Analysis of Canine Ultrasound Data

Wyatt Scherer – *University of Washington*
AMATH 582 HW 1, Due 1/24/2020
GitHub Repository: https://github.com/wcscherer/DATA_ANALYSIS-AMATH582

ABSTRACT
A small canine has ingested a marble that can potentially rupture the intestinal wall of the canine. The animal's veterinarian requested assistance with analyzing the ultrasound data to determine the marble's location within the canine's intestinal tract so the marble can be destroyed with a high-intensity sonic pulse. This paper presents the method with which the ultrasound data was filtered for plotting the marble's path and determining the final location of the marble.

I. Introduction: The Ultrasound Data

A veterinarian requested assistance analyzing ultrasound data for a small patient canine. This canine swallowed a marble, which will perforate the small intestines of the canine if the marble is not broken into smaller pieces. The veterinarian took 20 separate ultrasound readings on the same location on the canine over the intestines. These 20 readings must be filtered for the unique marble signal from the large noisy data sets to determine the location and path of the marble as it passes through the intestinal tract. Once the path of the marble is determined, the final location of the marble will be used for focusing a high intensity sonic pulse to break up the marble.

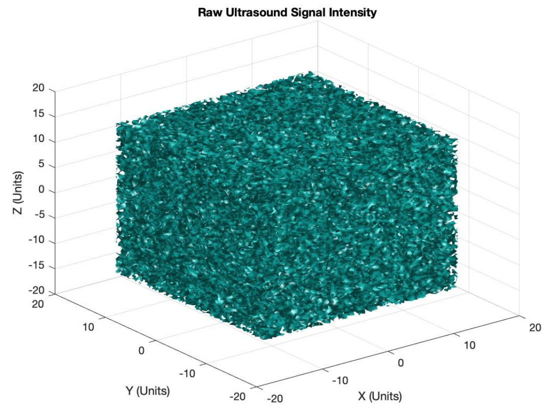

Figure 1: First Sample of Unfiltered Ultrasound Data

II. Theoretical Basis for Signal Filtering

Ultrasound imaging works by applying a transducer to the object that is to be imaged. This transducer emits sound waves between the frequency ranges of 1Mhz to 20 MHz into the object. This transducer also operates as a receiver to detect any sound vibrations coming from the object, including reflections from the initial ultrasonic pulse, referred to as echoes [1]. The time difference between when the transducer emits a pulse and receives a reflected pulse determines the depth of the feature within the object that the waves are reflecting and scattering off of [1].

The data from the ultrasound consists of the spatial location of detected features at a given time step. Each data point in (x,y,z) coordinates represents the detected signal intensity from that point in space. The marble's unique size and composition should be detectable by the ultrasound as a unique signal frequency. The presence of noise in the signal makes it impossible to reliably identify where the marble is in each time frame from the raw data – see Figure 1.

Assuming the canine ingested no other hard objects other than the one marble, only the motion of one marble should be detectable in the data. Furthermore, assuming the canine is restrained such that it cannot move around and therefore move the marble, the marble's path through the intestines should be free from sudden jerks and random movement. Therefore, any remaining signals in the spatial data should be the result of random white noise from spurious reflections, refractions, and scatterings of the ultrasound pulses along with the detection of random pulses from other sources.

Assuming the marble does not significantly change size or shape while ingested, it should provide a unique, constant signature in the ultrasound data. Because of its unique size and structure, the pulses reflected from the marble will have a distinct, constant frequency detected by the ultrasound. This distinct frequency should be common to all 20 data samples and can be filtered for in the signal frequency data. Before an appropriate filter can be developed, the frequencies detected by each pixel must be extracted from the intensity data. To convert the spatial intensity data, the data must be transformed into the associated frequency components. A Fourier transform allows for converting data between different 'spaces' or representations over a periodic domain. [2].

A Fourier transform is built on the definition of Fourier series. A Fourier series represents an arbitrary function *f(x)* as an infinite sum of sines and cosines – see Appendix A for further detail. The resultant series of sines and cosines produce a 2π periodic solution over x∈(-π, π] [2].

The sine and cosine functions can be represented in complex space, so Fourier series can be written using complex exponentials.

$$f(x) = \sum_{-\infty}^{\infty} c_n e^{in\pi/L} \;\; x \in (-L, L) \;(1)$$

where the Fourier coefficient c_n is defined as:

$$c_n = \frac{1}{2L}\int_{-L}^{L} f(x)e^{-inx/L}dx \quad (2)$$

Using this definition of a Fourier series, the Fourier transform can be developed. In general, a Fourier transform performs an eigenfunction expansion over all continuous eigenfunctions, k, which are interpreted as the wave numbers of the associated oscillatory expansion [2].

$$F(k) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty} e^{-ikx} f(x)dx \;\; (3)$$

$$f(x) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty} e^{ikx} F(k)dk \;\; (4)$$

Here the function *f(x)* is defined over the whole domain x∈[-∞ , ∞]. The function f(x) represents the function to be transformed, and F(k) represents the expansion of f(x) in frequency space. When performed over a finite domain, x∈[-L, L] the expansion becomes a discrete sum [2].

Applying a Fourier transform to the raw ultrasound signal intensity data will provide the associated signal frequency data at each corresponding location. Computationally, the Fast Fourier Transform (FFT) algorithm allows for fast computation, *O(Nlog(N))*, of Fourier transforms. The key to this speed is that the FFT algorithm discretizes the x∈[-L, L] domain into 2^n points. This means that the associated transformed frequencies, k, are also discretized into 2^n possible frequencies. This process produces periodic solutions of n points on x∈[-L, L] [2]. It is important to note that data operated on by the FFT algorithm must be shifted by half of a period along the transformed domain to be viewed in its mathematically correct position. A brief discussion of how the FFT algorithm works can be found in Appendix A. The magnitude of the wave numbers, k, produced by the transform represent the 'weight' that specific frequency consist of in the transform of *f(x)*.

Once the ultrasound signal data can be transformed into the frequency domain, it can be filtered to find the unique marble signal frequency. The frequency reflected from the marble is expected to be a pseudo-spherical wave reflected from the ultrasound pulse, therefore the marble frequency will have three components with one component along each dimension (x, y, z).

Since it is unknown what the frequency of the reflected wave from the marble will be, the marble cannot be directly filtered for by its frequency. The noise must be filtered out first to determine if there is any meaningful signal. Assuming the one marble is the only hard foreign object inside the canine's intestines, any localized peak in frequencies after filtering reasonably assumed to be the marble.

To filter the noise, the nature of the noise must first be determined. Without any other objects in the intestines, the resulting noise should be random noise from various scattering, reflection, and refraction events of sound waves passing through the canine at the moment of data acquisition. By applying the Strong Law of Large Numbers, as the number of measurements of signal noise increases, the average of the signal approaches the mean – see Appendix A for further details. Unbiased random noise should be uniformly distributed around an amplitude of 0. By averaging the frequency space ultrasound data over the 20 measurements, the frequencies representing random noise should average to a small number approaching 0 while the marble frequencies will not. This process allows the average signal to be de-noised in order to find the characteristic marble frequency. The marble frequency should then be the max frequency value in the averaged spectral data as it is the only biased measurement. The average signal can then be normalized to this maximum frequency value so the location of the peak can easily be searched for in the data.

With the three components of the marble frequency identified a suitable filter can be developed. With the presence of noise and measurement uncertainty on the data, the peak frequency may identified for the marble may not always be the peak. However, the peak frequency in each frame should be close to the identified max frequency. To capture frequencies within a tight bandwidth around the central max frequency a Gaussian filter should be implemented. The Gaussian filter centered around the identified max marble frequency $\boldsymbol{k_m}$ = (kx, ky, kz) with filter bandwidth τ is implemented on each of the 20 ultrasound data samples as

$$G_f(\boldsymbol{k}) = e^{-\tau\left((Kx-k_x)^2+(Ky-k_y)^2+(Kz-k_z)^2\right)} \quad \textbf{(5)}$$

where *Kx*, *Ky*, and *Kz* represent the frequency domain corresponding to the x, y, and z spatial dimensions [2]. The value of the filter exponentially approaches 1 as the evaluated frequency approaches the marble frequency $\boldsymbol{k_m}$. All other frequencies are reduced by a factor less than 1, making frequencies outside the bandwidth τ negligible.

By applying the filter to each data sample in the frequency frame, it filters out most of the noise in each sample, leaving only the frequencies corresponding to the marble signal. After each sample is filtered, it can be transformed back into the spatial domain with most of the noise filtered out. Now the spatial data consists mostly of the marble signal with some low amplitude residual noise. Each data sample can be searched for the max signal value in the spatial domain. This maximum signal value corresponds to the marble signal and the location of the maximum corresponds to the approximate location of the marble.

There is some uncertainty in the exact marble frequency and location associated with information lost through the transformation from the spatial domain and frequency domain, but this is negligible. If the frequency data is determined with high resolution, the resulting spatial data will be of lower resolution and vise versa. This is fundamental property of Fourier transforms and cannot be avoided, only mitigated [2]. To focus the sonic pulse to destroy the marble, approximate knowledge of the final location of the marble is sufficient.

III. Filtering Algorithm Development

The MATLAB code developed for this analysis can be found in Appendix B. This section describes the high level implementation of the filtering process described in Section 2. A description of the MATLAB functions used in this code can also be found in Appendix A.

1) Import the Data

The raw ultrasound data is stored as a 20x64^3 data structure: 20 independent measurements comprised of 64^3 pixel data points. The physical domain covers -15 units to 15 units along each dimension x, y, z. Each dimension is discretized into 64 nodes (2^8) so the FFT function can be applied.

- Use *linspace* to generate the spatial domain -15 units to 15 units along x, y, and z. Due to periodicity of the solution, the number of points generated is 64+1 and the span of the points goes from 1 to 64 out of the 65 points.
- Generate the frequency domain along each from 0 to (n/2-1) and –n/2 to -1 for the FFT shifted spectra. Scale the frequencies by 2π/L since FFT operates on a 2π periodic domain. Use *fftshift* to generate the shifted frequencies
- Use *meshgrid* to generate the 3D dimensions for the spatial domain X, Y,Z and the un-shifted and shifted frequency domains Kx, Ky, and Kz. These are used for filtering and plotting.

2) Average Frequency Data for Kmax

- Loop through all 20x64^3 data slices
- Within the loop, use *reshape* to import each data slice as a 1x64^3 matrix.
- Use *fftn* to transform the spatial data
- Add the absolute value using *abs* of each transformed data slice to an accumulator variable
- Use *reshape* to save the raw data and the transformed data as 64x64x64 matrices for use later in the algorithm
- Outside the loop, divide the average accumulator variable by the number of data slices.
- Use *max* to find the maximum value and the associated linear index of the value in the average spectrum matrix
- Use *ind2sub* to convert the linear index into a 3D index. Note that *ind2sub* outputs the 3D index components as z, y, x for un-shifted frequency space in MATLAB 2019b.
- Plug the indices into the Kx, Ky, and Kz matrices to find the unshifted k_x, k_y, k_z components of the marble frequency. Shifting the frequencies is only necessary for visualizing data, not processing.
- Plot the marble frequency using *isosurface* in frequency space using *fftshift* on the average frequency data normalized by the maximum and using the fftshifted Kx, Ky, and Kz domains. Set the *isosurface* plot to only show values above 0.6. This picks out the max max frequencies.

3) Build the Gaussian Filter

Use the calculated marble frequency to build a 64x64x64 Gaussian filter as described in equation (5). Use the unshifted frequency domains Kx, Ky, and Kz.

- For choosing the bandwidth, pick a small number on the order of 0.1 to only pick out the max frequency and frequencies close to the maximum.
- Apply the filter to all 20 of the 64x64x64 data slices shifted to the frequency spectra within a loop
- Within the loop inverse Fourier transform the filtered data back into the spatial domain by using *ifftn*. This will provide filtered spatial data for viewing.
- Within the loop, also *reshape* the 64x64x64 filtered data into 1x64^3 for
- Apply the *max* function to this reshaped data to find the max value and its associated index in space for each data slice.
- Use *ind2sub* to find the associated 3D spatial components of each maximum for each data slice using the X, Y, and Z 3D domain matrices.

4) Plot Marble Path and Final Location

With filtered frequency domain data and spatial domain data, the marble's path can be plotted and the location of the marble at the 20th measurement can be determined.

- Use *plot3* to plot the spatial location of each maximum location of the filtered spatial data
- Use the X, Y, and Z 3D domain matrices and the indices for the maximum spatial signal location to determine the x, y, z coordinate of the marble in the 20th data slice. This is where the sonic pulse should be focused to break up the marble.

Note that some of the plots generated for this report were modified after the main script was run and therefore not documented in the MATLAB code in APPENDIX B.

IV. Computational Results

Figure 2 shows the max frequency distribution of the averaged frequency domain data of the 20 data slices. This is the average shifted frequency spectrum of the marble signal over the 20 data slices normalized to the max value of the average. Figure 2 also shows the approximate location of the maximum average frequency signal on the isosurface plot for verification.

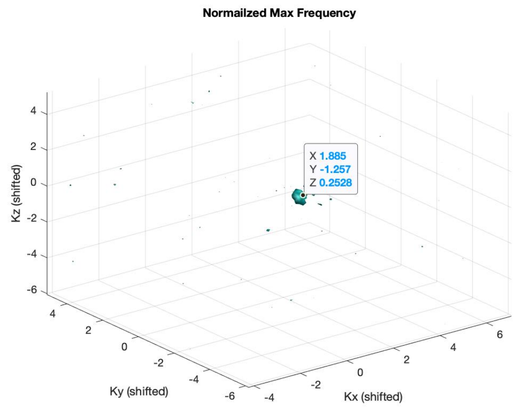

Figure 2: Normalized Average Frequency over 20 Samples

The indices of the max frequency location determined the marble's characteristic frequency components to be $\mathbf{k_m}$ = (1.8850, -1.0472, 0). These values were used to construct the Gaussian filter described in Sections II and III. The filter was applied to the frequency domain data for each data slice. This filtered frequency domain data was then transformed back into the spatial domain to locate and visualize the marble position in each data slice. Figure 3 shows the filtered spatial data for each data slice with a bandwidth value of 0.2.

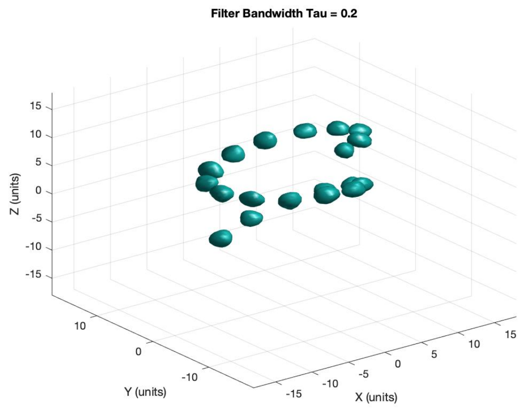

Figure 3: Filtered Spatial Data of each Measurement with τ = 0.2

Figure 4 shows the same spatial data filtered with a Gaussian filter with a bandwidth of 0.5.

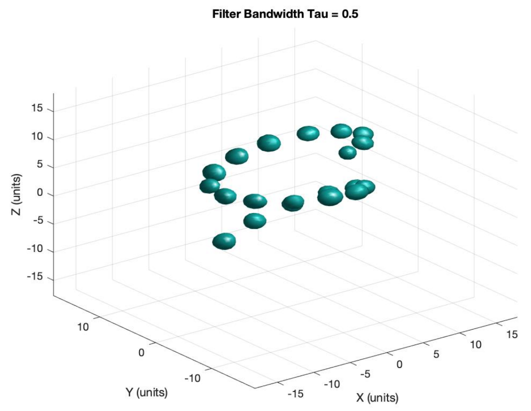

Figure 4: Filtered Spatial Data of each Measurement with $\tau = 0.5$

The outline of the marble is clear in both Figures 3 and 4 with differing filter bandwidths. Since both filters show essentially the same spatial signal for the marble, the effect of noise on the marble's signal location does not seem to be large for the filtered spatial data. This means the approximate locations of the marble in (x,y,z) coordinates translated from the signal can be considered accurate enough for determining the marble's final location in the 20th data slice. Figure 5 shows the simplified path the marble takes through the intestines over the 20 measurements.

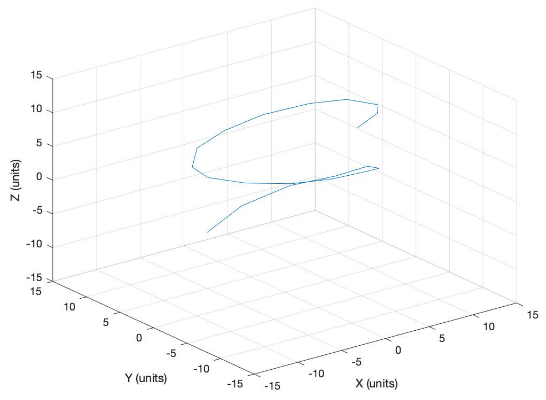

Figure 5: Path of Marble through the Intestines

The approximate final location of the marble was taken from the spatial indices of the max signal location of the filtered spatial data. The indices of the maximum filtered signal location of the 20th measurement correspond to the spatial coordinates L_{m20} = (-5.6250, 4.2188, -6.0938). These coordinates agree with the final location of the marble in Figures 3 and 4.

V. Summary

The ultrasound data was successfully filtered of noise and the marble signal frequency was determined to be $\mathbf{k_m}$ = (1.8850, -1.0472, 0). This characteristic frequency was used to filter the spatial data for the marble's location in the canine's intestines for each of the 20 measurements. The location of the marble in the 20th measurement was determined to be approximately:

$$L_{m20} = (-5.6, 4.2, -6.1).$$

A high amplitude sonic pulse focused at these coordinates should break up the marble, rendering it harmless to the canine.

VI. References

1. *"Medical diagnostic ultrasound- physical principles and imaging",* Jens E. Wiljem et. al., DTU Elektro Technical University of Denmark, Ver. 3.1, 12/2016. Web pdf. *http://bme.elektro.dtu.dk/jw/webbook/Ultrasound/main.pdf*
2. "Ch. 12-13." *Data-Driven Modeling Et Scientific Computation: Methods for Complex Systems Et Big Data*, by J. Nathan. Kutz, Oxford Univ. Press, 2013.
3. "Ch. 19." *Handbook of Mathematics*, by Bronshtein Ilía Nikolaevich. et al., 6th ed., Springer, 2015.

APPENDIX A) Supplemental Formulae

Fourier Series Definition

Representing an arbitrary function *f(x)* as a sum of sines and cosines.

$$f(x) = \frac{a_0}{2} + \sum_{n=1}^{\infty}[a_n\cos(nx) + b_n\sin(nx)]$$

where

$$a_n = \frac{1}{\pi}\int_{-\infty}^{\infty} f(x)\cos(nx)\,dx \quad n \geq 0$$

$$b_n = \frac{1}{\pi}\int_{-\infty}^{\infty} f(x)\sin(\mathrm{nx})\,dx \quad n > 0$$

The coefficients a_n and b_n correspond to the wave numbers extracted in a Fourier transform. The expansion basis can be changed to complex exponentials by recognizing that

$$e^{\pm ix} = \cos(x) \pm isin(x)$$

This allows the Fourier series to be written as such in Equations 1 and 2. The expanded function *f(x)* is still assumed to be real, which requires that the coefficients c_o is real and $c_{-n} = c_n^*$ [2].

Fast Fourier Transform (FFT) Discussion

The key to the speed of the FFT algorithm is the fact that a discrete complex Fourier transformation of length N = 2n can be reduced to two transformations with length N/2 = n. The reduction of the Fourier transformation into two Fourier transformations of half the original size can be carried out recursively over the number of points. As long as the discrete sum is a power of 2, the number of operations required is

$$\frac{N}{2} log_2(N)$$

as mentioned in Section II [3].

MATLAB Functions Used

Full explanations and examples for each function used can be found on MathWorks' website.

- *linspace* creates a linear distributed vector of n points between the lower bound and upper bound of the distribution.
- *meshgrid* creates 2D and 3D coordinates from the input vectors for each dimension
- *reshape* changes the dimensions of a vector or matrix depending on the users input
- *abs* returns the absolute value of real numbers and the real component of imaginary numbers
- *max* returns the maximum value of each column of data in a matrix and can also output the the linear index of each max value corresponding to each input matrix column
- *ind2sub* converts linear indices to the indices corresponding to the equivalent indices along the dimensions of an input matrix
- *fftshift* shifts the Fourier transformed data to its mathematically correct location along the frequency space
- *fftn* performs a Fast Fourier transforms a N-dimensional matrix, *fft* only works on 1D vectors
- *ifftn* performs an inverse Fourier transform on a transformed N-dimensional matrix
- *isosurface* plots surfaces of volumetric data that have the same value. It can also filter for values above a threshold to reduce the amount of plotted data.
- *plot3* plots vectors and points in 3 dimensions

APPENDIX B) MATLAB Code

```
%AMATH 582 Homework 1: Ultrasound Data Filtering - W Scherer 1/24/2020
% Find the frequency components of the marble in the data
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(k,k,k);
[Ksx,Ksy,Ksz]=meshgrid(ks,ks,ks);

ut_avg = 0;

% FFT the spacial data for each slice into frequency space to find the
% marble frequency
tdat = [];
kdat = [];
plot_sum = zeros(n,n,n);
for j=1:20
    % working with each slice as a 1D vector of 64^3 values to make finding
    % max value easier - unflond linear index later to find 3d indeces
    Un(:)=reshape(Undata(j,:),1,n^3);
    Unt(:) = fftn(Un(:,:,:));
    ut_avg = ut_avg + Unt; % accumulate fftn values

    tdat{j} = reshape(Undata(j,:),n,n,n);
    kdat{j} = fftn(tdat{j});
     % this is for plotting the marble k
    plot_sum = abs(kdat{j}) + plot_sum;
end

%Plot the first noisy unfiltered spatial data sample
figure(1)
    isosurface(X,Y,Z, abs(tdat{1}),0.5)
    title('Raw Ultrasound Signal Intensity')
    xlabel('X (Units)')
    ylabel('Y (Units)')
    zlabel('Z (Units)')
    grid on;
    axis([-20 20 -20 20 -20 20]);



% determing the average frequency at each point to find the max frequency
uk_avg = abs(ut_avg)/20; % avg over all 20 time measurement slices
[Mk, I] = max(uk_avg);    % find the max value and the associated linear
index
[kk, jj, ii] = ind2sub(size(Ky),I); % convert the linear index into 3d

%Plot the identified marble frequency
plot_avg = abs(fftshift(plot_sum))/20; % frequency space with noise filtered
figure(2)
    isosurface(Ksx,Ksy,Ksz, plot_avg/max(plot_avg,[],'all'),0.8)
    title('Normailzed Max Frequency')
    xlabel('Kx (shifted)')
    ylabel('Ky (shifted)')
    zlabel('Kz (shifted)')
    grid on;
    axis([-10 10 -10 10 -10 10]);

%determining the 3 components of the marble frequency using the indeces
kxm = Kx(ii,jj,kk);
kym = Ky(ii,jj,kk);
kzm = Kz(ii,jj,kk);
kmax = [kxm, kym, kzm];

% Now with the marble frequency located, filter each of the 20 frames to
%find the location of the marble in each frame

% Build 3D gaussian filter around each frequency for filter
tau2 = 0.2; tau5 = 0.5; % filter bandwidth
%gf_3d = exp(-tau.*(Ksx-kxm).^2).*exp(-tau.*(Ksy-kym).^2).*exp(-tau.*(Ksz-
kzm).^2);
gf_3dt2 = exp(-tau2.*((Kx-kxm).^2+(Ky-kym).^2+(Kz-kzm).^2));
gf_3dt5 = exp(-tau5.*((Kx-kxm).^2+(Ky-kym).^2+(Kz-kzm).^2));
dat_filt2 = [];
dat_filt5 = [];
dat_space2 = [];
dat_space5 = [];

marble_xyz2 = [];
maxval_xyz2 = [];

marble_xyz5 = [];
maxval_xyz5 = [];
% this loop should be modified to be a function later
for i = 1:length(kdat)

    %apply the filter to each instance of the frequency space
    dat_filt2{i} = gf_3dt2.*kdat{i};
    dat_filt5{i} = gf_3dt5.*kdat{i};

    %transform from the filtered frequency space to the now filtered time
    %space
    dat_space2{i} = ifftn(dat_filt2{i});
    dat2filt2{i} = dat_space2{i};

    dat_space5{i} = ifftn(dat_filt5{i});
    dat2filt5{i} = dat_space5{i};
    %flatten spatial data to find max
    dat_flat2 = reshape(dat2filt2{i},1,n^3);
    [xyz_max2, Imax2] = max(dat_flat2);

    dat_flat5 = reshape(dat2filt5{i},1,n^3);
    [xyz_max5, Imax5] = max(dat_flat5);

    maxval_xyz2{i} = xyz_max2;
    maxval_xyz5{i} = xyz_max5;
    % find the x y z coordinates the max for each time slice
    [xmax, ymax, zmax] = ind2sub(size(X),Imax2);
    Mx = X(xmax, ymax, zmax); My = Y(xmax, ymax, zmax);
    Mz = Z(xmax, ymax, zmax);
    marble_xyz2{i} = [Mx, My, Mz];

    [xmax, ymax, zmax] = ind2sub(size(X),Imax5);
    Mx = X(xmax, ymax, zmax); My = Y(xmax, ymax, zmax);
    Mz = Z(xmax, ymax, zmax);
    marble_xyz5{i} = [Mx, My, Mz];

end

% Plot the filtered spatial signalusing tau = 0.2
%showing the path of the marble
for j=1:20

    figure(3)
    isosurface(X,Y,Z, abs(dat_space2{j}),0.4)
    title('Filter Bandwidth Tau = 0.2')
    xlabel('X (units)')
    ylabel('Y (units)')
    zlabel('Z (units)')
    grid on;
    axis([-18 18 -18 18 -18 18]);
end

% Plot the filtered spatial signalusing tau = 0.5
%showing the path of the marble
for j=1:20

    figure(4)
    isosurface(X,Y,Z, abs(dat_space5{j}),0.3)
    title('Filter Bandwidth Tau = 0.5')
    xlabel('X (units)')
    ylabel('Y (units)')
    zlabel('Z (units)')
    grid on;
    axis([-18 18 -18 18 -18 18]);
end

% Plot the path of the marble using plot3 with 0.2 bandwidth filter
px = []; py = []; pz = [];
for k = 1:20

    % generate the vectors representing the marble location along each
    % dimension
    px(k) = marble_xyz2{k}(1);
    py(k) = marble_xyz2{k}(2);
    pz(k) = marble_xyz2{k}(3);

end

    figure(5)
    plot3(px,py,pz)
    grid on;
    xlabel('X (units)')
    ylabel('Y (units)')
    zlabel('Z (units)')
    axis([-15 15 -15 15 -15 15])
```