AMATH 582 Homework 1

Tianqi Gu

January 24, 2020

Abstract

In this report we are going to determine a marble's spatial coordinate inside my dog fluffy's stomach in order to save my dog. To do this I mainly use fast Fourier transform algorithm, and in the end the trajectory of the marble inside fluffy is found.

1 Introduction and Overview

The data given is of dimension 20×262144, where each row represents a measurement taken and this array of length 262144 can be reshaped into a $64 \times 64 \times 64$ matrix represents its signal strength in spatial domain. To first determine the frequency of marble, I simply apply the 3D Fourier transform on these 20 realizations to get data in frequency domain, and after that I average these 20 spectrum and find its peak. That is where the signature frequency of the marble locates.
Based on this peak location, I construct a Gaussian filter and apply it to each of 20 frequency domain signals, and use Fourier inverse transform on each of them afterwards. By doing this I expect to find marble's spatial location in our 20 snapshots and thus find its trajectory. If we use an intense acoustic wave focusing on the last position of the trajectory of the marble, my dog fluffy will be saved!

2 Theoretical Background

As we learned from our textbook [**kutz˙2013**], Fourier introduced the idea of transform a function in space(or time) domain to frequency domain and inverse transform back given by the following formulas.

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

In mid 60s, a Fast Fourier Transform algorithm is developed, aka FFT, which find transform on the interval $[-L, L]$, which in our case $L = 15$. In my report I mainly focus on applying FFT to explore the data and find peak signals.
When applying filter on frequency domain signals, I used Gaussian filter defined by

$$\mathbf{F}(k) = \exp(-\tau(k - k_0)^2)$$

where τ is the decay rate and I choose $\tau = 2$, and k_0 is the signature frequency

3 Algorithm Implementation and Development

The whole algorithm in general to determine the trajectory of marble can be list as below

1. FFT 20 measurements to get their frequency domain signals and average them.
2. Find the signature frequency and build a filter based on it.

3. Apply filter on frequecy domain signals and IFFT back to spatial domain. The locations of peak signal in spatial domain will be the locations of the marble.

Algorithm 1: Find trajectory of the marble

Import data from `Testdata.mat`
for $j = 1 : 20$ **do**
  Extract measurement j from `Undata`
  FFTn extracted measurements, save them
end for
Average frequency domain signals
Find the maximum value and coordinate of averaged frequency domain signals
Construct filter
for $j = 1 : 20$ **do**
  Filter $\times$ frequency domain signals
  IFFTn filtered signals
  Find the maximum value and coordinates in spatial domain
end for

4 Computational Results

By applying the above algorithms we first find out the signature frequency at $[k_x, k_y, k_z] = [1.8850,\ -1.0472,\ 0]$. In Figure 1 we can clearly see that there is a peak located at the place I wrote down above, which will be the signature frequency.
Figure 2 is the illustration for the Gaussian filter we are using, centered at signature frequency as we expected.
Figure 3 is the trajectory we finally find using algorithms above. In the 20th data measurement the final location of the marble is $[x, y, z] = [-5.6250,\ 4.2188,\ -6.0938]$. By hitting a ultrasonic wave at this point, the marble will be destroyed and my dog will be saved.

5 Summary and Conclusions

From this report we have seen that averaging over realizations is a quite useful to cleaning noisy data where the peak signal stays still, and Gaussian filter plays an important role recovering the signal. And FFT provides a powerful tool to analyzing both spatial and frequency domain data.

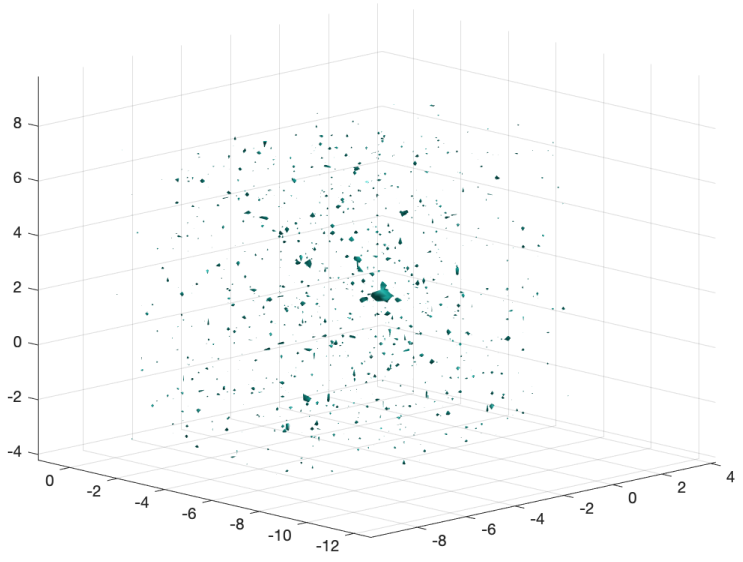

Figure 1: Averaged Frequency domain spectrum

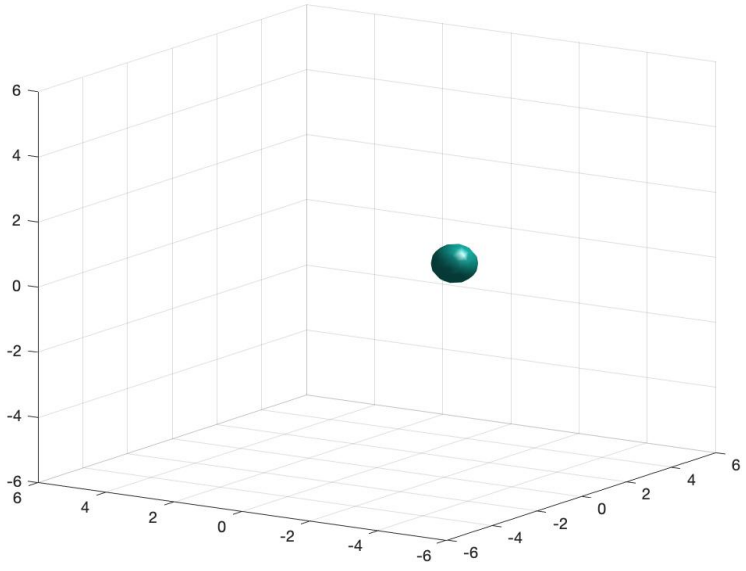

Figure 2: Gaussian filter with $\tau = 2$

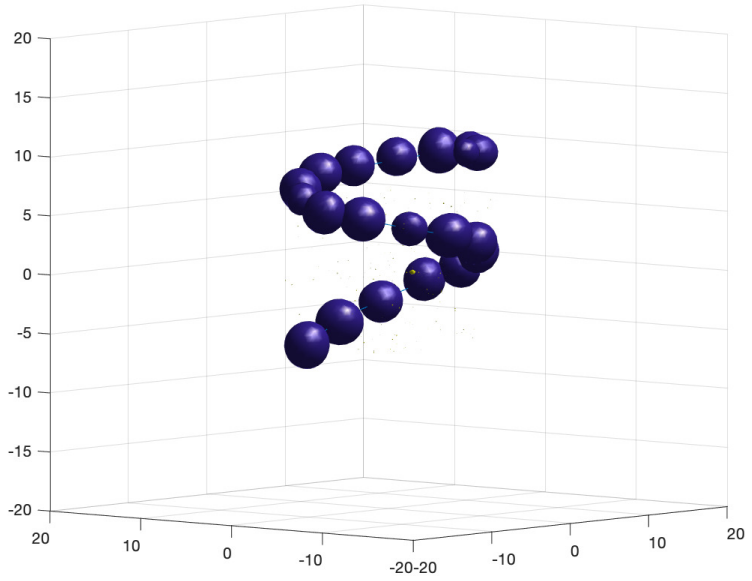

Figure 3: Trajectory of the marble

Appendix A MATLAB Functions

unordered list:

- `Ut = fftn(Un)` returns a Fourier transformed 3D signal in frequency domain.
- `Un = ifftn(Ut)` returns an inverse Fourier transformed 3D signal in spatial domain.
- `fftshift(Ut)` returns a shifted signal afte Fourier transform.
- `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 based on the coordinates contained in the vectors `x`, `y` and `z`. X is a matrix where each row is a copy of `x`, `Y` is a matrix where each column is a copy of `y` and `Z` is a matrix where each column is a copy of `z`. The grid represented by the coordinates `X`, `Y` and `Y` is of dimension `length(y)` $\times$ `length(x)` $\times$ `length(z)`.

Appendix B MATLAB Code

The code of this project can be accessed at my Github page: https://github.com/punkfloyd137/3D-signal-detection

```
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);

Ut_av = zeros(n,n,n);
for j=1:20
  Un(:,:,:)=reshape(Undata(j,:),n,n,n);
  Ut_av = Ut_av + fftn(Un);
end

Ut_av = Ut_av/20;
Ut_av = fftshift(Ut_av);
Ut_abs = abs(Ut_av);

[M,I] = max(Ut_abs(:));
%normalize
Ut_abs = Ut_abs/M;

%Plot frequency domain
close all, isosurface(Kx,Ky,Kz,abs(Ut_abs),0.6)
axis([-20 20 -20 20 -20 20]), grid on, drawnow
pause(1)


%Get coordinate of peak signal
[a,b,c]=ind2sub(size(Ut_abs),I);
kp = [ks(a),ks(b),ks(c)];


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

movement = zeros(20,3);
for j=1:20
    Un(:,:,:)=reshape(Undata(j,:),n,n,n);
    Ut = fftn(Un);
    Ut = fftshift(Ut);
    unft = Ut.*filter;

    unft = ifftshift(unft);
    Unf = ifftn(unft);
    %Unf = ifftshift(Unf);
    [M,I] = max(Unf(:));
    [a,b,c]=ind2sub(size(Unf),I);
    location = [x(a),x(b),x(c)];
    movement(j,:) = location;
    isosurface(X,Y,Z,abs(Unf),0.1),
    axis([-20 20 -20 20 -20 20]), grid on, drawnow, hold on
end
```

Listing 1: All codes for this report