# AMATH 582 HW 1

James Pratt

24 Jan 2020

**Abstract**

Using multidimensional Fast Fourier transforms, spectral averaging, and three-dimensional Gaussian filtering applied to noisy ultrasound data, I was able to locate and compute the trajectory of a marble that my dog Fluffy had swallowed. The trajectory of the marble resembled a vertical helix, with the marble traveling downward in the $-z$ direction. The final position of the marble was $(x, y, z) = (-5.6250, 4.2188, -6.0938)$; an intense acoustic wave was focused on this location to break up the marble and save Fluffy's life.

## 1 Introduction and Overview

The goal of this assignment was to locate and compute the trajectory of a marble that my dog Fluffy had swallowed using noisy ultrasound data. The ultrasound data consisted of twenty measurements of spatial variations of Fluffy's intestines, where the vet suspected the marble was located. Unfortunately, Fluffy could not remain still, and the resulting measurements were extremely noisy.

In order to save Fluffy's life, the data had to be filtered, first by averaging the frequency spectrum to minimize white noise present in each measurement, then by constructing a three-dimensional Gaussian filter centered around the frequency signature revealed by the previous step. Applying this filter to the Fast Fourier-transformed data in the frequency domain, then applying an inverse Fast Fourier transform to the filtered data revealed the trajectory of the marble in the spatial domain, as well as its final location in the twentieth ultrasound measurement.

## 2 Theoretical Background

This assignment required the use of multidimensional Fast Fourier transforms (FFTs), spectral averaging, and 3-D Gaussian filters. Spectral averaging and FFTs are detailed in our textbook, along with one-dimensional Gaussian filters. Each description below is based on those included in our textbook [1].

A Fourier transform is an integral transform defined over the domain $x \in [-\infty, \infty]$, where the transform and its inverse are described by the following equations:

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

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

Equation 1 above transforms the function $f(x)$ from the spatial domain to the frequency domain, where Equation 2 does the inverse; i.e., it transforms the function $F(k)$ from the frequency domain back to the spatial domain.

For the purposes of this assignment, a multidimensional Fast Fourier transform (FFT) was used, in which the concepts outlined above were applied to discretized spatial data. The FFT algorithm assumes that the input signal is $2\pi$ periodic, therefore the frequency input domain was modified to comply with this assumption in the custom algorithm developed for this assignment.

The following equation is an example of a simple Gaussian filter in one dimension:

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

where $\tau$ is the bandwidth of the filter, $k$ is the wavenumber (or frequency), and $k_0$ is the central frequency. The filter eliminates high-frequency components in the system of interest, depending on the central frequency parameter $k_0$. A three-dimensional Gaussian filter was used for this assignment, based on the equation below:

$$F(kx, ky, kz) = exp(-\tau((kx - kx_0)^2 + (ky - ky_0)^2 + (kz - kz_0)^2)) \tag{4}$$

where each $kx_0, ky_0$, and $kz_0$ represent central frequency components in the $x, y$, and $z$ directions of the frequency domain.

Frequency averaging involves transforming the original spatial data to the frequency domain using an FFT and adding each set of measurements to the previous set. This minimizes any white noise in the original data. This technique was used to minimize white noise in the original ultrasound data to enable the identification of the central frequency of the marble.

# 3 Algorithm Implementation and Development

The algorithm used to complete this assignment is outlined below. More specific details may be seen in the MATLAB code included in Appendix B.

1. Load the test data into the work environment.
2. Create vectors of spatial points, and rescale the points to the $2\pi$ periodic domain.
3. Discretize the spatial and frequency domains, and shift the zero value frequency to the center of the frequency domain.

4. Create a 3D matrix to store average frequency values.

5. Apply a multidimensional FFT to the noisy data and average each "slice" to minimize white noise.

6. Obtain the coordinates of the central frequency components to construct a 3D Gaussian filter.

7. Apply the Gaussian filter to the averaged, normalized data in the frequency domain to eliminate any remaining white noise, and plot the results.

8. Apply a multidimensional FFT to the original noisy data, then apply the Gaussian to this data in the frequency domain.

9. Transform the results back to the spatial domain using an inverse FFT.

10. Obtain the coordinates of the marble at each spatial point and plot.

11. Record the final coordinates of the marble.

# 4 Computational Results

The following figures illustrate the results of applying the algorithm outlined above. The central frequency components of the marble were $(kx_0, ky_0, kz_0) = (1.8850, -1.0472, 0)$. These values were used to construct the Gaussian filter, with bandwidth $\tau = 0.35$. The final position of the marble was determined to be $(x, y, z) = (-5.6250, 4.2188, -6.0938)$, which is where an intense acoustic wave should be focused to save Fluffy's life.

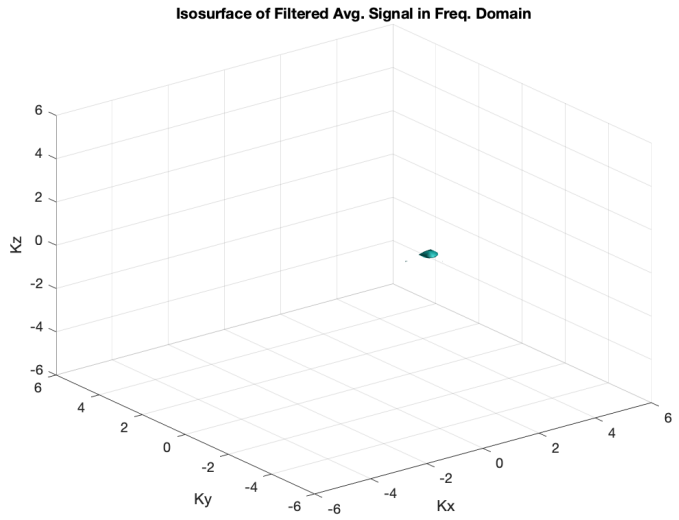


Figure 1: The isosurface plot above displays the location of the marble's central frequency, after applying spectral averaging and a 3D Gaussian filter. The coordinates of the central frequency were used to construct the filter, and to locate the marble in the spatial domain.

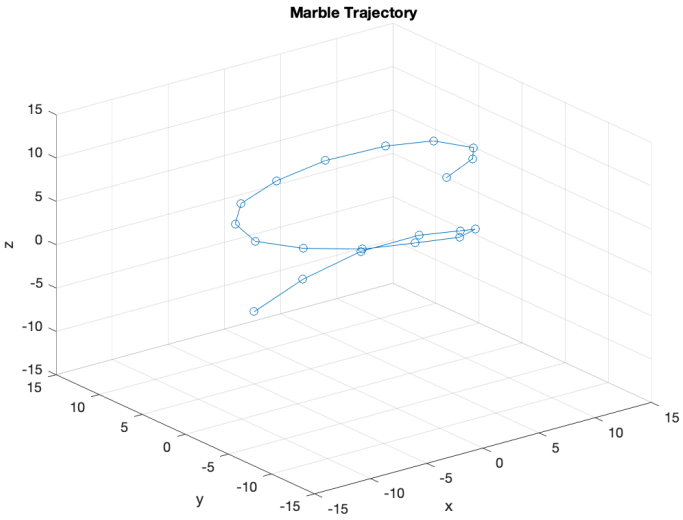


Figure 2: The 3D plot above illustrates the trajectory of the marble after obtaining the marble's coordinates using Gaussian-filtered data. Note how the marble starts near the top-right of the plot, and descends in a downward spiral or helix to the bottom-left of the plot.

# 5 Summary and Conclusions

After Fluffy swallowed the marble, the veterinarian took twenty ultrasound measurements of the spatial variations in Fluffy's intestines. This ultrasound data was noisy, so the marble could not immediately be located.

Through the application of multidimensional FFTs, spectral averaging, and Gaussian filtering, the frequency signature and trajectory of the marble was revealed, and its final position was determined. The coordinates of this final position enabled the vet to focus an intense acoustic wave on the marble to break it up without harming Fluffy, thereby saving Fluffy's life. Fluffy should probably be kept away from any potentially ingestible small objects in the future.

# References

[1] Jose Nathan Kutz, *Data-Driven Modeling & Scientific Computation: Methods for Complex Systems & Big Data*, Oxford University Press, Oxford, 2013.

# 6 Appendix A: MATLAB Functions

The following MATLAB functions were utilized in the course of completing this assignment. More detailed descriptions of each function may be found in the MATLAB documentation.

- `x2 = linspace(-L,L,n+1)` returns a row vector of `n+1` evenly spaced points between `-L` and `L`. This function was used to discretize the spatial domain.
- `ks = fftshift(k)` rearranges a Fourier transform k by shifting the zero-frequency component to the center of the array. This function was used to after implementing an FFT to the original ultrasound data to allow for easier visualization and location of the center frequency components.
- `Un(:,:,:) = reshape(Undata(j,:),n,n,n)` reshapes the `jth` row and all columns of `Undata` into a 3D `n x n x n` array. This function was used to reshape the original ultrasound data into a 64 x 64 x 64 matrix.
- `fftn(Un)` returns the multidimensional Fast Fourier transform of `Un`. It assumes that the signal is $2\pi$ periodic. This function was used on the ultrasound data to extract frequency information.
- `ifftn(Unt)` returns the inverse multidimensional FFT of `Unt`; i.e., it undoes what `fftn(Un)` does.
- `isosurface(X,Y,Z,V,isovalue)` computes isosurface data from the volume data V at the isosurface value specified in isovalue. The `X, Y, Z` values are three-dimensional matrices of points, and `V` is the 3D volume data. This function was used to visualize

both the filtered spatial and frequency ultrasound data to aid in locating the marble and viewing its trajectory.

- `plot3(x,y,z)` produces a 3D plot of the data contained in the vectors/arrays `x,y,z`. This was used to visualize the marble's trajectory.
- `[M,I] = max(Uave(:))` returns the maximum value of the matrix `Uave`, and its indices within the matrix. `M` is the maximum value, and `I` is its linear index. This was used to obtain the maximum values of the entire Fourier-transformed frequency domain.
- `ind2sub(size(Uave),I)` converts the linear index `I` to three subscript indices, analogous to `x, y, z` coordinates in a Cartesian grid. This was used to obtain coordinates for maximum values in both the spatial and frequency domains of the ultrasound data.

# 7 Appendix B: MATLAB Code

The following code was used to complete this assignment. Lines of code whose function or purpose may not be obvious have comments adjacent to them containing brief descriptions or explanations. Note that the code produces additional figures that were not included in the Computational Results section.

Please visit `https://github.com/jpratt84/amath582` to view or download the MATLAB code below.

```
% AMATH 582 HW 1
clear all; close all; clc;
load Testdata

%% Determine Central Freq. of Marble, De-noise and Filter Signal
L = 15; % spatial domain
n = 64; % Fourier modes
x2 = linspace(-L,L,n+1); x = x2(1:n); y=x; z=x; %create vector of spatial
% points
k = (2*pi/(2*L))*[0:(n/2-1) -n/2:-1]; ks = fftshift(k); %scale wavenumbers;
% FFT assumes 2pi periodicity

[X,Y,Z] = meshgrid(x,y,z); % discretize spatial domain
[Kx,Ky,Kz] = meshgrid(ks,ks,ks); % discretize frequency domain
Uave = zeros(n,n,n); % 3-D matrix of zeros for storing average signal
% values

for j=1:20
    Un(:,:,:) = reshape(Undata(j,:),n,n,n); % row, column, slice
    Utn = fftshift(fftn(Un)); %transform data from spatial to freq. domain
    Uave = Uave + Utn; %average each of the 20 measurement sets to minimize
%   white noise
end
```

```
%Plot normalized Uave data as isosurface in freq. domain
figure(1),
close all, isosurface(Kx,Ky,Kz,abs(Uave)/max(abs(Uave(:))),0.6)
axis([-6 6 -6 6 -6 6]), grid on;
xlabel('Kx'), ylabel('Ky'), zlabel('Kz')
title('Isosurface of Averaged Signal in Freq. Domain');
[M,I] = max(Uave(:)) %max value of Uave, index of max value of Uave
% M = 5.4258e+03 + 3.5367e+02i
% I = 133724
[row, col, slice] = ind2sub(size(Uave),I) %convert linear index to row,
% col, slice
% row = 28, col = 42, slice = 33

kx0 = ks(col); %central freq. in Kx, row and col switched from meshgrid
% indexing
ky0 = ks(row); %central freq. in Ky
kz0 = ks(slice); %central freq. in Kz

% Construct 3-D Gaussian filter around central frequency listed above
tau = 0.35; % bandwidth of filter
filter = exp(-tau*((Kx-kx0).^2+(Ky-ky0).^2 + (Kz-kz0).^2));

% Isosurface of filtered, normalized Uave data in freq. domain
figure(2),
isosurface(Kx,Ky,Kz,filter.*abs(Uave)/max(abs(Uave(:))),0.6)
axis([-6 6 -6 6 -6 6]), grid on
xlabel('Kx'), ylabel('Ky'), zlabel('Kz')
title('Isosurface of Filtered Avg. Signal in Freq. Domain');

%% Compute Trajectory of Marble after Filtering

% Create vectors to store spatial position data after filtering
rowf = zeros(1,20);
colf = zeros(1,20);
slicef = zeros(1,20);

figure(3),
for i=1:20
    Un(:,:,:) = reshape(Undata(i,:),n,n,n);
    Utn = fftshift(fftn(Un)); % apply 3-D FFT to noisy spatial data,
%     transform to freq. domain
    Utnf = filter.*Utn; % apply 3-D Gaussian filter around central freq.
%     to noisy data
    Unf = ifftn(Utnf); % transform filtered data back to spatial domain
    [mx,idx] = max(Unf(:)); %obtain max value of filtered spatial data,
%     and index of value
    [row,col,slice] = ind2sub(size(Unf),idx); %convert index to row, col,
%     slice
    rowf(i) = row; %store row index for each of 20 measurements
    colf(i) = col;
    slicef(i) = slice;
```

```
    isosurface(X,Y,Z,abs(Unf),0.4) % plots position of marble for each of
%     the 20 measurements as an isosurface
    axis([-15 15 -15 15 -15 15]), grid on, drawnow
    pause(0.1)
    xlabel('X'), ylabel('Y'), zlabel('Z')
    title('Marble Trajectory over Time')
end

figure(4),
plot3(x(colf),y(rowf),z(slicef),'-o') %plot position of marble based on
% spatial position coordinates obtained above
grid on;
axis([-15 15 -15 15 -15 15]);
xlabel('x'),ylabel('y'),zlabel('z');
title('Marble Trajectory')

% Compute final position of marble
x20 = x(colf(20))
y20 = y(rowf(20))
z20 = z(slicef(20))
```