AMATH 482 Homework 1

Gg Tran / Seattle, WA

Abstract

Fluffy the Dog swallowed a marble. Using ultrasound, the vet obtained the data of the spatial variations in Fluffy's intestine, where the marble might have located. 20 rows of the data corresponded to 20 measurements. In order to save Fluffy, we implemented the Fast Fourier Transform algorithm to analyze the data. The goal was to determine the trajectory of the marble and its exact location in the dog's intestine.

I. Introduction and Overview

Fourier analysis is an important concept in signal processing. The basic idea behind the Fourier Transform is that given any mathematical function $f(x)$, one can express said function as a sum of sine and cosine functions. Fourier transform has many applications in different science fields.

II. Theoretical Background

Fourier Transform

We denote $F(k)$ as the Fourier form of the function $f(x)$. The Fourier transform is defined as

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

and the inverse of the Fourier transform is defined as

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

with k: wavenumber, $x \in [-\infty, \infty]$, but during computation, we approximate f(x) over a finite $x \in [-L, L]$.

The Kernel e^{-ikx} is the sum of $coskx + isinkx$, which describes the oscillatory behavior of the function[1]. When performing a Fourier transform on a signal/function, we are viewing the signal in the Fourier space (frequency domain). When taking the inverse of the Fourier transform, we switch back to viewing the signal in the real space (time domain).

Fast Fourier Transform (FFT) algorithm

The Fast Fourier Transform is a computation algorithm to approximate the Fourier series of a discretized function. Discretizing the function by transforming the domain x into 2^n points lowers the operation count to *O(NlogN)*[1]. This makes the the FFT an efficient algorithm for Fourier transform.

The FFT assumes 2π periodic domain[1]. For functions that do not work on a 2π periodic domain, FFT will yield significant errors. Thus, we have to rescale our k (wavenumber) via $\frac{2\pi}{2L}$. Another feature of the FFT is that it shifts the data so that $x \in [-L, 0] \Rightarrow x \in [0, L]$ and $x \in [0, L] \Rightarrow x \in [-L, 0]$.[1]

III. Algorithm Implementation and Development

Denoising the data through Averaging and Filtering

Because Fluffy the Dog was moving around a lot while we were taking the measurements, it had generated a lot of noises around our signal. Therefore, the first step of the analysis was to denoise our data using Averaging and Filtering. For our filter, we chose the 3D Gaussian Filter. The purpose of the filter was to "filter" out the white noise around the true signal. The Gaussian filter for the 3D data took the form of

$$exp(-\tau[(k_x - k_{xo})^2 + (k_y - k_{yo})^2 + (k_z - k_{zo})^2])$$

with τ: the bandwidth of the filter. We set $\tau = 0.2$.

k_x, k_y, k_z: the 3D coordinates of the data in the frequency domain.

k_{xo}, k_{yo}, k_{zo}: the 3D coordinates of the maximum of the central frequency.

The central frequency, generated by the marble, was found using the Averaging method. First, we performed FFT on each data slice, then accumulated all 20 FFT results into one 3D matrix. Next, we took the average of that matrix, which yielded us the average of the cumulative frequencies **(Utave)**. We then found the coordinates of the maximum frequency, then used those coordinates to construct the Gaussian Filter.

Determining the trajectory of the marble

After constructing the Gaussian Filter, we applied the filter to the FFT of each of the 20 data slices. We then returned to the time domain to find the maximum value of each slice. The 3D coordinates in the spatial domain (x, y, z) of each 20 maximum value were saved into 3 vectors **x**, **y**, **z**. Finally, we plotted the trajectory of the marble using the three vectors of the coordinates.

IV. Computational Results

After averaging over 20 data slices, we had significantly reduced the noise and got a clean central frequency (Figure 1). The max value of the central frequency was at `(1.8850, -1.0472, 0)`. Therefore, our Gaussian Filter became

$$exp(-0.2[(k_x - 1.8850)^2 + (k_y + 1.0472)^2 + (k_z)^2])$$

After analyzing the data, we obtained the plot of the marble's trajectory (Figure 2). The helix-shaped path also told us where the marble ended up in the dog's intestine. The marble was at the coordinate `(-5.6250, 4.2188, -6.0938)`.

Figure 1: Central Frequency after Denoising

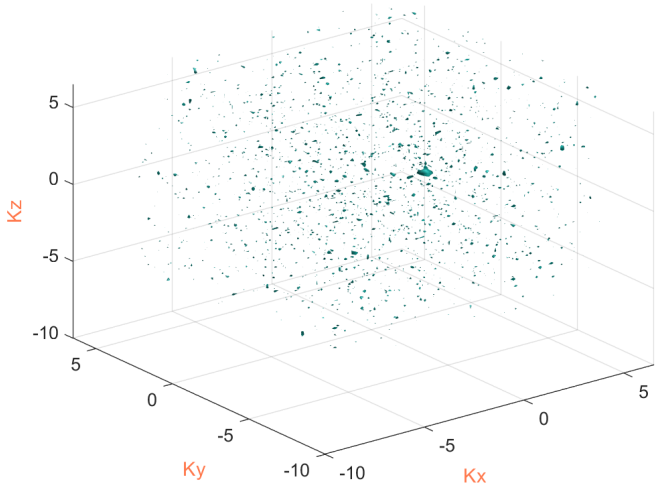

Figure 1: Visualization of the central frequency generated by the marble. After averaging over 20 noisy data slices, we achieved a relatively clean signal.

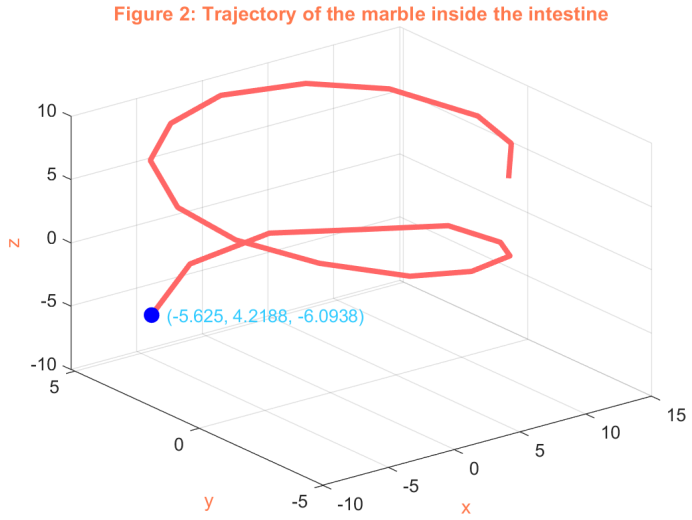

Figure 2: The trajectory of the marble inside the dog's intestine. The blue dot represents the final location of the marble.

V. Summary and Conclusions

Our data presented 20 ultrasound measurements. Our goal was to determine the trajectory of the marble inside Fluffy the Dog. To do so, we first denoised our data around the data's central frequency. After denoising the Fourier form of our data, we obtained the inverse Fourier form and located the marble at the coordinate [-5.6250 4.2188 -6.0938]. We broke up the marble using acoustic wave and saved our dog.

Appendix A: MATLAB functions used

reshape: Reshapes the initial 2D dataset into twenty 3D data slices. Each slice has a 64x64x64 dimension.

fftn: Performs the Fast Fourier Transform on the 3D data.

fftshift: Shifts the Fourier form of the function back to its mathematically correct position.

ifftshift: "Undoes" the effect of the **fftshift**, thus shifts the Fourier form back to its before-fftshift position.

ifftn: Takes the inverse of the Fourier transform.

[M, I] = max(): Finds and returns the maximum value (M) in a 3D matrix and its linear index (I).

[l, m, p] = ind2sub(): Converts the linear index (I) into a 3D index (l, m, p). The 3D index is then used to find the coordinates of the max value either in the frequency domain (Kx, Ky, Kz), or in the spatial domain (X, Y, Z).

Appendix B: MATLAB code

```
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);

slice = 20;
Utave = zeros(n, n, n);
for j=1:20
    Un(:,:,:) = reshape(Undata(j,:),n,n,n);
%    close all, isosurface(X,Y,Z,abs(Un),0.5)
%     axis([-20 20 -20 20 -20 20]), grid on, drawnow
%     pause(1)
    Unt(:,:,:) = fftn(Un);
%   Untp(:,:,:) = fftshift(abs(Unt)/max(abs(Unt(:))));
%     close all, isosurface(Kx, Ky, Kz, Untp, 0.5)
%     pause(1)
    Utave = Utave + Unt;
end
Utave = fftshift(Utave)/slice;
Utave_norm = abs(Utave)/max(abs(Utave(:)))
isosurface(Kx, Ky, Kz, Utave_norm,0.5);
title('Figure 1: Central Frequency after Denoising',...
    'Color', [255, 121, 77]/255)
xlabel('Kx', 'Color', [255, 121, 77]/255),
ylabel('Ky', 'Color', [255, 121, 77]/255),
zlabel('Kz', 'Color', [255, 121, 77]/255)
grid on

[M, I] = max(Utave_norm(:));
[l, m, p] = ind2sub(size(Utave_norm), I); %get the index of max k
kx0 = Kx(l, m, p);
ky0 = Ky(l, m, p);
kz0 = Kz(l, m, p);
filter = exp(-0.2 *((Kx - kx0).^2 + (Ky - ky0).^2 + (Kz - kz0).^2));

for j=1:slice
    Un(:,:,:) = reshape(Undata(j,:),n,n,n);
    Unt = fftn(Un);
    Unt = fftshift(Unt);
    Untf = Unt .* filter;
    Untf = ifftshift(Untf);
    Unf = ifftn(Untf); % go back to time domain
```

```
    [M, I] = max(Unf(:));
    [i1_max, i2_max, i3_max] = ind2sub(size(Unf), I);
    x_max(j) = X(i1_max, i2_max, i3_max);
    y_max(j) = Y(i1_max, i2_max, i3_max);
    z_max(j) = Z(i1_max, i2_max, i3_max);
end

plot3(x_max, y_max, z_max, 'Color', [255, 102, 102]/255,...
    'LineWidth', 3);
grid on
title('Figure 2: Trajectory of the marble inside the intestine',...
    'Color', [255, 121, 77]/255)
xlabel('x', 'Color', [255, 121, 77]/255),
ylabel('y', 'Color', [255, 121, 77]/255),
zlabel('z', 'Color', [255, 121, 77]/255)
hold on
end_point = [x_max(end) y_max(end) z_max(end)];
plot3(end_point(1), end_point(2), end_point(3), '.', ...
    'Color', 'b', 'markersize', 30)
text(x_max(end), y_max(end), z_max(end),...
    ['   (' num2str(x_max(end)) ', ' num2str(y_max(end)) ', ' ...
    num2str(z_max(end)) ')'], 'color', [51, 204, 255]/255)
```

Reference

1. Kutz, Nathan J. "Computational Methods for Data Analysis." *In Data-Driven Modeling & Scientific Computation: Methods for Complex Systems & Big Data.* Oxford University Press, 2013.