Saving Mr. Fluffy: A Mathematical Approach

Kuan H. Chen
Chemical Engineering
Applied & Computational Mathematical Sciences

Abstract

Mr. Fluffy is a dog who swallowed a marble that is hazardous to its wellbeing. Using ultrasound techniques, signals of the marble provide insight into the position of it in Mr. Fluffy's intestines. However, noise interrupts these signals and must be cleaned. By using signal-averaging techniques, the Fast Fourier Transformation (FFT) algorithm, and filters, the path of the marble can be found, and its future trajectory predicted.

Keywords: signal-averaging, FFT, characteristic frequency, filter, path, trajectory

Saving Mr. Fluffy: A Mathematical Approach

I. Introduction

Traveling waves are commonly used by scientists and animals to detect the presence of various objects. Sonar uses sound waves to detect the presence of fish; radars use electromagnetic waves to detect the presence of aircrafts; bats use echolocation to detect the presence of prey. In this report, wave data collected by ultrasound used to detect the position of a marble in the intestines of a dog named Fluffy is analyzed. As a result of flowing body-fluids, the ultrasound data is riddled with "noise" – signals that result from various sources that interrupt those coming from the target. In order to locate the marble at various time points and predict its trajectory accurately, noise must be cleaned to reveal the signals coming from the marble. Failure to do so will result in the demise of Mr. Fluffy as the marble is expected to be hazardous to his livelihood. The following sections explore signal-averaging techniques used to clean data, the Fast Fourier Transformation (FFT) and inverse FFT algorithm used for processing signals, and an interpretation of the path and future trajectory of the marble.

II. Theoretical Background

1. **Signal-averaging**:

   In a typical ultrasound operation, a wave of a known characteristic frequency is sent into the body to detect the presence of objects. However, all ultrasound signal detections come with noise. Assuming that the variations in magnitude of the noise is completely random and follow a normal (Gaussian) distribution, one can expect that the noise will cancel each other after infinite additions to itself according to the expectation value of the normal distribution function shown in Equation 1.

$$\mathbf{1}.\int_{-\infty}^{\infty} e^{-x^2} x\, dx = 0$$

   Thus, by overlaying all signals, dominant signals of various frequencies will emerge, including the characteristic frequency emitted by the ultrasound equipment.

2. **Fourier Transformation**:

   The Fourier Transformation is used to decompose a function in the spatial domain into the frequencies that make it up, the formula for which is shown in Equation 2.

$$\mathbf{2}.\ F(k_x, k_y, k_z) = \iiint_{-\infty}^{\infty} f(x,y,z) e^{-i(k_x x + k_y y + k_z z)}\, dxdydz$$

   After the transformation, the function becomes one that is dependent on frequencies instead of space. In the frequency domain, the magnitude of signals represents the amount of detected signals at a given frequency. In this context, it is useful in assisting one to find the characteristic frequency at which waves are being emitted by the ultrasound equipment. A dominant signal in the frequency domain will correspond to the characteristic frequency.

3. **Characteristic frequency**:

   In the frequency domain, one is expected to find a dominating signal that has a greater magnitude than those created as noise. This signal will correspond to the characteristic frequency since it reoccurs frequently as it is emitted by the equipment. Only the data in the close proximity of this frequency matters for the observer since it provides information regarding the target object.

4. **Filter**:

   Filters are applied to the signals in the frequency domain in to clean out unwanted data that results from noise. A commonly used filter is given by the Gaussian curve and is represented as Equation 3 in 3-D,

$$\mathbf{3}.\, f(\tau, k_x, k_y, k_z) = e^{-\tau((k_x-k_{x0})^2+(k_y-k_{y0})^2+(k_z-k_{z0})^2))}$$

   where τ represents the width of the "window" in which one might determine is useful data; k_x, k_y, k_z are points in the frequency domain and k_{x0}, k_{y0}, k_{z0} are the points in the frequency domain that correspond to the characteristic frequency. Upon application of this filter, signals that are outside of the "window" are reduced to 0, greatly reducing the amount of noise left.

5. **Inverse Fourier Transformation**:

   The inverse Fourier Transformation takes data in the frequency domain and transforms it back into the spatial domain. It gives one information about where certain frequencies occur, the formula for which is shown in Equation 3.

$$\mathbf{3}.\ f(x,y,z) = \frac{1}{2\pi}\iiint_{-\infty}^{\infty} F(k_x,k_y,k_z)e^{i(k_x x+k_y y+k_z z)}dk_x dk_y\, dk_z$$

   The inverse Fourier Transformation will be useful in determining where the target object is detected. After filtering the signals in the frequency domain, the remaining data can be inverse Fourier Transformed to yield the time and spatial variations of the signals of the object, yielding a well-defined path to which the object follows.

III. Algorithm Implementation and Development

The following MATLAB code is used to define the spatial and frequency domains on which analysis will be carried out:

```
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);
```

The raw data provided represents 4-D wave signals whose magnitude depends on 3 spatial components, x, y, and z, and can be visualized as isosurfaces in 3-D, as seen in Figure 1.

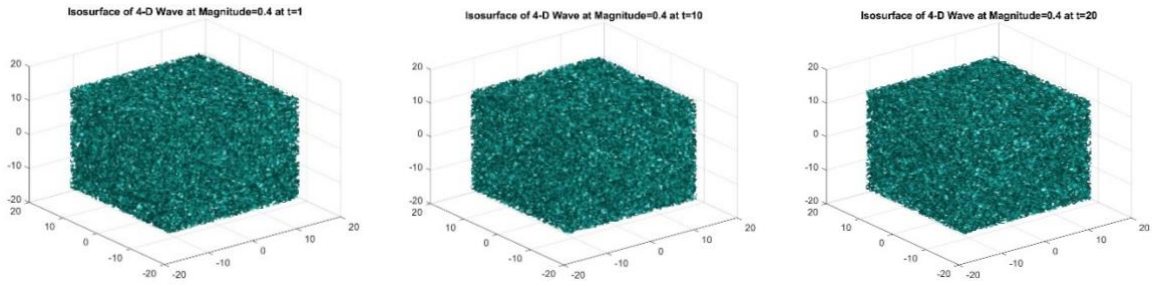

Figure 1. 4-D wave signals visualized as 3-D iso-surfaces of magnitude 0.4 at time 1, 10, 20

This data contains unwanted noise as well as signals at the characteristic frequency. However, due to the sheer size of this data and difficulty to visualize it, it is not apparent to the observer which frequencies are dominant. Thus, data cleaning is eminent.

1. **Finding the characteristic frequency and filter:**

   The raw data from ultrasound is stored in the vector `Undata`, which is reshaped into a 3-D "cube" of data, `Un`. Noise is significantly reduced by adding `Un` at various time points on top of each other with the following MATLAB commands,

```
for i = 1:20
  Un(:,:,:)=reshape(Undata(i,:),n,n,n);
  UnAve = UnAve + Un;
end
```

   with `UnAve` representing the time-averaged data. The 4-D wave signals can then be transformed into the frequency domain using the FFT algorithm by the following MATALB command,

```
UntAve = fftn(UnAve);
```

   With `UntAve` representing the transformed time-averaged data. Although the frequencies of much noise is retained, the most frequently occurring one, i.e., the characteristic frequency, will have the highest magnitude. It can be found using a simple search method with the following MATLAB commands,

```
[UntMax, Ik] = max(abs(UntAve(:)));
charFreq = [Kx(Ik), Ky(Ik), Kz(Ik)];
```

   where `charFreq = [-4.8171,5.6549,-6.7021]` is the position of the characteristic wave in the frequency domain. Assuming a τ value of 0.2, the filter needed can be defined as in Equation 4,

$$\mathbf{4.}\ f(\tau, k_x, k_y, k_z) = e^{-0.2((k_x+4.8171)^2+(k_y-5.6549)^2+(k_z+6.7021)^2))}$$

   and represented as follows in MATALB:

```
filter = @(tau)...
    exp(-1*tau*( ((Kx-charFreq(1)).^2)...
    +((Ky-charFreq(2)).^2)...
    +((Kz-charFreq(3)).^2)));
```

2. **Determining the path of the marble:**

   The raw data, `Un`, collected from the ultrasound must be Fourier Transformed before the filter may be applied; upon application, all noise outside of the designated "window" value, τ, will be reduced to zero, isolating the signal corresponding to the characteristic frequency. The position of the marble in the first discrete point of time can then be found by applying the inverse Fourier Transformation. This series of steps can be achieved by the following MATLAB commands,

```
Unt = fftn(Un);
Untf = filter(0.2).*Unt;
Unf = ifftn(Untf);
```

   with `Unt`, `Untf`, and `Unf` each representing the Fourier Transformed `Un`, the filtered `Unt`, and the inverse Fourier Transformed `Untf`. This reveals that the marble is positioned at `[x,y,z]=[4.6875,-4.6875,9.8438]` at the first discrete time point. By repeating this process with a `for` loop over the 20 discrete time points, the path of the marble can be visualized in a 3-D plot, as discussed in the following section.

IV. Computational Results

Plotting the 20 spatial positions of the marble found using the previously discussed methods yields the red line graphs presented in Figure 2 a – d. The final recorded position of the marble is at `[x,y,z]=[-5.625,4.21875,-6.09375]`. As one can see, the path is shaped like a helix, and can be represented by periodic functions such as sine and cosine.

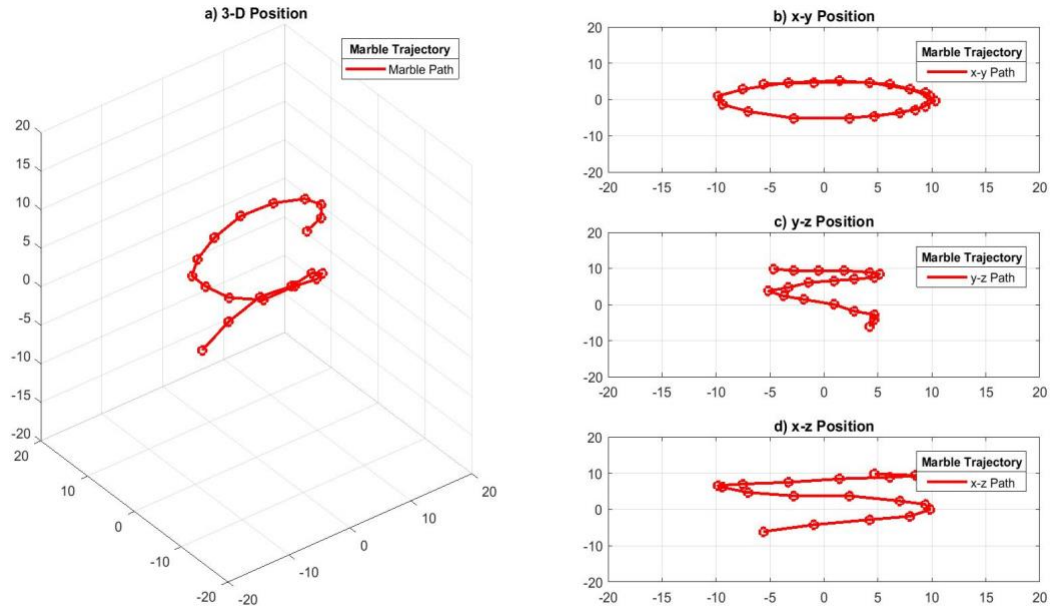

Figure 3. The path of the marble in 3-D, x-y plane, y-z plane, and x-z plane

The blues curves in Figure 4 present the path of the marble in the x, y, and z directions with respect to time; Equations 5, 6, and 7 each closely model these paths, respectively, and are presented as the orange curves in the figure.

$$\mathbf{5}. \quad 10\sin(0.5t)$$

$$\mathbf{6}. \quad 5\sin(0.5t - 1.5)$$

$$\mathbf{7}. \quad 10\cos(0.1t)$$

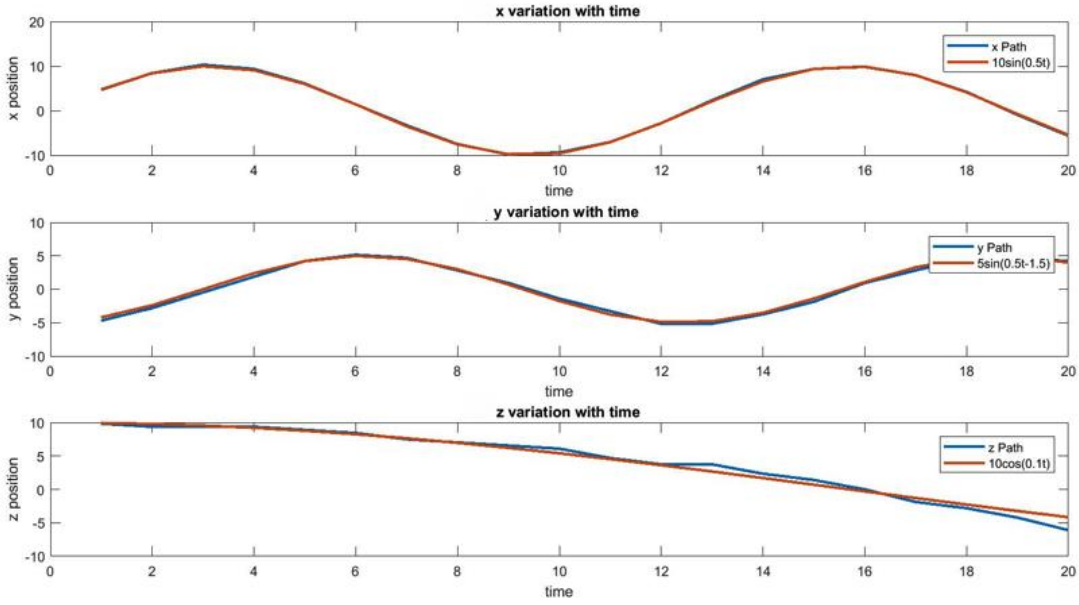

Figure 4. The path and models of the marble in the x, y, and z directions

Equations 5, 6, and 7 can be used to predict the trajectory of the marble as time continues to progress. The blue curves in Figure 5 a – d show the expected behavior of the marble in the next 20 discrete time points. The coordinates of these data points in space are attached in Appendix C.

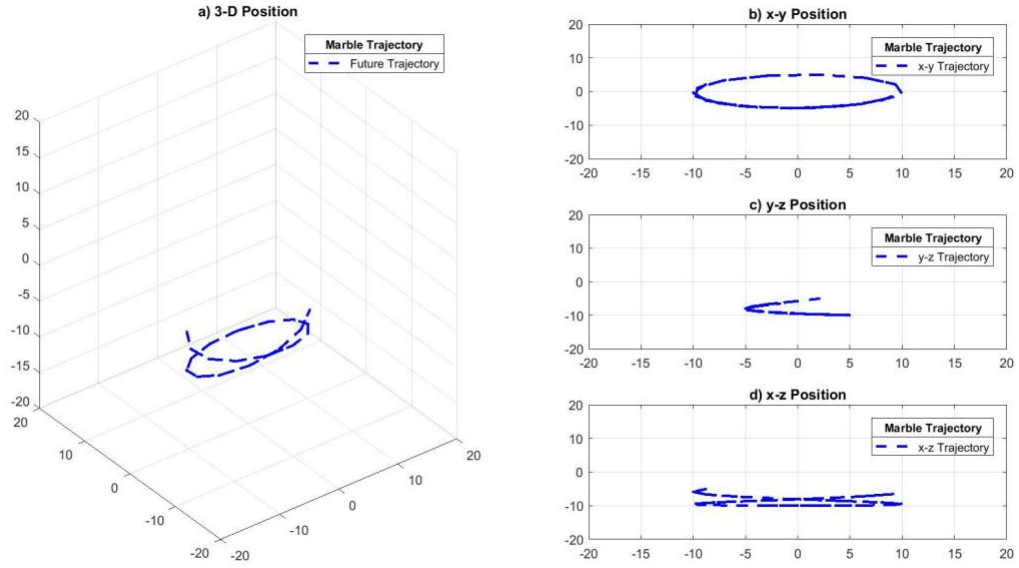

Figure 5. The expected behavior of the marble in the next 20 discrete time points

V. Summary and Conclusions

By mapping the spatial coordinates provided in this analysis to Fluffy's body, the marble can be located in his body. An intense acoustic wave can then be focused on the part of Fluffy's body that corresponds to the position, `[x,y,z]=[-5.625,4.21875,-6.09375]` to break down the marble and save him. In the case when one fails to do so in time, the acoustic wave can be instead focused on positions predicted by Equations 5, 6, and 7. Figure 6 presents the marble's path in Fluffy's body at 40 discrete time points; the first 20 representing the path already traveled and the latter 20 representing the path that the marble is projected to take. Provided with this information, Fluffy will be given the opportunity to tell the world about the power of data analysis!

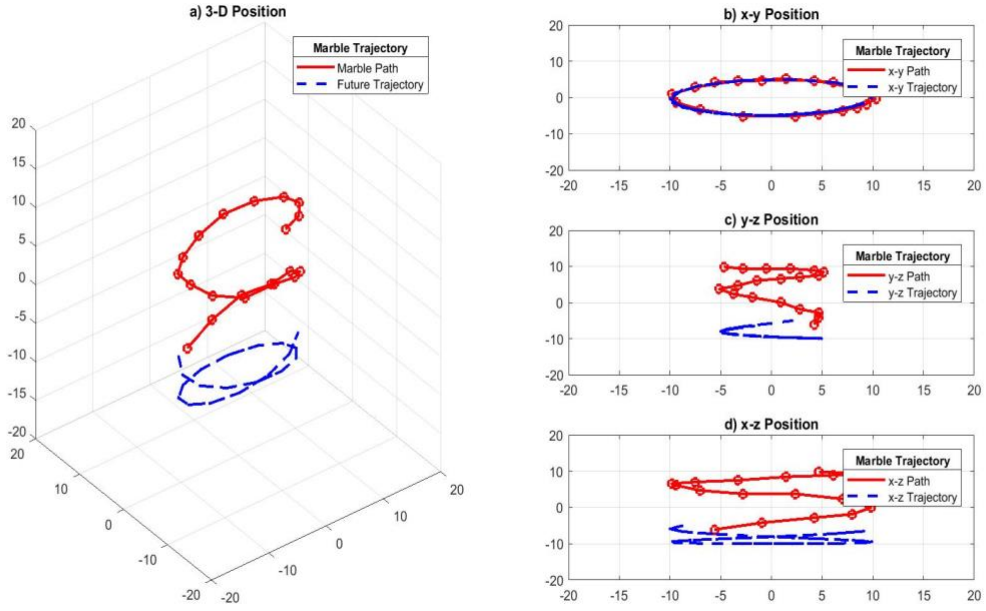

Figure 6. The path of the marble in 40 discrete time points

Appendix A

The following are the MATLAB functions that were used to complete this report:

- `Un=reshape(Undata,n,n,n) % reshapes the 1-D array of data, Undata, into a "cube" of data with size n*n*n.`
- `[X,Y,Z]=meshgrid(x,y,z) % reshapes coordinates defined by x,y,z into 3-D grid coordinates.`
- `fftn(UnAve) % performs the Fast Fourier Transformation algorithm on the set of data, UnAve, with n dimensions where n>=3.`
- `[UntMax, Ik]=max(abs(UntAve(:))) % returns the largest data point, UntMax, in the set of absolute values in UntAve(:) with its index, Ik.`
- `Unf=ifftn(Untf) % performs the inverse Fourier Transformation algorithm on the set of data, Untf, where n >=3.`

Appendix B

```
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);

UnAve = zeros(n,n,n);
for i = 1:20
  Un(:,:,:)=reshape(Undata(i,:),n,n,n);
  UnAve = UnAve + Un;
end

UntAve = fftn(UnAve);

[UntMax, Ik] = max(abs(UntAve(:)));
charFreq = [Kx(Ik), Ky(Ik), Kz(Ik)];
filter = @(tau)...
    exp(-1*tau*( ((Kx-charFreq(1)).^2)...
    +((Ky-charFreq(2)).^2)...
    +((Kz-charFreq(3)).^2)));
xPath = [];
yPath = [];
zPath = [];
for j=1:20
Un(:,:,:)=reshape(Undata(j,:),n,n,n);
Unt = fftn(Un);
Untf = filter(0.2).*Unt;
Unf = ifftn(Untf);

[UnMax, Ixyz] = max(abs(Unf(:)));
xPath = [xPath X(Ixyz)];
yPath = [yPath Y(Ixyz)];
zPath = [zPath Z(Ixyz)];
if j > 1
    % 3-D Plot
    subplot(3,2,[1,3,5]);
    title('a) 3-D Position')
    %title(strcat('3-D Position at t=',num2str(j)))
    plot3([xPath(end-1), xPath(end)],...
    [yPath(end-1), yPath(end)],...
    [zPath(end-1), zPath(end)],'ro-','linewidth',2); hold on
    axis([-20 20 -20 20 -20 20]), grid on
    % x-y Plot
    subplot(3,2,2)
    title('b) x-y Position')
    %title(strcat('x-y Position at t=',num2str(j)))
              plot([xPath(end-1),xPath(end)],[yPath(end-1),yPath(end)],'ro-
','linewidth',2);hold on
```

```
    axis([-20 20 -20 20]), grid on
    % y-z Plot
    subplot(3,2,4)
    title('c) y-z Position')
    %title(strcat('y-z Position at t=',num2str(j)))
            plot([yPath(end-1),yPath(end)],[zPath(end-1),zPath(end)],'ro-',
'linewidth',2);hold on
    axis([-20 20 -20 20]), grid on
    % x-z Plot
    subplot(3,2,6)
    title('d) x-z Position')
    %title(strcat('x-z Position at t=', num2str(j)))
              plot([xPath(end-1),xPath(end)],[zPath(end-1),zPath(end)],'ro-
','linewidth',2);hold on
    axis([-20 20 -20 20]), grid on
end

pause(0.1)
end
%pause
subplot(3,2,[1,3,5]);
past          =          plot3(xPath,yPath,zPath,'r','displayname','Marble
Path','linewidth',2);

subplot(3,2,2)
xyPlot(1)             =            plot(xPath,yPath,'r-','displayname','x-y
Path','linewidth',2);hold on

subplot(3,2,4)
yzPlot(1)             =            plot(yPath,zPath,'r-','displayname','y-z
Path','linewidth',2);hold on

subplot(3,2,6)
xzPlot(1)             =            plot(xPath,zPath,'r-','displayname','x-z
Path','linewidth',2);hold on

marbleTrajectory = @(t) [10*sin(0.5*t);
                         5*sin(0.5*t-1.5);
                         10*cos(0.1*t)];
trajectoryX = [];
trajectoryY = [];
trajectoryZ = [];
marbleTrajectory = @(t) [10*sin(0.5*t);
                         5*sin(0.5*t-1.5);
                         10*cos(0.1*t)];
for i = j:40
   next = marbleTrajectory(i);
   trajectoryX = [trajectoryX next(1)];
   trajectoryY = [trajectoryY next(2)];
   trajectoryZ = [trajectoryZ next(3)];
   if i > j+1
    % 3D plot
```

```
    subplot(3,2,[1,3,5])
    title('a) 3-D Position')
    %title(strcat('3-D Position at t=', num2str(i)))
    plot3([trajectoryX(end-1), trajectoryX(end)],...
    [trajectoryY(end-1), trajectoryY(end)],...
    [trajectoryZ(end-1), trajectoryZ(end)],'b.--','linewidth',2); hold on
    axis([-20 20 -20 20 -20 20]), grid on
    % x-y plot
    subplot(3,2,2)
    title('b) x-y Position')
    %title(strcat('x-y Position at t=', num2str(i)))
               plot([trajectoryX(end-1),trajectoryX(end)],[trajectoryY(end-
1),trajectoryY(end)],'b.--','linewidth',2); hold on
    axis([-20 20 -20 20]), grid on
    % y-z plot
    subplot(3,2,4)
    title('c) y-z Position')
    %title(strcat('y-z Position at t=', num2str(i)))
               plot([trajectoryY(end-1),trajectoryY(end)],[trajectoryZ(end-
1),trajectoryZ(end)],'b.--','linewidth',2); hold on
    axis([-20 20 -20 20]), grid on
    % x-z plot
    subplot(3,2,6)
    title('d) x-z Position')
    %title(strcat('x-z Position at t=', num2str(i)))
               plot([trajectoryX(end-1),trajectoryX(end)],[trajectoryZ(end-
1),trajectoryZ(end)],'b.--','linewidth',2); hold on
    axis([-20 20 -20 20]), grid on
   end
    pause(0.1)
end
subplot(3,2,[1,3,5]);
future     =     plot3(trajectoryX,     trajectoryY,     trajectoryZ,'b--
','displayname','Future Trajectory', 'linewidth', 2);

subplot(3,2,2)
xyPlot(2)      =      plot(trajectoryX,trajectoryY,'b--','displayname','x-y
Trajectory','linewidth', 2); hold on

subplot(3,2,4)
yzPlot(2)      =      plot(trajectoryY,trajectoryZ,'b--','displayname','y-z
Trajectory','linewidth', 2); hold on

subplot(3,2,6)
xzPlot(2)      =      plot(trajectoryX,trajectoryZ,'b--','displayname','x-z
Trajectory','linewidth', 2); hold on

lgdThreeD = legend([past future],'Location','Northeast');
lgdXYPlot = legend(xyPlot,'Location','Northeast');
lgdYZPlot = legend(yzPlot,'Location','Northeast');
lgdXZPlot = legend(xzPlot,'Location','Northeast');
title(lgdThreeD, 'Marble Trajectory')
```

```
title(lgdXYPlot, 'Marble Trajectory')
title(lgdYZPlot, 'Marble Trajectory')
title(lgdXZPlot, 'Marble Trajectory')

pause
close all
tspan = linspace(1,j,j);
subplot(3,1,1)
plot(tspan, xPath,'linewidth',2); hold on
plot(tspan, 10*sin(0.5*tspan),'linewidth',2)
title('e) x variation with time')
xlabel('time')
ylabel('x position')
legend('x Path', '10sin(0.5t)')

subplot(3,1,2)
plot(tspan, yPath,'linewidth',2); hold on
plot(tspan, 5*sin(0.5*tspan-1.5),'linewidth',2)
title('f) y variation with time')

xlabel('time')
ylabel('y position')
legend('y Path', '5sin(0.5t-1.5)')

subplot(3,1,3)
plot(tspan, zPath,'linewidth',2); hold on
plot(tspan, 10*cos(0.1*tspan),'linewidth',2)
title('g) z variation with time')

xlabel('time')
ylabel('z position')
legend('z Path','10cos(0.1t)')
```

Appendix C

	Path of marble traveled		
time	xpath	ypath	zpath
1	4.6875	-4.6875	9.84375
2	8.4375	-2.8125	9.375
3	10.3125	-0.46875	9.375
4	9.375	1.875	9.375
5	6.09375	4.21875	8.90625
6	1.40625	5.15625	8.4375
7	-3.28125	4.6875	7.5
8	-7.5	2.8125	7.03125
9	-9.84375	0.9375	6.5625
10	-9.375	-1.40625	6.09375
11	-7.03125	-3.28125	4.6875
12	-2.8125	-5.15625	3.75
13	2.34375	-5.15625	3.75
14	7.03125	-3.75	2.34375
15	9.375	-1.875	1.40625
16	9.84375	0.9375	0
17	7.96875	2.8125	-1.875
18	4.21875	4.6875	-2.8125
19	-0.9375	4.6875	-4.21875
20	-5.625	4.21875	-6.09375
	Future trajectory of marble		
	xtrajectory	ytrajectory	ztrajectory
21	-5.440211	3.992436	-4.161468
22	-8.796958	2.060592	-5.048461
23	-9.999902	-0.375756	-5.885011
24	-8.754522	-2.720106	-6.66276
25	-5.365729	-4.398479	-7.373937
26	-0.663219	-4.999951	-8.011436
27	4.20167	-4.377261	-8.568888
28	8.037844	-2.682865	-9.040721
29	9.906074	-0.331609	-9.422223
30	9.348951	2.100835	-9.709582
31	6.502878	4.018922	-9.899925
32	2.064675	4.953037	-9.991352
33	-2.879033	4.674475	-9.982948
34	-7.117853	3.251439	-9.874798
35	-9.613975	1.032337	-9.667982
36	-9.75626	-1.439517	-9.364567
37	-7.509872	-3.558927	-8.967584
38	-3.424806	-4.806987	-8.481
39	1.498772	-4.87813	-7.909677

40	6.055399	-3.754936	-7.259323