AMATH 582 Homework 1: An Ultrasound Problem

Eric A. Silk

***Abstract*—This paper serves to describe the process by which a noisy ultrasound can be cleaned and analyzed to determine the path of an foreign object within the intestines of a dog. Through the use statistical methods, Fast Fourier Transforms, spectrum averaging, and Gaussian filtering, highly noisy data can be resolved into a clear trajectory.**

***Index Terms*—Filter, Fourier, Gaussian, Histogram, Spectrum, Ultrasound**

I. Introduction

In the example proposed by Dr. Kutz, a dog has swallowed a marble and it is moving through its intestines. 20 ultrasound measurements were taken sequentially in time, producing three-dimensional matrices of the local magnitudes. Unfortunately, they are highly noisy. In order to determine the trajectory of the marble, filtering must be employed to remove the noise.

II. Theoretical Background

Ultrasounds produce spatial data, indicating the intensity of the reflection of an acoustic wave off of a surface at some distance. Objects produce reflections at characteristic frequencies, even as their physical position may change. This fact can be exploited to isolate a given object within a noisy environment and determine its trajectory through multiple measurements.

A. Fourier Transforms

Converting a temporal or spatial problem into the frequency domain can be achieved with a Fourier transform (Equation 1)

$$\hat{f}(\omega) = \int_{-\infty}^{\infty} f(x) e^{-2\pi j x \omega} dx \tag{1}$$

This assumes a given section of data is periodic and fully contained within the range of $-\pi$ to π. This requires the results be scaled appropriately (Equation 2).

$$\omega_{scaled} = \frac{\pi}{L} \omega_{nominal} \tag{2}$$

where the signal exists between in the region $[-L : L)$.

Given that processing is occurring on a computer, the Discrete Fourier Transform (Equation 3), or "DFT" is used – in particular, the Fast Fourier Transform (hereafter "FFT").

$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{j2\pi}{N} kn} \tag{3}$$

Eric Silk is a Masters Student in Applied Mathematics at the University of Washington, and a Research Engineer for Schweitzer Engineering Laboratories, Pullman, WA 99163 (email: esilk16@uw.edu, eric.silk@ericsilk.com)

The FFT works by recursively decomposing the transform into 2 smaller transforms until a prime length is reached (ideally 1) along some branch of the recursion, then shifting the results in a "butterfly" fashion. For this reason, FFT's of size 2^n are fastest, but any vector with small prime factors will perform reasonably quickly.

B. Noise

Here, noise is defined as the signal produced by a stochastic, stationary phenomena unrelated the primary signal of interest. These may include the random fluctuations of electrons within the measuring apparatus, movement of fluids within the dog's body, or the process of discretizing analog measurements.

Noise is often characterized by its distribution within the frequency spectrum. For instance, white noise has equal intensities across the whole spectrum, producing a signal with a finite variance and zero mean. Thus, while a single measurement of white noise may produce a fairly energetic spectrum, averaging multiple measurements will tend towards a zero energy spectrum.

Utilizing these pieces of information, the 20 provided measurements can be averaged in the Fourier domain to force any white noise present towards zero, while preserving the frequency components of the object of interest. Once these frequencies are determined, a filter can be applied to remove noise in individual measurements. Doing so will allow the trajectory of the marble to be determined.

C. Gaussian Filter

Gaussian filters are commonly used in image processing to reduce noise and smooth data, and are fairly straightforward to implement. As such, it was deemed an appropriate choice for the problem of filtering the data. They can be created in arbitrary dimensions, including 1D (Equation 4) and 3D (Equation 5)

$$G(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-x_\omega)^2}{2\sigma^2}} \tag{4}$$

$$G(x,y,z) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-x_\omega)^2+(y-y_\omega)^2+(z-z_\omega)^2}{2\sigma^2}} \tag{5}$$

x_ω, y_ω, and z_ω are the frequencies to shift the center of the kernel to. These equations assume the same variance in all dimensions.

III. Algorithm Implementation and Development

The first portion of work involved exploring the `isosurface` plots of each sample, which proved largely fruitless (Fig. 1). The data is highly noisy, and attempts

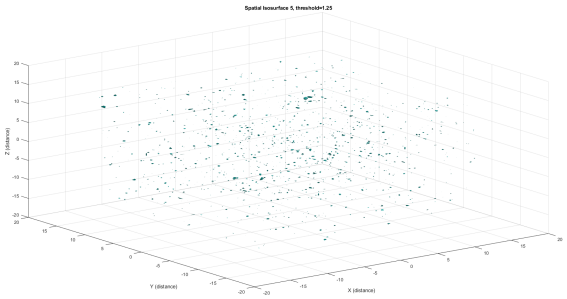

Fig. 1. Thresholded spatial isosurface, showing the clearly noisy data.

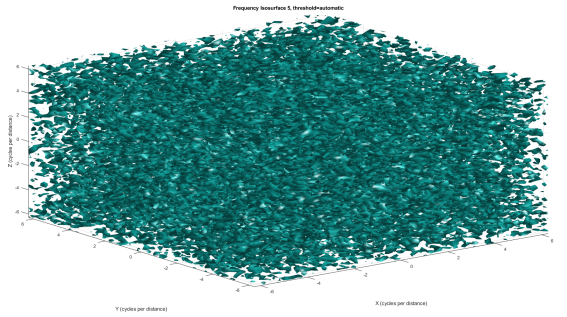

Fig. 2. Thresholded frequency isosurface, showing the clearly noisy data.

at increasing the threshold of the `isosurface` plot were ineffective in isolating the trajectory of the marble. Next, `fftn` was used to take a multidimensional FFT of each sample. The absolute value of the results of these were also plotted via `isosurface` (taking care to `fftshift` the data) with little success – the Fourier transform of noisy data produces a noisy spectrum. An example can be seen in Fig. 2.

A loop was used to sum each sample's spectrum together, before dividing by the number of samples to average the spectrums. This was done to take advantage of the previously mentioned zero mean of noise. In addition, while looping, a histogram of the absolute value of the spectrum was plotted (Fig. 3). Finally, a histogram of the resulting averaged spectrum was plotted to visually verify the shift in the spectrum (Fig. 4.

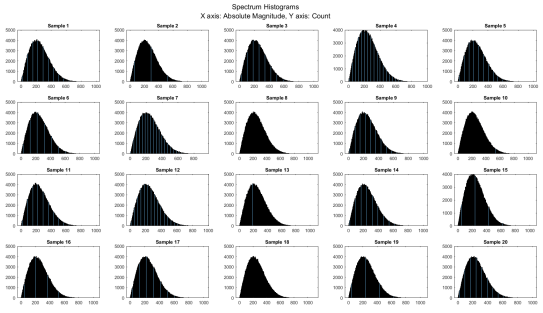

Fig. 3. Histograms of the frequency spectrum for each sample.

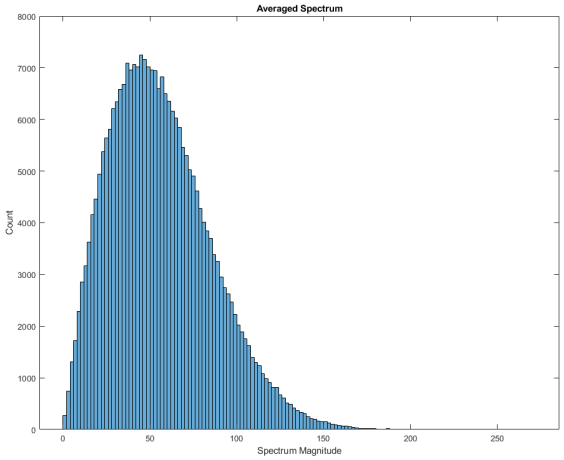

Fig. 4. Histograms of the frequency spectrum for each sample.

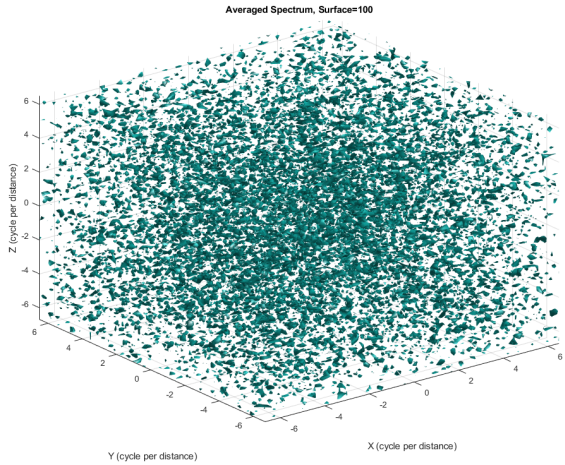

Fig. 5. Averaged spectrum, Threshold=100

This averaged spectrum was then plotted via `isosurface`, with increasing thresholds (informed by the distribution of the histogram). A more rigorous, fully automatic method (e.g. threshold on percentile) would be preferred in a production environment. From this a trend to exclude more and more noise can be seen visually (Figs. 5, 6, and 7).

Using the most stringent `isosurface` threshold, the plot was manually inspected to determine the location of the maxima of the whole spectrum. These values were used to set the guassian kernel offsets. The kernels standard deviation was selected somewhat arbitrarily as 1, but the results appeared satisfactory. An improvement for a production environment would involve determining the maximum location in the spectrum programatically, but for exploratory work this was deemed acceptable in the interim.

To apply the filter (Fig. 8), each spectrum was `fftshift`-ed and multiplied by the filter kernel, a process commonly known as convolution. This process could likely be optimized to be more performant if the volume of data was larger or there were specific latency requirements, but no preemptive

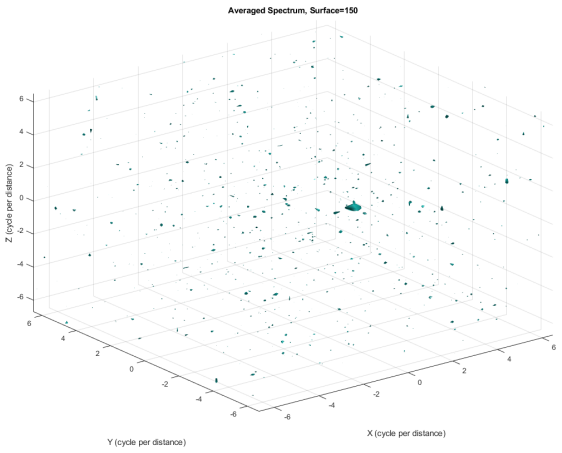

Fig. 6. Averaged spectrum, Threshold=150

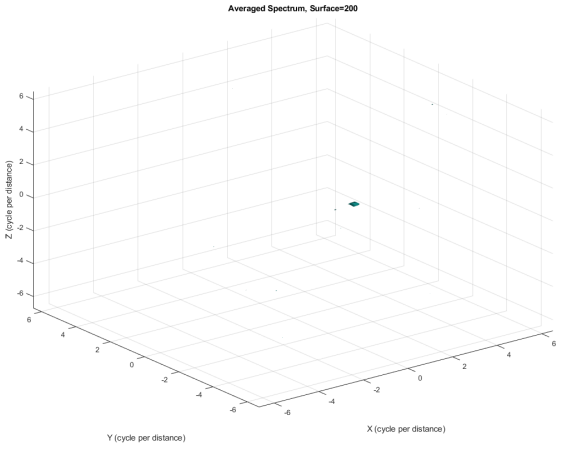

Fig. 7. Averaged spectrum, Threshold=200

attempts were made. The results were then again `fftshift`-ed and an inverse FFT was taken via `ifftn` to bring each filtered spectrum into the spatial domain.

Within each loop, another histogram was taken and plotted (Fig. 9) to determine verify the proper application of the filter and to determine an appropriate cutoff for `isosurface` plots.

Initially, each `isosurface` was plotted separately, but this was converted to plot each on the same figure in order to show the trajectory of the marble (Fig. 10). To indicate the temporal trend of each sample, each isosurface was shaded differently, going from light to dark in time. Note that in the code, due to differences in indexing between `meshgrid` and `ind2sub`, the X and Y mesh axes have been swapped to match the `plot3` trajectory.

Finally, per the requirements of the assignment, the peak of each filtered spatial sample was used to plot the trajectory with `plot3`, along with careful scaling of the indices (determined via `max` and `ind2sub`) to reflect the spatial domain (Fig. 11).

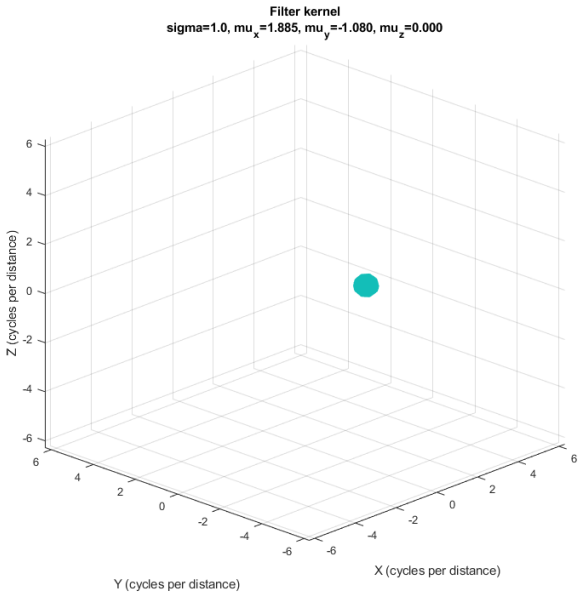

Fig. 8. Normalized Gaussian Filter Kernel, threshold=0.9/1.0

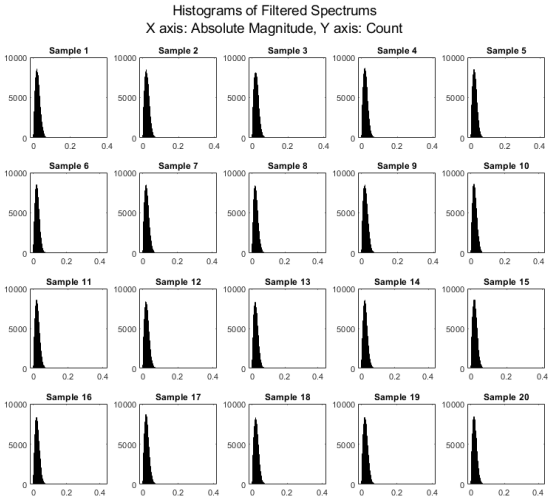

Fig. 9. Filtered Spectrum Histograms

IV. Computational Results

The center frequency of the filter was determined to be at:

$$\omega_x = 1.885 \tag{6}$$

$$\omega_y = -1.08 \tag{7}$$

$$\omega_z = 0.0 \tag{8}$$

cycles per unit distance.

Based upon this trajectory, the vet should direct a blast of energy at:

$$x = 4.688 \tag{9}$$

$$y = -5.156 \tag{10}$$

$$z = -5.625 \tag{11}$$

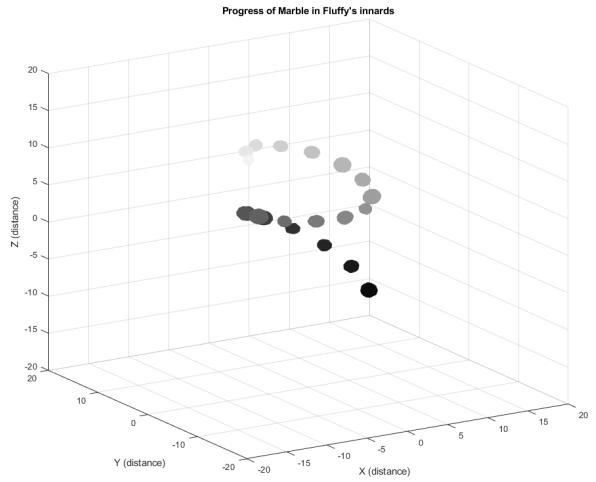

Fig. 10. Trajectory of the marble as an isosurface

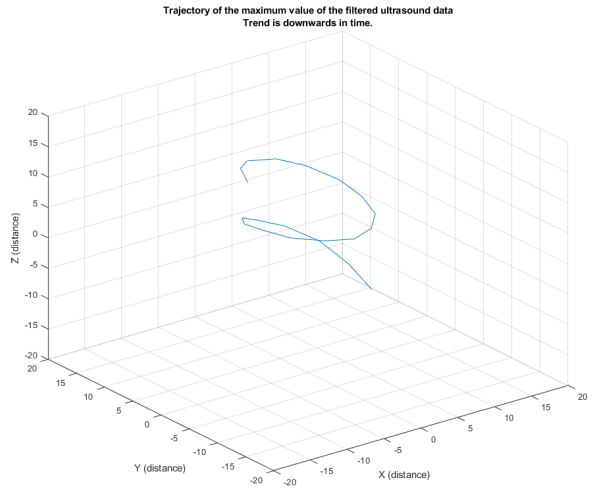

Fig. 11. Trajectory as determined by the maximum value of the filtered data

units of distance.

V. Summary and Conclusion

By taking advantage of the stochastic, stationary nature of noise and the power of the Fourier transform, one can resolve even highly noisy data into meaningful trends. At least, Fluffy appreciates it.

VI. Appendix A: Functions Used

A. `abs`

Given a scalar, vector, or N dimensional matrix, returns a structure of the same size with all elements set to the absolute value of the original structure. Produces all positive numbers when the input is purely real, and produces purely real numbers when the input is complex.

B. `fftn, ifftn`

The n-dimensional fast Fourier transform (and its inverse). The prior assumes data is non-shifted, whereas the latter assumes it has been. The prior returns shifted data, the latter returns unshifted.

C. `fftshift, ifftshift`

Because `fft` and `fftn` produce data that has been shifted, `fftshift` is provided to undo the operation. Similarly, because `ifft` and `ifftn` expect the data to be shifted, `ifftshift` was provided to re-shift the data in preparation. In 1D, the data is swapped about the midpoint; in 2D, the quadrants are swapped diagonally, and so on into higher dimensions.

D. `meshgrid`

When given a pair of vectors containing cooordinates, returns a 2D grid coordinates for plotting 3D images (such as surface plots). When given 3 vectors, returns 3D grid coordinates for 4D plots (such as isosurface).

E. `isosurface`

Given a 3D mesh grid coordinates, a 3D matrix of magnitudes at each point defined in the grid coordinates, and a threshold value, produces a plot showing the surface(s) that connects all points equal to the threshold within the 3D space. If no threshold is provided, it determines one automatically. Analogous to a contour map, but in higher dimensions.

F. `histogram`

Given a vector or N-dimensional matrix, plots the distribution of the data after sorting it into bins. That is, provides a bar chart of the count of each number of elements that fit within a range (or ”bin”).

G. `sprintf`

Provides a C-like function for string formatting. Useful for programattically defining plot titles or messages to display.

H. `disp`

Displays its arguments to the command line. Useful for printing messages and variable values, preferred to eschewing semicolon suppression.

I. `patch`

Creates a graphics object composed of one or more polygons. Used for displaying more advanced plots or arbitrary shapes/objects.

J. `isonormals`

Similar to `isosurface`, but accepts a `patch` object and gives greater control over the appearance of the resulting plot.

K. `max`

Given a vector, returns the largest value and its location. For higher dimensional data, returns a subspace containing the highest value as a vector. Often most easily used when flattening a higher dimensional structure and using in conjunction with `ind2sub`.

L. `ind2sub`

Given a linear index and the shape of a data structure, converts from a linear index to the N-dimensional indices.

M. `plot3`

Given 3 vectors of positions in time (x, y, and z), produce a 3D plot of the path.

N. *Plotting Functions*

Functions, such as `figure`, `axis`, `subplot`, etc., are deemed trivially understood/searched for and thus not explained here.

VII. Appendix B: Matlab Code

See my Github for the full repository, including this source code for this IEEE template Latex document.

```
clear all; close all; clc;
% Configuration Constants
% Note that plotting all isosurfaces will take considerable
% time -- 5 minutes is not unrealistic.
% If PLOT_ALL_*_SURFACES is set to false, the user must
% press enter to continue after each plot
%
% Woe to ye with shoddy cooling
PLOT_TIME_ISOSURFACE = false;
PLOT_ALL_TIME_SURFACES = false;
PLOT_FREQ_ISOSURFACE = false;
PLOT_ALL_FREQ_SURFACES = false;

load Testdata
SAMPLES = 20;
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); % Seems like an odd way but ok
[X,Y,Z]=meshgrid(x,y,z);
[Kx,Ky,Kz]=meshgrid(ks,ks,ks);

% Time Domain
if PLOT_TIME_ISOSURFACE
    for j=1:SAMPLES
        figure();
        Un(:,:,:)=reshape(Undata(j,:),n,n,n);
        isosurface(X,Y,Z,abs(Un),1.25)
        axis([-20 20 -20 20 -20 20]), grid on;
        if ~(PLOT_ALL_TIME_SURFACES)
            drawnow;
            input("Press ENTER to continue...");
            close all;
        end
        title(sprintf("Spatial Isosurface %u, threshold=1.25", j))
        xlabel("X (distance)")
        ylabel("Y (distance)")
        zlabel("Z (distance)")
    end
    drawnow;
end

% Frequency Domain
if PLOT_FREQ_ISOSURFACE
    for j=1:SAMPLES
        figure();
        Un(:,:,:)=reshape(Undata(j,:),n,n,n);
        Un_fft=fftn(Un);
        isosurface(Kx,Ky,Kz,abs(fftshift(Un_fft)))
```

```
52          axis([-2*pi 2*pi -2*pi 2*pi -2*pi 2*pi]), grid on;
53          if ~(PLOT_ALL_FREQ_SURFACES)
54              drawnow;
55              input("Press ENTER to contiue...");
56              close all;
57          end
58          title(sprintf("Frequency Isosurface %u, threshold=automatic", j))
59          xlabel("X (cycles per distance)")
60          ylabel("Y (cycles per distance)")
61          zlabel("Z (cycles per distance)")
62      end
63      drawnow;
64  end
65
66  averaged_spectrum = zeros(n, n, n);
67  figure
68  % Plot histograms of the frequency magnitudes while summing
69  % all the spectrums
70  % This will inform our cutoff for the isosurface
71  for j=1:SAMPLES
72      tmp(:,:,:) = reshape(Undata(j, :), n, n, n);
73      fft_tmp = fftn(tmp);
74      subplot(4,5,j)
75      histogram(abs(fft_tmp))
76      title(sprintf("Sample %d", j))
77      averaged_spectrum = averaged_spectrum + fft_tmp;
78  end
79  sgtitle({"Spectrum Histograms",...
80           "X axis: Absolute Magnitude, Y axis: Count"})
81
82  % Now get the average instead of just sum
83  averaged_spectrum = abs(averaged_spectrum) / SAMPLES;
84  figure
85  histogram(averaged_spectrum)
86  xlabel("Spectrum Magnitude")
87  ylabel("Count")
88  title("Averaged Spectrum")
89
90  plot_spectrum_isosurface(Kx, Ky, Kz, averaged_spectrum, 100);
91  title("Averaged Spectrum, Surface=100")
92  plot_spectrum_isosurface(Kx, Ky, Kz, averaged_spectrum, 125);
93  title("Averaged Spectrum, Surface=125")
94  plot_spectrum_isosurface(Kx, Ky, Kz, averaged_spectrum, 150);
95  title("Averaged Spectrum, Surface=150")
96  plot_spectrum_isosurface(Kx, Ky, Kz, averaged_spectrum, 175);
97  title("Averaged Spectrum, Surface=175")
98  plot_spectrum_isosurface(Kx, Ky, Kz, averaged_spectrum, 200);
99  title("Averaged Spectrum, Surface=200")
100 plot_spectrum_isosurface(Kx, Ky, Kz, averaged_spectrum, 250);
101 title("Averaged Spectrum, Surface=250")
102
```

```
103 % Generate a 3D gaussian centered about x=1.885, y=-1.08, z=0
104 sigma_x = 1;
105 sigma_y = 1;
106 sigma_z = 1;
107 mu_x = 1.885;
108 mu_y = -1.08;
109 mu_z = 0;
110
111 coeff = 1/(2*pi*sigma_x*sigma_y*sigma_z);
112 exponent = (Kx-mu_x).^2/(2*sigma_x^2) + (Ky-mu_y).^2/(2*sigma_y^2) + (Kz-mu_z).^2/↙
(2*sigma_z^2);
113 gauss_filter = coeff*exp(-exponent);
114 % normalize it
115 gauss_filter = gauss_filter/(max(gauss_filter(:)));
116 figure();
117 title(sprintf("Filter kernel\nsigma=%.1f, mu_x=%.3f, mu_y=%.3f, mu_z=%.3f", sigma_x,↙
mu_x, mu_y, mu_z));
118 isosurface(Kx, Ky, Kz, gauss_filter, 0.9);
119 axis([-2*pi 2*pi -2*pi 2*pi -2*pi 2*pi]);
120 xlabel("X (cycles per distance)")
121 ylabel("Y (cycles per distance)")
122 zlabel("Z (cycles per distance)")
123 grid on, drawnow
124
125 figure();
126 for j=1:20
127     tmp(:,:,:) = reshape(Undata(j, :), n, n, n);
128     fft_tmp = fftshift(fftn(tmp));
129     filtered = fft_tmp.*gauss_filter;
130     filtered_time = ifftshift(ifftn(filtered));
131     subplot(4,5,j);
132     histogram(abs(filtered_time));
133     title(sprintf("Sample %d", j))
134 end
135 sgtitle({"Histograms of Filtered Spectrums",...
136          "X axis: Absolute Magnitude, Y axis: Count"})
137 % Colors, dark to light for time
138 figure();
139 title("Progress of Marble in Fluffy's innards")
140 xlabel("X (distance)")
141 ylabel("Y (distance)")
142 zlabel("Z (distance)")
143 hold on;
144
145 num = SAMPLES;
146 den = num + 1;
147 init_color = num / den;
148 color_iter = -[1/den 1/den 1/den];
149 facecolor = [init_color init_color init_color];
150 for j=1:SAMPLES
151     tmp(:,:,:) = reshape(Undata(j, :), n, n, n);
```

```
152     fft_tmp = fftshift(fftn(tmp));
153     filtered = fft_tmp .* gauss_filter;
154     filtered_time = ifftn(ifftshift(filtered));
155     % Swap the X and Y to account for the differences in indexing
156     % between meshgrid and ind2sub
157     hiso = patch(isosurface(Y, X, Z, abs(filtered_time), 0.35),...
158                  'FaceColor', facecolor,...
159                  'EdgeColor', 'none');
160     isonormals(abs(filtered_time), hiso);
161     facecolor = facecolor + color_iter;
162 end
163 axis([-20 20 -20 20 -20 20]);
164 grid on;
165
166 % Now lets do the plot3 trajectory plot
167 x3d = zeros(1, SAMPLES);
168 y3d = zeros(1, SAMPLES);
169 z3d = zeros(1, SAMPLES);
170 for j=1:SAMPLES
171     tmp(:,:,:) = reshape(Undata(j, :), n, n, n);
172     fft_tmp = fftshift(fftn(tmp));
173     filtered = fft_tmp .* gauss_filter;
174     filtered_time = abs(ifftn(fftshift(filtered)));
175     [max_val, max_index] = max(filtered_time(:));
176     [indx, indy, indz] = ind2sub(size(filtered_time), max_index);
177     if not (max(filtered_time(:)) == filtered_time(indx, indy, indz))
178         disp("Ya done messed up, A-A-ron!")
179     end
180     x3d(j) = indx;
181     y3d(j) = indy;
182     z3d(j) = indz;
183 end
184
185 % scale indices to spatial domain
186 x3d = (x3d-32)*(2*L)/n;
187 y3d = (y3d-32)*(2*L)/n;
188 z3d = (z3d-32)*(2*L)/n;
189
190 figure
191 plot3(x3d, y3d, z3d)
192 axis([-20 20 -20 20 -20 20])
193 grid on;
194 xlabel("X (distance)")
195 ylabel("Y (distance)")
196 zlabel("Z (distance)")
197 title({"Trajectory of the maximum value of the filtered ultrasound data"...
198         "Trend is downwards in time."})
199 disp(sprintf("Direct the blast at (%u, %u, %u).",...
200     x3d(SAMPLES), y3d(SAMPLES), z3d(SAMPLES)))
```

```
function plot_spectrum_isosurface(X, Y, Z, unshifted_fft, mag)
    % User must provide titles!
    figure();
    isosurface(X, Y, Z, fftshift(unshifted_fft), mag);
    axis([min(X(:)) max(X(:)) min(Y(:)) max(Y(:)) min(Z(:)) max(Z(:))]);
    xlabel("X (cycle per distance)")
    ylabel("Y (cycle per distance)")
    zlabel("Z (cycle per distance)")
    grid on
    drawnow
end
```