean in the frequency domain, the techniques of averaging and filtering discussed in this paper are very effective. To highlight this fact, we apply these techniques to highly noisy ultrasound data. This data is a collection of 20 ultrasound measurements of a dog's digestive tract captured at consecutive points in time. Each measurement captures the spatial fluctuations at each point in a $64 \times 64 \times 64$ grid at that point in time. The data is complex-valued to capture both the phase and amplitude of the fluctuations at each point.

Here, we are looking for a moving, solid object (a marble) that is traveling through our system, perturbing it in a consistent way. The goal is to predict its location following the final data measurement. **Figure 1** below gives an isosurface plot of the initial data. This plot connects points of equal value, perturbation levels in our case, much like a countour plot in 2D. We expect the marble to consistently perturb the system and thus form an isosurface, but the noise obscures the data.

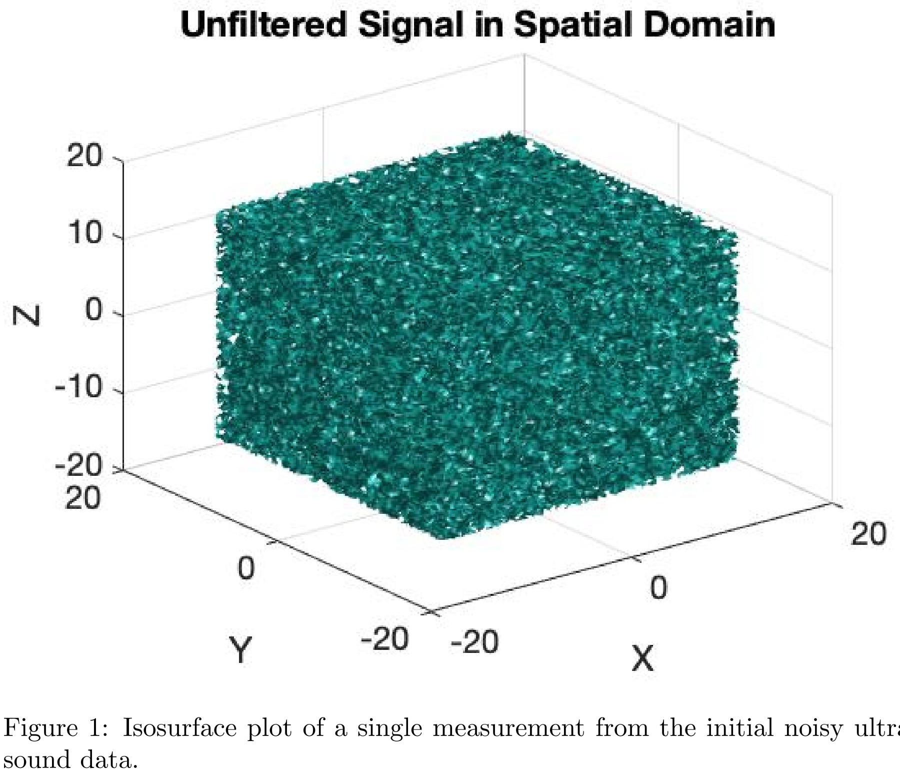


Figure 1: Isosurface plot of a single measurement from the initial noisy ultrasound data.

2 Theoretical Background

Suppose our system is suffering from zero-mean noise in the frequency domain. That is to say we have some true signal μ that is mixed in with some random noise N with $E[N] = 0$ and $V[N] = \sigma_n^2$. Thus, at any given time our signal S is:

$$S = \mu + N \tag{1}$$

Since $E[N] = 0$ and $V[\mu] = \sigma_\mu^2$, $E[S] = \mu, V[S] = \sigma_n^2 + \sigma_\mu^2$. Thus, by the Law of Large Numbers, as we increase our sample size $\frac{1}{n}\sum_{i=1}^{n} S_i \to \mu$. Therefore, if we have multiple realizations of our data, all with similar frequency contents, we can reduce the uncertainty in our signal simply by taking their average in the frequency domain. This allows us to approximate the central frequency of the system at hand.

The downside of this technique is that it convolutes a lot of spatial and/or temporal information. By averaging across multiple realizations of data that occur across space or time, we lose spatial/temporal information about each individual realization of the data. This can be an issue if your signal is not stationary in space. However, by narrowing down our frequency information to a specific central frequency, we are now able to properly filter the data around this desired frequency. In the frequency domain, we can apply a function that quickly degrades frequencies away from the frequency of interest, frequencies which are likely simply noise. Below is an example of a 1D - Gaussian Filter:

$$\mathcal{F}(k) = exp\left(-\tau\,(k - k_0)^2\right) \tag{2}$$

Applying **Equation 2** above to the frequency domain, we see any frequencies k away from our central frequency, k_0, quickly approach zero. Expanding this idea to 3D systems yields a 3D filter:

$$\mathcal{F}(k_x, k_y, k_z) = exp\left(-\tau\left((k_x - k_{x_0})^2 + (k_y - k_{y_0})^2 + (k_z - k_{z_0})^2\right)\right) \tag{3}$$

In this case, any oscillations that are far from the central frequencies $k_{x_0}, k_{y_0}, k_{z_0}$ in the x, y, z directions respectively will quickly be filtered out. We can manipulate τ a bit to change the selectivity of our filter as it effects how quickly signals are degraded away from our central frequency.

3 Algorithm Implementation and Development

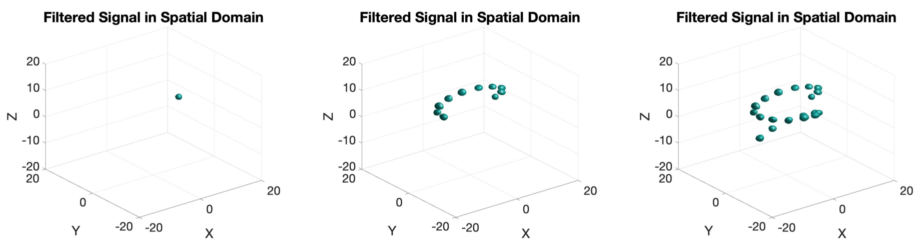
Our primary tool for converting between the spatial and frequency domain will be the Fast Fourier Transform (FFT) and its inverse. The spatial and spectral resolution of our data is known ($L = 15, n = 64$), so we begin by defining our spatial and spectral domains. As FFT assumes our domain to be $[-\pi, \pi]$ and not $[-L, L]$, we rescale our wavenumbers, k, accordingly and shift them to ks for convenience. Since our signals are 3D, we map these wavenumbers into 3D frequency space, generating K_x, K_y, K_z (App B, code: 10-13).

In order to determine the central frequency, we begin by converting each of the 20 realizations of our data into the frequency domain using the n-dimensional Fourier Transform and then averaging them across this domain. For simplicity, we normalize the data (App B, code: 36-51). As we mentioned earlier, this reduces the variation surrounding our true signal. From here, we can find the strongest frequency present by locating the index of the maximum value within our averaged data. Then, we simply map this index into our rescaled wavenumber space in order to determine the true central frequency generated by our object (App B, code: 55-60). Since our averaged data is shifted using fftshift, we extract the wavenumbers we need from shifted frequency space, K_x, K_y, K_z. As we performed a 3D fourier transform, this generates three separate frequencies, one for each of the x, y, z directions which we obtain by mapping our index into the x, y, z wavenumber domains respectively.

Using **Equation 3**, we can generate our $3D$ filter using these central frequencies. The choice of τ is somewhat arbitrary, so we tested the performance of several different values before choosing $\tau = 0.35$. As our central frequencies exist in shifted frequency space, we unshift the filter before applying it (App B, code: 65-73). From here, all that is left is to apply the filter to each realization of the data: for each realization, transform it into frequency space, apply the frequency filter, then shift it back into the spatial domain. During this process, we keep track of the coordinates of the maximum signal in the spatial domain for each measurement. This is likely the object we are looking for, and later we will confirm this (App B, code: 78-104).

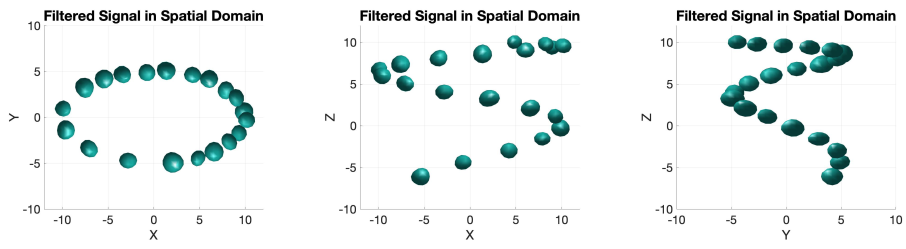
4 Computational Results

Averaging our data and finding its maximum yielded the central frequencies (wavenumbers) $(k_{x_0}, k_{y_0}, k_{z_0}) = (1.885, -1.0472, 0)$. After filtering around these



(a) First Measurement (b) First 10 Measurements (c) All 20 Measurements

Figure 2: Isosurface plots of post-filtering data in spatial domain. Consecutive realizations of data are overlaid to show progression of object over time.



(a) XY Plane (b) XZ Plane (c) YZ Plane

Figure 3: Isosurface plots of post-filtering data in spatial domain, with all measurements overlaid.

central frequencies we found a clear and consistent perturbation in our spatial domain across all 20 measurements. **Figure 2** highlights these results and shows evidence of an object following a helical trajectory throughout the digestive tract. The perturbations are similar to the size and shape of the marble we were expecting, and a helical path centered in the z-direction lines up with our findings that the central frequency in the z-direction is zero (no oscillations in z). In **Figure 3** we can see that the object seems to follow an ellipse in the XY plane, and exhibits no oscillations in the z-direction. Similarly, the object appears to follow sinusoidal trajectories in the x and y direction, although its amplitude appears larger in the x direction. This observation is again inline with the central frequencies we found. In **Figure 4** below, we find the actual coordinates of the object at each point and time, revealing its final position to be $(-5.625, 4.21875, -6.09375)$ given the reference frame we have chosen. This is where efforts to remove or destroy the object should be targeted.

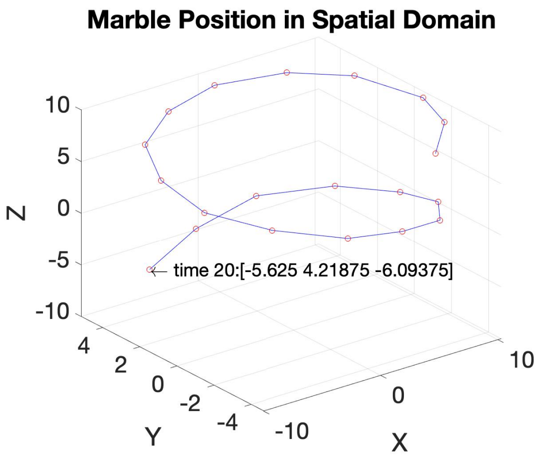


Figure 4: Marble trajectory through the spatial domain. Begins in top right and follows path to bottom left. Final point is labeled.

5 Summary and Conclusions

Averaging and filtering are effective tools for dealing with zero-mean noise in the frequency domain. While averaging is effective for denoising the data, it loses some temporal information when multiple realizations of data are combined together. This can be problematic when your signal is not stationary and varies in time. Thus, after capturing the central frequencies of your target signal using averaging, we can apply frequency filters to each individual realization of the data in order to regain that temporal information. While the Fourier Transform does not preserve any time information, we were able to preserve this information by separating our data realizations during filtering. This allowed us to track the path of the marble over time, locating its final position.

6 Appendix A

Below is a brief summary of the MATLAB functions I used during this project and their functions.

isosurface: Plots the isosurfaces of the given 3D matrix in 3D space. An isosurface is a surface where all points along the surface have the same value.

fftn/ifftn: The former computes the n-dimensional Fourier Transform of the given n-dimensional array. Generates frequency contents in the given n-dimensions. ifftn is its inverse.

fftshift/ifftshift: The former shifts zero frequency component to the center of the frequency spectrum. Works for data in any number of dimensions. Mostly helpful for plotting in the frequency domain, and may not be necessary otherwise. ifftshift is its inverse.

7 Appendix B

```matlab
% AMATH 582/482 HW1

clear all; close all; clc;
load Testdata
%%
L=15; % 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);

total_t = zeros(n,n,n);

for j=1:20 % for each realization of data
    Un(:,:,:)=reshape(Undata(j,:),n,n,n);
    figure(1)
    close all, isosurface(X,Y,Z,abs(Un), 0.4)
    axis([-20 20 -20 20 -20 20]), grid on, drawnow
    title('Unfiltered Signal in Spatial Domain')
    xlabel('X')
    ylabel('Y')
    zlabel('Z')
    set(gca, 'fontsize', 20)
    pause(2)
end

%% Averaging of the spectrum

% we have 20 samples of data: time data
% use these different realizations of the object/data to
% average out zero mean frequency noise.

% sum up all frequency data
for j=1:20
    Un(:,:,:)=reshape(Undata(j,:),n,n,n);
    total_t = total_t + fftn(Un);
end

% normalize the data and shift in the frequency domain so
% it is centered around zero.
```

```

ave = abs(fftshift(total_t)) ./ 20;

% find max value so we can normalize
flat = reshape(ave, 1, n^3);
M = max(abs(flat));

ave = ave / M;

% since we normalized ave, the max must be one.
% find frequency index where max occurs
ind2 = find(ave == 1);

% central frequencies in x,y, and z directions
    respectively.
xc = Kx(ind2);
yc = Ky(ind2);
zc = Kz(ind2);


%% Filtering of the spectrum

tau = 0.35; % filtering bandwidth

% create 3D gaussian filter. If you are far from central
    x,y, OR z
% frequency, your signal will be filtered out. This is in
    shifted
% frequency space however.
filter = exp(-1*tau * ((Kx - xc).^2 +  (Ky - yc).^2 + (Kz
    - zc).^2));

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


% apply this filter to frequency domain at each timestep
% to find marbles location at each step
marble_coords = zeros(20, 3);

figure(2)

for j=1:20 % for each realization of data
    Un(:,:,:)=reshape(Undata(j,:),n,n,n);
    Unt = fftn(Un);
    Unft = filter_s .* Unt;
    Unf = ifftn(Unft);

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

    isosurface(X,Y,Z, abs(Unf), 0.4)
    axis([-20 20 -20 20 -20 20]), grid on, drawnow
    title('Filtered Signal in Spatial Domain', 'fontsize'
        , 20)
    xlabel('X', 'fontsize' , 20);
     ylabel('Y', 'fontsize' , 20);
     zlabel('Z', 'fontsize' , 20);
     xlim([-12, 12]);
     ylim([-10, 10]);
     zlim([-10, 12])
     set(gca, 'fontsize', 25);
     title('Filtered Signal in Spatial Domain', 'fontsize'
         , 30)
     set(gcf, 'position', [100, 100, 600, 500]);
     saveas(gcf, strcat('images/marble_iso', num2str(j), '
         .jpg'));
end


%%
final_marble_coordinate_xyz = marble_coords(end, :)

figure (3)
plot3(marble_coords(:, 1), marble_coords(:,2),
    marble_coords(:, 3), 'ro'), grid on;
hold on;
plot3(marble_coords(:, 1), marble_coords(:,2),
    marble_coords(:, 3), 'b')
hold on;
txt = strcat('\leftarrow time 20: ', mat2str(
    final_marble_coordinate_xyz));
text(final_marble_coordinate_xyz(1),
    final_marble_coordinate_xyz(2),
    final_marble_coordinate_xyz(3), txt, 'fontsize', 20);
xlabel('X');
ylabel('Y');
zlabel('Z');
title('Marble Position in Spatial Domain');
set(gca, 'fontsize', 25);
set(gcf, 'position', [100, 100, 600, 500]);
saveas(gcf, strcat('images/marble_position.jpg'));
```