Computational Methods for Data Analysis

Homework 1

Due on January 24, 2020 at 11:00pm

Professor J. Nathan Kutz

Section A

Chandan Sharma Subedi

Abstract

In this paper, fast Fourier transform (FFT) is used to find the location of the marble, swallowed by Fluffy, from the noisy ultrasound data of a small region of the intestine. The frequency spectrum obtained from FFT for each data frame is averaged to reduce the white noise and obtain the center frequency. A Gaussian filter is implemented to attenuate higher frequencies. The inverse FFT is then used to get the denoised ultrasound data, from which the path of the marble is estimated.

1 Introduction and Overview

My dog has accidentally swallowed a marble. Upon inspection, the vet suspects that the marble has made its way up to the intestine. The vet performs ultrasound on a small region of the intestine. Due to internal fluid and Fluffy's movement, the ultrasound generates highly noisy data. In order to find the precise location of the marble, the noise from the data must be removed.

The data is collected for 20 different instances of time over a specific spatial domain. Each dataset, a 3D image, has the information about the location of the marble at that instance of time, as shown in Figure 1. However, due to excess noise, it is not possible to localize the marble.

In order to reduce the noise content of the data, FFT approach is used. FFT allows us to transfer problem from time domain to the frequency domain and vice-versa. In frequency domain, the frequency associated with the marble (center frequency) can be obtained by averaging the spectrum, assuming the noise content is random (white noise). Once the center frequency is obtained, the noise in spectrum can be attenuated by using band-filters like Gaussian filter. Gaussian filter acts as a low-pass filter, in relation to the center frequency, that removes higher frequencies. This low-noise spectrum can then be tranformed back into the time domain through inverse FFT to find the position of marble.
In Section 2, more detailed theoretical background on Fourier transform, spectral averaging and filtering is discussed. Section 3 outlines algorithms used as pseudocode and Section 4 contains computational results of analysis the ultrasound data using FFT approach.

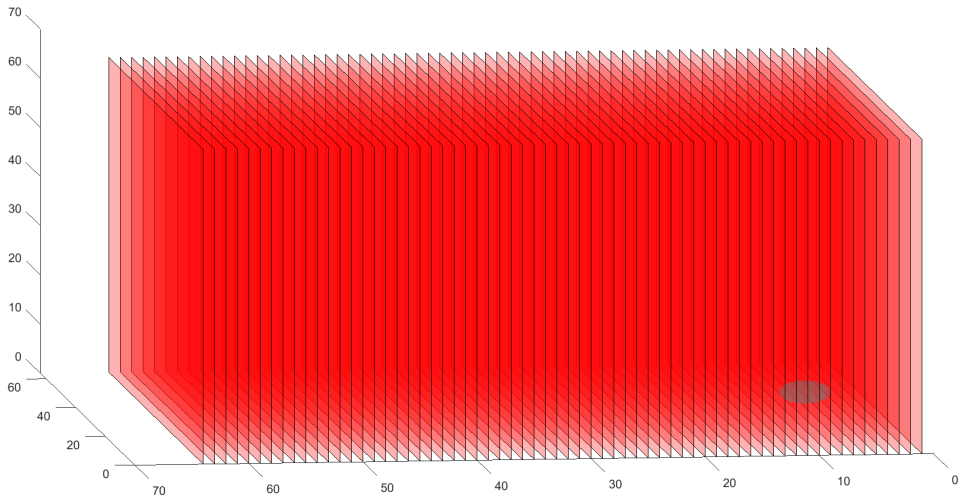

Figure 1: Visualization of a dataset (64 x 64 x 64 3D image). A marble is shown in the bottom left.

2 Theoretical Background

Any function in the interval $x \in (-L, L]$, can be expressed in terms of linear combination of sinusoidal functions $A\cos(2\pi fx + \phi)$ of different amplitudes (A), frequencies (f) and phases(ϕ), known as Fourier series. For functions with discontinuity in the interval, mean value can be used.

$$f(x) = \frac{a_0}{2} + \sum_{i=1}^{\infty} \left(a_n \cos \frac{n\pi}{L} x + b_n \sin \frac{n\pi}{L} x \right) \quad x \in (-L, L]. \tag{1}$$

Using the orthoganility of $\cos(x)$ and $\sin(x)$ functions, we can find the coefficients as

$$a_0 = \frac{1}{2L} \int_{-L}^{L} f(x)dx \qquad a_n = \frac{1}{L} \int_{-L}^{L} f(x) \cos \frac{n\pi}{L} x dx \qquad b_n = \frac{1}{L} \int_{-L}^{L} f(x) \sin \frac{n\pi}{L} x dx$$

Using exponential form of the $\cos(x) = \frac{e^{ix}+e^{-ix}}{2}$ and $\sin(x) = \frac{e^{ix}-e^{-ix}}{2i}$ in above equation, we can also express $f(x)$ in complex exponential form.

$$f(x) = \sum_{-\infty}^{\infty} c_n e^{\frac{in\pi}{L}x} \quad \in (-L, L]. \qquad c_n = \frac{1}{2L} \int_{-L}^{L} f(x) e^{-\frac{in\pi}{L}x} dx$$

Fourier tranform of any periodic function in the interval $x \in (-\pi, \pi]$ is a complex valued function of frequency whose modulus is the amount of that frequency and argument is the phase of the sinusoid. Fourier transform and its inverse are defined as

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

While the equation might look mysterious, it is conceptually very intuitive. e^{-ikx} rotates (winds) in clockwise direction about the origin in a unit circle on a complex plane with frequency k (also known as winding frequency). Thus, the integrand $f(x)e^{-ikx}$ is the original function $f(x)$ rotating around the complex plane with frequency k (like a coil around sphere). The integration of this integrand over all values of x in the interval gives the center of mass of the function $f(x)e^{-ikx}$. The center of mass stays near 0 for all values of frequencies, except when it is equal to the frequency of $f(x)$. When the winding frequency and the frequency of the function matches, the center of mass moves away from 0 and the frequency signature of function is detected.

Fast Fourier transform was the algorithm developed by Cooley and Tukey to quickly find the discrete Fourier tranform (DFT). Since there are n multiplication and n addition necessary, computing DFT directly requires $\mathcal{O}(n^2)$ operations. FFT uses the fact that odd and even sub-series can be computed concurrently. This forms a binary tree set of instructions that can be computed in $\mathcal{O}(n \log(n))$.

$$\begin{aligned} x[k] = \sum_{0}^{N-1} x[n] e^{-\frac{i\pi k[n]}{N}} &= \sum_{0}^{N/2-1} x[2n] e^{-\frac{i\pi k[2n]}{N}} + \sum_{0}^{N/2-1} x[2n+1] e^{-\frac{i\pi k[2n+1]}{N}} \\ &= \sum_{0}^{N/2-1} x[2n] e^{-\frac{i\pi k[n]}{N/2}} + e^{-ik\pi} \sum_{0}^{N/2-1} x[2n+1] e^{-\frac{i\pi k[n]}{N/2}} \end{aligned}$$

Since $x[-n] = x[n]^*$, the values for the negative frequencies can be simply appended at the end after finding complex conjugates.

Spectral averaging is a technique to increase signal to noise ratio. Since FFT is a complex valued function, there are multiple ways one can average real and imaginary parts of complex numbers. Vector averaging involves averaging real and imaginary components separately. This is the method that has been implemented in this paper. Since, imaginary components arise due to the phase shift of the signal, averaging it separately could lead to random phase cancellation. Thus if one is using vector averaging, the signal in each data must be strictly in phase. This is equivalent to averaging in time-domain and it reduces the noise floor. RMS

averaging is another technique that preserves the energy of the spectrum. While it does not lower the noise floor, it decreases the noise fluctuation that helps to distinguish non-noise component.

Filtering the spectrum allows one to isolate signal from the noise. Gaussian filter is a simple low-pass band filter.

$$G_f(k) = e^{-\tau(k-k_0)^2}$$

where τ is the measure of the bandwidth and k is the wave number.

3 Algorithm Implementation and Development

The function implemented that estimates the path of the marble in the Fluff's intestine has four main components.

- Data acquisition and formatting:
  Once the data is loaded, each instance is transformed into a 64 x 64 x 64 multidimensional array representing a 3D image. Spatial and spectral meshgrid is created over which input signal and spectrum assume their values respectively. Since FFT is defined over an interval $(-\pi, \pi]$, the wave number k is scaled by π/L. $(-L, L]$ is the domain of input data with spatial resolution $\frac{2L}{N+1}$, where N is the number of frequency modes chosen.

- Spectral averaging:
  Vector averaging of 20 spectra obtained from MATLAB in-build FFT command *fftn()* is performed to find the center frequency.

- Gaussian filtering:
  3D Gaussian filter of unit bandwidth is implemented in frequency domain to isolate the center frequency from noise.

- Path generation:
  The filtered spectra are then inverse Fourier tranformed using MATLAB in-built command *ifftn()*. The peak value for each signal data is calculated along with its coordinates. The coordinates are plotted to show the path traced by the marble.

Algorithm 1: Homework01.m

```
Import data from Testdata.mat
Initialize average spectrum U_ave
for j = 1 : 20 do
    Extract measurement jth, U_n from Undata
    U_ave = U_ave + FFT(U_n)
end for
Find the center frequencies k_c
Implement Gaussian filter, G_f(k)
Initialize path array
for j = 1 : 20 do
    Extract measurement jth from Undata
    FFT
    Filter the spectrum
    Inverse FFT
    Find peak value and location
    Append location to path
end for
```

4 Computational Results

The isosurface plot of vector averaged spectrum is shown in the Figure 2. The green blob represents the center frequency vector k_c.

$$k_c = [1.8850, -1.0472, 0]$$

Based on this frequency, Gaussian filter was designed to filter the noise in the spectrum. The Figure 3 shows the isosurface plot of performing FFT, Gaussian filtering and inverse FFT operations on the 20th dataset. The denoised spatial data clearly shows the location of the marble. The coordinates of marble at each instance of time were obtained by finding the indices associated with the peak value of the denoised data.The location at 20th instance was found to

$$[x, y, z] = [-5.6250, 4.2188, -6.0938]$$

The Figure 4 shows the path of the marble for 20 instances of time.

5 Summary and Conclusions

The center frequency was estimated after spectral averaging of 20 spectra obtained from the FFT of raw datasets. Gaussian filter was implemented in frequency domain to isolate the center frequency from the noise. Denoised spatial data was then obtained by inverse FFT of the filtered spectrum. By finding the indices associated with the peak value of denoised data, marble's location was estimated. It was found that in order to break the marble, the laser must be pointed at the location $[-5.6250, 4.2188, -6.0938]$.

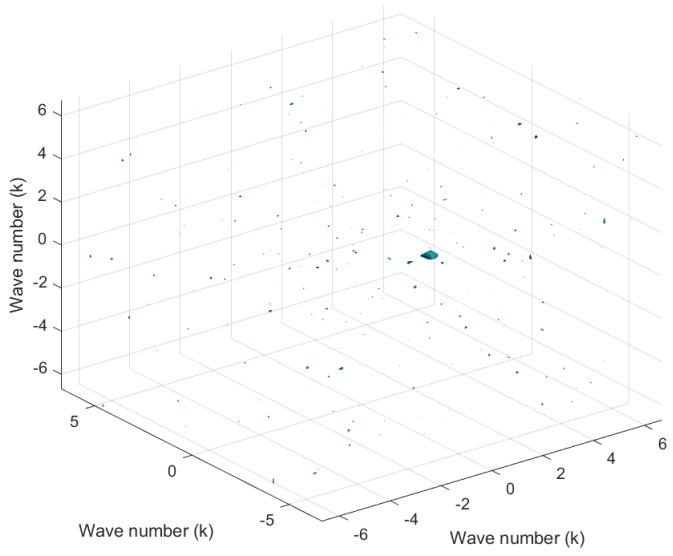

Figure 2: Isosurface plot of the shifted and normalized average spectrum.

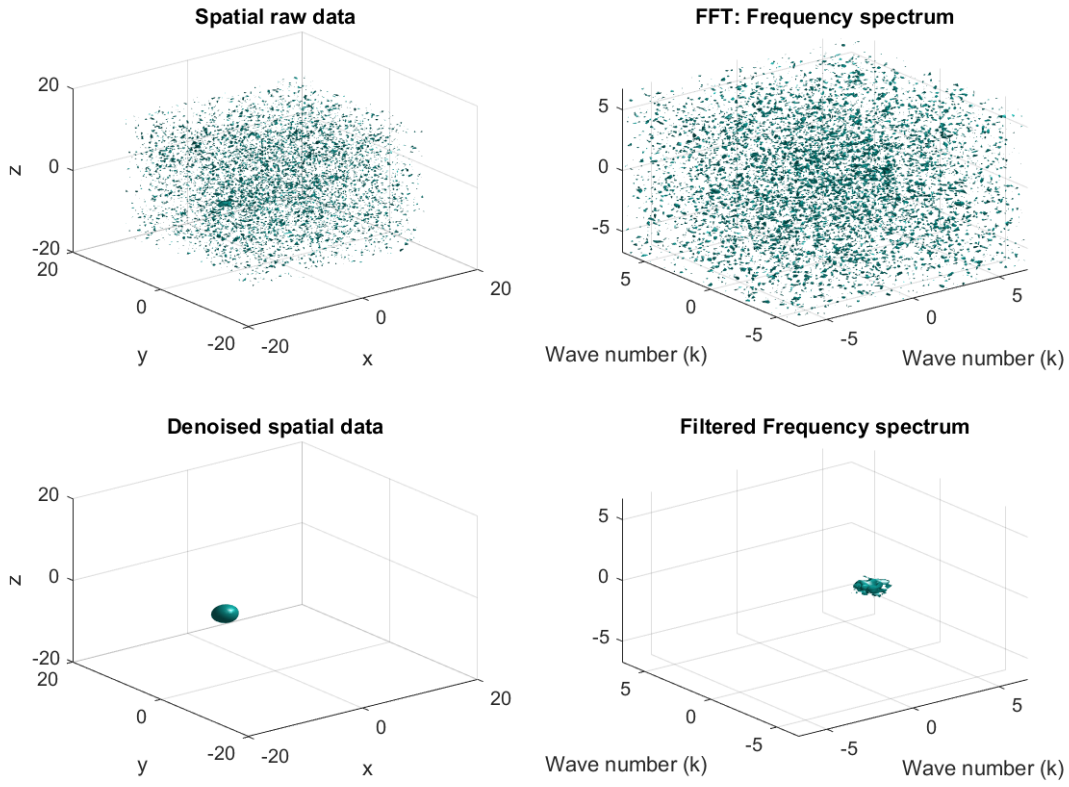

Figure 3: Figure shows isosurface plots of data at different stages of analysis. Raw spatial data (top left) is FFT to obtain frequency spectrum (top right). The frequency spectrum is then filtered using Gaussian filter to obtained filtered frequency spectrum (bottom right). And finally it is inverse FFT to obtain the denoised spatial data (botton left)

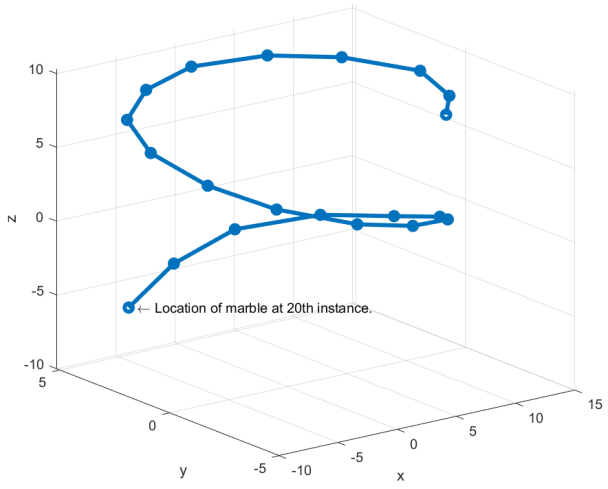

Figure 4: Path traced by the marble over 20 instances of time.

Appendix A MATLAB Functions

Some important MATLAB functions used during the implementation.

- `y = linspace(x1,x2,n)` returns a row vector of `n` evenly spaced points between `x1` and `x2`.
- `[X,Y,Z] = meshgrid(x,y,z)` returns 3-D grid coordinates defined by the vectors x, y, and z.
- `ks = fftshift(k)` rearranges FT by shifting zero frequency component to the center of the array. Exchanges two halves of the array.
- `Y = fftn(X)` returns multidimensional Fourier transform of an N-D array using FFT algorithm
- `Y = ifftn(X)` returns multidimensional discrete inverse Fourier transform of an N-D array using FFT algorithm
- `fv = isosurface(X,Y,Z,V,val` computes isosurface data from the volume data V at the isosurface value specified in val. Isosurface connects all points whose values at those points is val.
- `[row, col] = ind2sub(sz, ind)` returns the arrays row and col containing the equivalent row and column subscripts corresponding to the linear indices ind for a matrix of size sz.
- `B = reshape(A, sz)` reshapes A using size vector sz.
- `fill3` function creates flat-shaded and Gouraud-shaded polygons.

A note to remember. When plotting isosurface plot (or any 3D plot) into subplot, you need to overwrite the view and lighting variables. When subplot is initialized, it assigns 2D plotting axes by default. `view(3)`; `camlight`; `lighting gouraud`

Appendix B MATLAB Code

```
%% Clear Workspace
clear all; close all; clc;
%% Load the ultrasound data
load Testdata
%% Setup global variable
AnalysisPlotting = false;
%% Averaging the data in frequency domain to find center frequency.
L=15;
n=64;                                   % fourier modes

x2 = linspace(-L, L, n+1); x=x2(1:n); y=x; z=x;
[X,Y,Z]=meshgrid(x,y,z);

k=(2*pi)/(2*L)*[0:(n/2-1) -n/2:-1];     % scaled wavenumber
ks=fftshift(k);
[Kx,Ky,Kz]=meshgrid(ks,ks,ks);

K=max(abs(k));

Uave=zeros(n,n,n);
for t=1:20
    Un(:,:,:)=reshape(Undata(t,:),n,n,n);
    Utn=fftn(Un);
    Uave=Uave+Utn;
end
Uave=fftshift(Uave)/20;                 % shifted average spectrum

[M,I]=max(Uave(:));
[j,i,k]=ind2sub(size(Uave), I);
kc=[ks(i), ks(j), ks(k)];               % center frequencies
%% 3D Gaussian Filter to filter the data around center frequency.
% filt(:,1)=exp(-(ks-kc(1)).^2);
% filt(:,2)=exp(-(ks-kc(2)).^2);
% filt(:,3)=exp(-(ks-kc(3)).^2);
%
% gauss_filter=meshgrid(filt(:,1),filt(:,2),filt(:,3));
gauss_filter= exp(-((Kx-kc(1)).^2 + (Ky-kc(2)).^2 + (Kz-kc(3)).^2));

%% Find the path of the marble
path=[];
for t=1:20
    Un(:,:,:)=reshape(Undata(t,:), n,n,n);
    Utn=fftn(Un);                       % fourier transform
    Utf=gauss_filter.*fftshift(Utn);    % filter around center freq
    Uf=ifftn(fftshift(Utf));            % inverse fourier transform

    [M,I]=max(abs(Uf(:)));
    [j,i,k]=ind2sub(size(Uf), I);
    location=[x(i), y(j), z(k)];
    path=[path; location];
end
%% Plot the path of the marble
```

```
figure(1)
plot3(path(:,1), path(:,2), path(:,3), '-o', 'Linewidth', [3])
grid on, xlabel("x"), ylabel("y"), zlabel("z")
text(path(20,1),path(20,2),path(20,3),...
    "  \leftarrow Location of marble at 20th instance.",...
    'HorizontalAlignment','left','FontSize',10);
fprintf('The location of the marble at 20th instance is %f, %f, %f \n',
   path(20,:))

%% Additional plot for analysis
if AnalysisPlotting
    points=[[0 0 64]; [64 0 64]; [64 0 0]; [0 0 0]];
    for jj=1:64
        offset(:,:,jj)=[zeros(1,4);jj*ones(1,4);zeros(1,4)]';
    end

    points = bsxfun(@plus, points, offset);
    size(points)

    Xr = reshape(points(:,1,:),4,64);
    Yr = reshape(points(:,2,:),4,64);
    Zr = reshape(points(:,3,:),4,64);

    % Plot the representation of data
    figure(2); grid on;
    fill3(Xr, Yr, Zr, 'r');
    hold on;
    [Xs,Ys,Zs] = meshgrid(0:0.1:20,0:0.1:20,0:0.1:20);
    isosurface(Xs, Ys, Zs, (Xs-10).^2+(Ys-10).^2+(Zs-10).^2, 5)
    alpha(0.3);                          % transparency

    % Plot the isosurface of averaged spectrum
    figure(3);
    set(0, 'defaultTextFontSize',15);
    isosurface(Kx,Ky,Kz,abs(Uave)/max(abs(Uave(:))),0.6)
    axis([-K K -K K -K K]), grid on, drawnow
    xlabel("Wave number (k)")
    ylabel("Wave number (k)")
    zlabel("Wave number (k)")

    % Plot the isosurface for 4 stages of tranformation
    figure(4);
    set(0, 'defaultTextFontSize',15);
    hold on;

    Un(:,:,:)=reshape(Undata(10,:), n,n,n);
    subplot(2,2,1); view(3); camlight; lighting gouraud
    isosurface(X,Y,Z,abs(Un)/max(abs(Un(:))),0.5)
    axis([-20 20 -20 20 -20 20]), grid on
    xlabel('x')
    ylabel('y')
    zlabel('z')
    title("Spatial raw data")
```

```
    Utn=fftn(Un);
    Utn=fftshift(Utn);
    subplot(2,2,2); view(3); camlight; lighting gouraud
    isosurface(Kx,Ky,Kz,abs(Utn)/max(abs(Utn(:))),0.5)
    axis([-K K -K K -K K]), grid on
    xlabel("Wave number (k)")
    ylabel("Wave number (k)")
    xlabel("Wave number (k)")
    title("FFT: Frequency spectrum")

    Utf=gauss_filter.*Utn;
    subplot(2,2,4); view(3); camlight; lighting gouraud
    isosurface(Kx,Ky,Kz,abs(Utf)/max(abs(Utf(:))),0.3)
    axis([-K K -K K -K K]), grid on
    xlabel("Wave number (k)")
    ylabel("Wave number (k)")
    xlabel("Wave number (k)")
    title("Filtered Frequency spectrum")

    Uf=ifftn(fftshift(Utf));
    subplot(2,2,3);view(3); camlight; lighting gouraud
    isosurface(X,Y,Z,abs(Uf)/max(abs(Uf(:))),0.6)
    axis([-20 20 -20 20 -20 20]), grid on
    xlabel('x')
    ylabel('y')
    zlabel('z')
    title("Denoised spatial data")
end
```