Filtering Ultrasound Data to Locate an Object

Christina Smith

AMATH 482 - Winter 2020

Abstract

We are attempting to locate an object at a certain point in time using ultrasound data. We have signal data in 3-D space, however the data contains a lot of noise.

We will perform a Fast Fourier transform to obtain the signal values and filter them in the frequency domain. We will then use an inverse Fast Fourier transform to view the de-noised data in 3-D space.

Finally, we will use this data to determine the position of the object. We will plot the trajectory of the object and find its position at the final time sample.

I. Introduction

We are given a data set that consists of noisy signal data in the spatial domain. In order to filter this data, we will use a discrete Fast Fourier transform (FFT) to to analyze our signals in a frequency domain. Once we have transformed the data, we can compute the mean signal strengths. With this information, we may find the frequencies corresponding to the maximum value of the mean of our signals. We will use these frequencies, the central frequencies, to filter the signal data with a Gaussian filter.

Having filtered the data in the frequency domain, we can then perform an inverse discrete Fast Fourier transform (IFFT) to view the de-noised signal data in the spatial domain. By evaluating the signal in the spatial domain using an isoplot, we are able to estimate the position of the marble at each point in time and plot its trajectory.

II. Theoretical Background

The noise in our data set is white noise, which affects all frequencies equally. White noise is modeled with a normal distribution, having mean of zero, therefore

when we compute the mean of our signal data in a frequency domain, the mean of the white noise will not affect the mean of the of our signal data.

We will need to use a discrete Fast Fourier transform (FFT) to determine our discrete signal values in the frequency domain. This will allow us to determine a localized maximum signal strength in the frequency domain.

The transformation of our data to a frequency domain is necessary to find useful mean values. In the spatial domain, the signal strength at any given point will fluctuate with time as the signal is reflected and travels back through 3-D space. If we attempt to average this signal data from our observations we will find that the means are fairly evenly matched throughout the spatial domain.

Our signal strength however will not fluctuate in this way in the frequency domain. Regardless of the signal's location in a 3-D space at any given time, the maximum signal strength will correspond to specific frequencies, known as the central frequencies. By transforming our data into the frequency domain, we can average our signal strengths for each time sample and find a fairly well-defined maximum signal relating to specific frequencies. Given these central frequencies we can de-noise the data by using a Gaussian filter.

Function for the Gaussian filter in 3-D:

$$f(k_x, k_y, k_z) = e^{\left(-\tau\left((k_x-k_{x0})^2+(k_y-k_{y0})^2+(k_z-k_{z0})^2\right)\right)} \tag{1}$$

where τ is the bandwidth of our filter, k_x, k_y, k_z are our frequencies at any given data point, and k_{x0}, k_{y0}, k_{z0} are our central frequencies.

We implement this filter by multiplying it with our signal data in the frequency domain. The differences between the frequencies at any given point and the central frequencies affect how much of our signal data is filtered. When all the frequencies match, the exponent is 0 and the value of the Gaussian filter is at its max of 1, so all of the signal is retained. A increase in the difference between the frequencies at a point and the central frequencies will reduce the value of the Gaussian filter and thereby reduce the signal.

With the filtered signal data we may perform an inverse Fast Fourier transform (IFFT) to revert the de-noised data to the spatial domain. By evaluating the greatest signal strength and its corresponding coordinates we can estimate the object's position. We can view an isoplot of the signal data in the spatial domain and increase

the isovalue (the value of the signal data) closer to the mean maximum signal across our observations. This will cause our isosurface to become smaller, giving us a more accurate spatial location of the object. (See Part IV, Fig. 3) We will find the spatial coordinates corresponding to our given isovalue of 0.4 for each observation and plot a trajectory of the object over time.

III. Algorithm Implementation and Development

Our data has 20 observations of 262144 (64^3) data points, therefore we can create grids with dimension 64x64x64 to contain our signal, spatial, and frequency data. We choose the limits of our spatial domain at [-15,15]. (See Appx. B, lines 7-9)

We create a grid containing spatial domain data by creating vectors with 64 equally spaced coordinates within the range of of our spatial domain. In order to form our frequency domain grid we create a vector with frequency values. This is a base vector of [0:31 -32:-1], then scaled by $2\pi/(2 * 15)$. The 2π in the scaling factor ensures that our frequency increments are periodic in the range of 2π and thus compatible with an FFT. The denominator 30 is to scale the values to our spatial domain. Next we shift the halves of the vector so that values are ordered from lowest to highest. This will be useful for visualizing plots of signal data. (See Appx. B, lines 12-32)

To begin processing the signal data, each observation vector is reshaped into a 64x64x64 grid. We then perform an FFT, while at the same time, we shift the data so that we may plot our signal in the frequency domain. With this information, we will find the maximum mean signal strength as well as the coordinates. The corresponding coordinates in the frequency domain grid will contain our central frequencies for x, y, z. (See Appx. B, lines 42-65)

The central frequencies will be used in the Gaussian filter (Eq. 1). Determining the bandwidth of the filter (τ) requires some trial and error. We have been given the direction to consider 0.4 the value of the true isosurface of the marble so we experiment with different filters and plots to find a smoothest surface and choose the value 0.3 for filter bandwidth. After this, we filter the frequency data at each point and can create an isoplot. (See Part IV, Fig. 2; Appx. B, lines 71-88)

After filtering the data for each observation we revert it back to the spatial domain with an IFFT. It is important to note that because we have shifted our data for plotting, will need to shift it back before computing the IFFT. For each observation, we find the maximum signal value in the spatial domain and the corresponding spatial coordinates, which we will use to plot the object's trajectory. (See Fig. 1; App. B, lines 95-128)

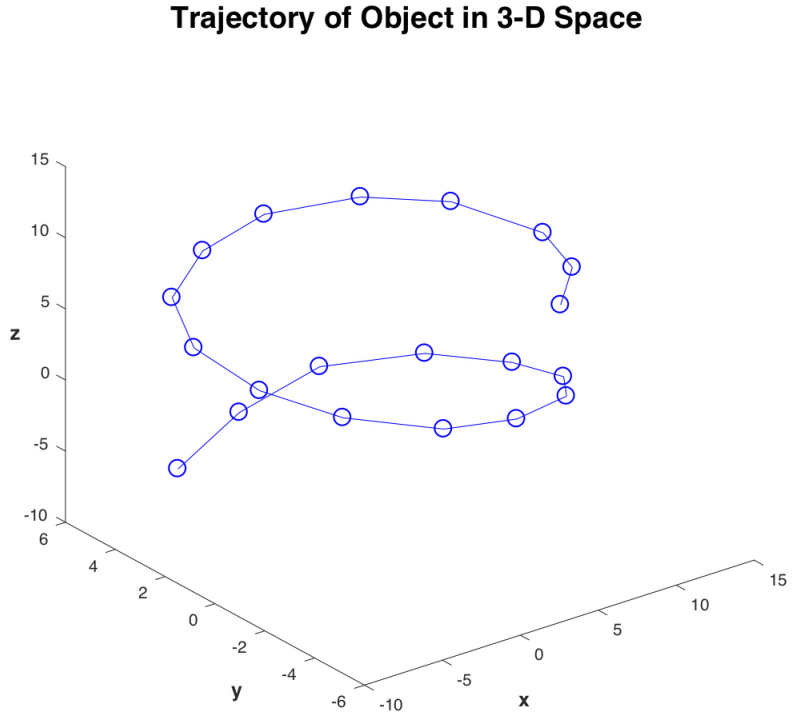

Figure 1: Path of the marble over time

IV. Computational Results and Supplementary Plots

Maximum of mean signal data in frequency domain: 271.8673

Indices of max mean in signal grid: (28, 42, 33)

Central Frequencies:
k_x = 1.8850
k_y = -1.0472
k_z = 0

Spatial Coordinates of Object Position:

Starting Position (t = 1): (4.6875, -4.6875, 9.8438)
Ending Position (t = 20): (-5.6250, -4.2188, -6.0938)

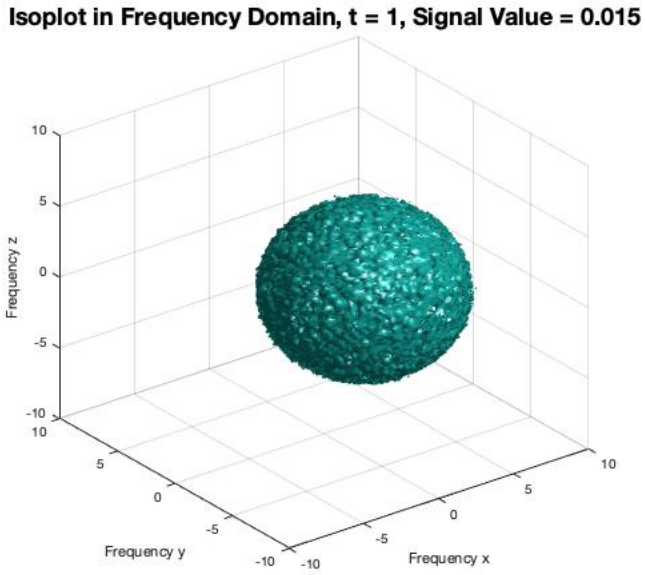

Figure 2: Isoplot in Frequency Domain

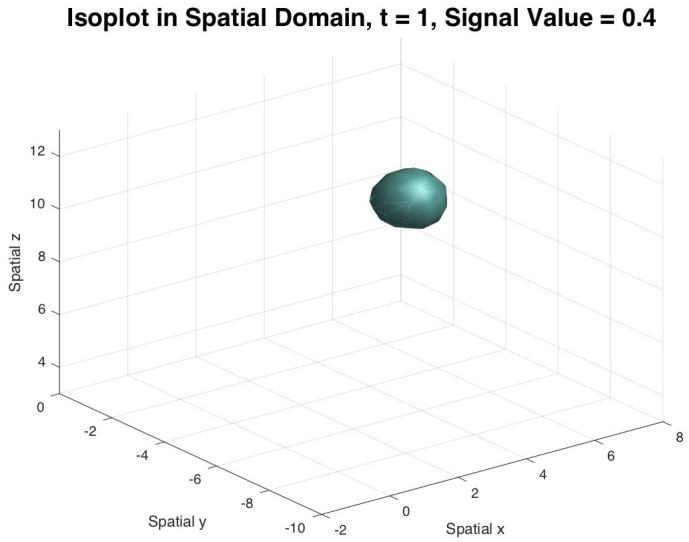

Figure 3: Isoplot in Spatial Domain

V. Summary and Conclusion

The trajectory plot of the marble shows that its path of travel spirals downward roughly perpendicular the z-axis. Its final position has shifted in the negative direction along the x and y axes.

The ending position coordinates show that to make contact with the marble at the last position recorded, a signal should aim for the coordinates (-5.6250, -4.2188, -6.0938).

Appendix A: Matlab Functions

reshape(): This changes the dimension of a given array or vector to given values. In this problem, we change a vector of length 262144 to a 64x64x64 grid.

meshgrid(): Given three vectors as an input, this function creates a 3-D grid with values corresponding to the three vectors as they overlap.

fftn(): This performs a discrete Fast Fourier transform on a multidimensional array (in our case 3-D).

ifftn(): This performs an inverse discrete Fast Fourier transform on a multidimensional array (in our case 3-D).

fftshift(): This shifts the halves of a vector or matrix so that values are ordered from lowest to highest. This is useful for plotting.

ifftshift(): This shifts the halves of a vector back to the position matching the output of the fftn() function.

ind2sub(): Given the value of a linear index and the size of an array, this returns coordinates relating to each different dimension of the array.

isosurface(): In our 3-D domains this plots a surface relating to the points in the domain which have a specified value for the related signal data.

plot3(): Used for plotting points in 3-D space.

Appendix B: Matlab Code

```
%% Filtering Ultrasound Data to Locate an Object

clear all; close all; clc;

load Testdata;

L = 15; % determines spatial domain
t = 20;  % number of data samples
n = 64; % Fourier modes, i.e. # of points in spatial range (2^6)

% 1x65 vector, includes upper & lower bounds of spatial range [-15,15]
x2 = linspace(-L,L,n+1);

% coordinates of each point in the discretized spatial range [-15,15)
x = x2(1:n);
y = x;
z = x;

% grid with spatial coordinates
[X,Y,Z] = meshgrid(x,y,z);

% vector representing frequency domain
% scaled by 2pi/30
k= (2*pi/(2*L))*[0:(n/2-1) -n/2:-1];

% shifts the halves of the vector so that
% values go from lowest to highest
% this is useful for visualizing plots
ks=fftshift(k);

% grid with frequency values with a min value at origin
[Kx,Ky,Kz] = meshgrid(ks,ks,ks);

%% 1
% Through averaging of the spectrum, determine the frequency signature
%(center frequency) generated by the marble.

Un = zeros(n,n,n,t);  % for storing the reshaped original Un data
Un_fft = zeros(n,n,n,t); % for storing the transformed data
sum_fft = zeros(n,n,n); % for storing the cumulative sum of values of transformed dat
```

```
for j=1:t
   % Creates a 3D grid for each observation in Undata
    Un(:,:,:,j) = reshape(Undata(j,:),n,n,n); % stores the grid data

    % transform the grid data to the frequency domain and shift
    Un_fft(:,:,:,j) = fftshift(fftn(Un(:,:,:,j)));

   % cumulative sum of signal values at each point in the grid
    sum_fft = sum_fft + Un_fft(:,:,:,j);
end

% compute the mean signal strength for each point in the array
mean_fft = abs(sum_fft(:,:,:))./t;

% returns the linear index corresponding to the max value in mean_fft
[max_signal, max_idx] = max(mean_fft(:));

% converts the linear index to corresponding x, y, z indices
[max_x, max_y, max_z] = ind2sub(size(mean_fft), max_idx);

% using the max coordinates, find frequencies associated with max signal
kx_center = Kx(max_x, max_y, max_z);
ky_center = Ky(max_x, max_y, max_z);
kz_center = Kz(max_x, max_y, max_z);

%% 2
% Filter the data around the center frequency determined above in
% order to denoise the data and determine the path of the marble.

tau = 0.3;

filter = exp(-tau*((Kx(:,:,:) - kx_center).^2 ...
    + (Ky(:,:,:) - ky_center).^2 ...
    + (Kz(:,:,:) - kz_center).^2));

Un_fft_filtered = zeros(n,n,n,t);

% for each sample, multiply the fiter grid by data grid
for j=1:5
   Un_fft_filtered(:,:,:,j) = filter.*Un_fft(:,:,:,j);
   % plot the signal data in the frequency domain
%    close all, isosurface(Kx,Ky,Kz,abs(Un_fft_filtered(:,:,:,j)),0.015)
```

```
%    xlabel('Frequency x'); ylabel('Frequency y'); zlabel('Frequency z')
%    title("Isoplot in Frequency Domain, t = 1, Signal Value = 0.015", 'Fontsize',18)
%    axis([-10 10 -10 10 -10 10]), grid on, drawnow
%    pause(7)
end

Un_filtered = zeros(n,n,n,t);  % stores filter data (spatial domain)
spatial_coord = zeros(t, 3) % stores the spatial coordinates for trajectory plot

% Inverse FFT on the filtered f-domain data and then plot in spatial domain

for j=1:t
    % transform filtered data to spatial domain
    Un_filtered(:,:,:,j) = ifftn(ifftshift(Un_fft_filtered(:,:,:,j)));

    % get max signal in the spatial domain
    Un_filtered_3x3 = abs(Un_filtered(:,:,:,j));
    [max_signal_spatial, max_idx_spatial] = max(Un_filtered_3x3(:));

    % x, y, z indices of the DATA
    [max_x_data, max_y_data, max_z_data] = ind2sub(size(Un_filtered_3x3), max_idx_spa

    % spatial coordinates
    x_coord = X(max_x_data, max_y_data, max_z_data);
    y_coord = Y(max_x_data, max_y_data, max_z_data);
    z_coord = Z(max_x_data, max_y_data, max_z_data);

    spatial_coord(j,:) = [x_coord y_coord z_coord];

   % plot the signal data in the spatial domain
%    close all, isosurface(X,Y,Z,abs(Un_filtered(:,:,:,j)),0.4)
%    xlabel('Spatial x')
%    ylabel('Spatial y')
%    zlabel('Spatial z')
%    title("Isoplot in Spatial Domain, t = 1, Signal Value = 0.4", 'Fontsize',18)
%    axis([-2 8 -10 0 3 13]), grid on, drawnow
%    pause(2)
end

close all
plot3(spatial_coord(:,1), spatial_coord(:,2), spatial_coord(:,3), '-o','Color','b','M
title('Trajectory of Object in 3-D Space', 'Fontsize',18)
```

```
xlabel('x', 'fontweight','bold','fontsize',12);
ylabel('y', 'fontweight','bold','fontsize',12);
zlabel('z', 'fontweight','bold','fontsize',12, 'Rotation', pi/2);


%% 3
% Where should an intense acoustic wave be focused
% to breakup the marble at the 20th data measurement?

% (-5.6250, 4.2188, -6.0938)
final_coordinates = spatial_coord(20,:);
```