AMATH482 Homework 1: An Ultrasound Problem

Kurumi Watanabe

Due: Jan 25, 2019

Abstract

Two applications of FFT in denoising were explored in this report, through analyzing a spatial trajectory of a marble in a noisy three-dimensional ultrasound data. Spectral averaging was used to discriminate between the center frequency produced by the marble and noise, and frequency filtering about the center frequency of the marble was utilized to extract information from the surrounding noise. The filtered frequency was then transformed back to the spatial domain to locate the position of the marble.

Contents

1 Introduction and Overview

Fluffy the dog swallowed a marble. The vet suspected that it is making its way through Fluffy's intestines, and performed an ultrasound to determine the trajectory. However, the intestinal fluid movement causes significant noise in the signal. To track the movement of the marble, the data must be denoised by 1) performing spectral averaging to find the frequency signature of the marble, then 2) filtering out the noise to focus only on the frequency of interest (determined previously by averaging).

2 Theoretical Background

Ultrasound is a commonly used medical imaging technique that utilizes high-frequency sound waves and its reflections. The sound wave is emitted by a probe on the ultrasound machine, which then travels through tissue boundaries where it reflects. An image is constructed using information about the distance, which is calculated using time from emission to return of the reflection of the sound wave, and intensity of the echo.

The data provided includes the signal data of ultrasound scans at 20 successive time points. To find the trajectory of the marble, denoising steps were performed on the noisy data, as follows:

1. **Spectral Averaging:** The data provides 20 signals collected over the time domain. By averaging the Fourier transform of the 20 signals (assuming that the marble is producing the strongest signal within the defined spatial domain), we can expect to see a convergence to a clean spectral signature in which the signal is easily discriminable from the noise.

2. **Frequency Filtering:** From spectral averaging, the frequency signature generated by the marble was determined. Because the frequency of interest is known, other frequency components can be filtered out.

3 Algorithm Implementation and Development

3.1 Domain and the Fourier mode

- Spatial domain is defined between $[-15, 15]$

```
x2=linspace(-L,L,n+1);
x=x2(1:n);
```

A vector of 65 linearly spaced points between -15 and 15 is first created; of these, we consider the first 64th, for periodicity (i.e. 1st and 65th are equal)

- Frequency domain is defined between $[-32, 31]$

```
k=(2*pi/(2*L))*[0:(n/2-1) -n/2:-1];
ks=fftshift(k);
```

The numeric frequency domain is rescaled by $\frac{2\pi}{2*L}$ as FFT expects a 2π periodic domain.

3.2 Spectral averaging to find frequency signature

- Fourier transform of each data point is added, then averaged at end

```
uave = zeros(n,n,n);
for t = 1:20
un = reshape(Undata(t,:),n,n,n);  % reshape the data to 64x64x64
unf = fftn(un);  % 3D fft
uave = uave + unf;  % Add to Uave
end

uave = abs(fftshift(uave)) / 20; % Average from measurements at 20
  % timepoints!
```

- Frequency signature produced by the marble is determined by finding the maximum in the average of the frequencies

```
[M,I] = max(uave(:));
[Kx_I, Ky_I, Kz_I] = ind2sub(size(uave),I);  % coord of I in 64x64x64

f_sig = [Ksx(Kx_I, Ky_I, Kz_I),
        Ksy(Kx_I, Ky_I, Kz_I),
        Ksz(Kx_I, Ky_I, Kz_I)];
```

Center frequency: $[1.8850, -1.0472, 0]$

3.3 Filtering about the center frequency to track the trajectory of marble

- Constructed Gaussian filter around center frequency: $F(k) = exp(-\tau(k - k0)^2)$. Inverse Fourier transform to find X,Y,Z coordinates

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

% Part 2: Applying the filter to denoise
for t = 1:20  % for 20 timepoints

 un = reshape(Undata(t,:),n,n,n);  % reshape each row to 64x64x64
 unt = fftn(un);
 untf = unt .* filter;  % apply frequency filter
 unf = ifftn(untf);  % inverse fft

 [M,I] = max(unf(:));
 [x_I,y_I,z_I] = ind2sub(size(unf),I);

 X_marb(t,:) = X(x_I, y_I, z_I);
 Y_marb(t,:) = Y(x_I, y_I, z_I);
 Z_marb(t,:) = Z(x_I, y_I, z_I);

end

final_marb = [X_marb(20), Y_marb(20), Z_marb(20)]
```

Location of marble at 20th datapoint (Labeled as blue circle, below): $[-5.6250, 4.2188, -6.0938]$

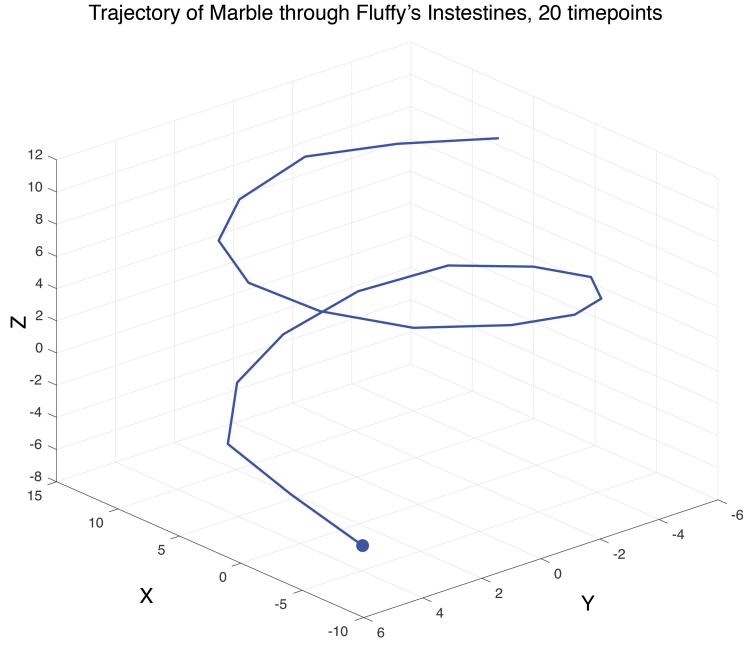

Figure 1: Trajectory of the marble through Fluffy's intestines.
At 20th data point: [-5.6250 4.2188 -6.0938]

4 Computational Results

By averaging the frequency/filtering around the center frequency to the Fourier transform, the trajectory of the marble could be determined from the noisy data. The acoustic wave should be focused at $[-5.6250, 4.2188, -6.0938]$ to break up the marble.

5 Summary and Conclusions

A system can be denoised by averaging the frequency over multiple realizations of the data, as noise (specifically, white noise) should average to zero. With enough realizations, the signal should become increasingly easier to differentiate from noise. Filtering is also useful when frequency of interest is known as it allows extraction of information at a specific frequency. In this case, it was determined from the previous frequency averaging. After the filter was applied, inverse FFT was applied to bring back the data into the spatial domain.

6 Appendix

6.1 Appendix A. MATLAB functions used

- `fftn()`

  returns the multidimensional Fourier transform of an N-D array. Equivalent to computing the 1-D transform along each dimension of X.

- `Y = fftshift(X)`

  rearranges Fourier transform X by shifting the zero-frequency component to the center of the array. Adjusts for the $x \in [0, L] \rightarrow [-L, 0]$ and $xin[-L, 0] \rightarrow [0, L]$ shift in data (performed by FFT algorithm for computational speed).

- `ifft(), ifftn()`

  returns the multidimensional discrete inverse Fourier transform of an N-D array. Equivalent to computing the 1-D inverse transform along each dimension of Y.

- `[I1, I2, I3, ..., IN] = ind2sub(siz, IND)`

  Takes linear index and returns the corresponding coordinate in an array of dimension siz

- `[M,I] = max()`

  Finds maximum element in an array and returns the value in M and index in I

- `[X,Y,Z] = meshgrid(x,y,z)`

  returns 3-D grid coordinates defined by the vectors x, y, and z

- `B = reshape(A, SIZ)`

  reshapes A to return B of size defined by SIZ

6.2 Appendix B. MATLAB codes

6.2.1 Spatial Domain/Fourier Modes

(a) Defining the domain and the Fourier mode

```
clear  all; close all; clc;
load Testdata

L=15; % spatial domain
n=64; % define 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];  % define k and rescale to 2pi domain
ks=fftshift(k); % shifted version of k

[X,Y,Z]=meshgrid(x,y,z); % creates
[Ksx,Ksy,Ksz] = meshgrid(ks,ks,ks);
[Kx,Ky,Kz] = meshgrid(k,k,k);
```

6.2.2 Frequency Averaging

(a) Frequency averaging

```
%% Spectrum averaging / Frequency signature of marble
% Part 1: Spectrum Averaging
uave = zeros(n,n,n);

for t = 1:20
un = reshape(Undata(t,:),n,n,n);  % reshape the data to 64x64x64
unf = fftn(un);  % 3 dimensional fft
uave = uave + unf;  % Add to Uave
end

uave = abs(fftshift(uave)) / 20; % Average spectrum from measurements at 20
% timepoints!
```

(b) Finding Frequency Signature of Marble

```
% Part 2: Find frequency signature of marble
[M,I] = max(uave(:));  % M=max value / I=linear index of max value
[Kx_I, Ky_I, Kz_I] = ind2sub(size(uave),I);  % Find coord of I in 64x64x64

f_sig = [Ksx(Kx_I, Ky_I, Kz_I), Ksy(Kx_I, Ky_I, Kz_I), Ksz(Kx_I, Ky_I, Kz_I)];
% [1.8850, -1.0472, 0]
```

6.2.3 Frequency Filter

(a) Constructing filter around center frequency

```
%% Frequency filtering
% Part 1: Construct filter around frequency signature determined previously

% F(k) = exp(-t(k - k0)^2)
filter = exp(-1 * ( (Kx-f_sig(1)).^2 + (Ky-f_sig(2)).^2 + (Kz-f_sig(3)).^2 ));

% Part 2: Applying the filter to denoise
for t = 1:20  % for 20 timepoints
un = reshape(Undata(t,:),n,n,n);  % reshape each row of raw data to 64x64x64
unt = fftn(un);
untf = unt .* filter;  % apply frequency filter
unf = ifftn(untf);  % inverse fft

[M,I] = max(unf(:));
[x_I,y_I,z_I] = ind2sub(size(unf),I);

X_marb(t,:) = X(x_I, y_I, z_I);
Y_marb(t,:) = Y(x_I, y_I, z_I);
Z_marb(t,:) = Z(x_I, y_I, z_I);
end
```

6.2.4 Plotting

(a) Plotting

```
%% Plotting
close all;
figure
plot3(X_marb, Y_marb, Z_marb, 'b', ...
'LineWidth', 2)
hold on
plot3(X_marb(end), Y_marb(end), Z_marb(end), '-bo', ...
    ...
'MarkerSize', 10, ...
'MarkerFaceColor', [.1 .1 1])
title('Trajectory of Marble in Fluffy''s Intestines, ...
    20 timepoints', ...
'FontWeight', 'normal', ...
'FontSize', 20)
xlabel('X', ...
'FontSize', 14)
ylabel('Y', ...
'FontSize', 14)
zlabel('Z', ...
'FontSize', 14)
set(gca,'FontSize',12)
grid on
hold off
```