An Ultrasound Problem

AMATH 482 Homework 1

Robin Yan

Jan. 17, 2020

Abstract

Fourier transform can translate signals in time domain into frequency domain. In practice, Fast Fourier transform (FFT) is used for its high efficiency. This homework demonstrates one of the most common usage of FFT – noise reduction. The given signal data contains 20 measurements of a marble inside a given volume contaminated by high level of noise. The script uses FFT to identify the signature frequency of the marble and damper the noisy frequency and it successfully extracts the location of the marble and calculates its corresponding trajectory.

Section I – Introduction and Overview

The purpose of this homework is to implement FFT to perform noise reduction. The given data is a 20 by 262144 double matrix in MATLAB data format. It is from an ultrasound measurement of a marble inside a dog's intestine but is contaminated by the signal of the moving fluid around the marble. The goal is to design a script that extract the location and trajectory of the marble to allow the use of intense acoustic wave to break up the marble.

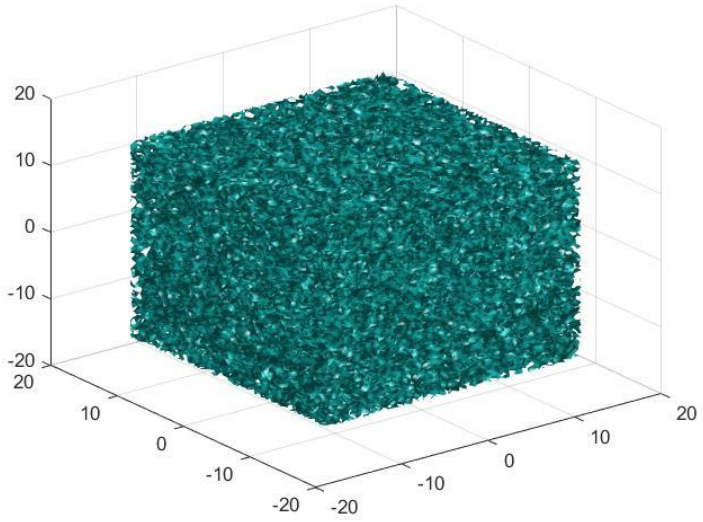

Fig. 1 Spatial data contaminated by noise (at timeframe 1)

The principles of this assignment is to use FFT to transform the contaminated time domain data into frequency domain, average the multiple measurements to identify the signature frequency, damper the remaining frequency using a filter and ultimately transform the filtered data back to time domain for interpretation.

Section II – Theoretical Background

Fourier Transform

The basis of Fourier transform is Fourier series, which states that a given function can always be represented by a series of sines and cosines:

$$f(x) = \frac{a_0}{2} + \sum_{n=0}^{\infty} (a_n \cos nx + b_n \sin nx) \qquad (E.1)$$

in which x is from $-\pi$ to π. This is true even for discontinuous functions. To calculate the Fourier coefficients a_n and b_n, the following formulae can be used:

$$a_n = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\cos nx\,dx \quad (E.2)$$

$$b_n = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\sin nx\,dx \quad (E.3)$$

The above formulae can be combined into complex domain for simplicity, and then the Fourier transform is defined as:

$$F(k) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty} e^{-ikx} f(x)dx \quad (E.4)$$

$$f(x) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty} e^{ikx} F(k)dk \quad (E.5)$$

In practice, the computational domain is limited to a finite domain from $-L$ to L.

Fast Fourier Transform

Fast Fourier transform lowers the operational count of Fourier transform from $O(N^2)$ to $O(N\log N)$, which significantly improves the computational speed of Fourier transform. The algorithm discretizes the time domains into 2^n points to speed up the process.

Gaussian Filter

Gaussian filter in 3D creates an array of indices decaying around a given point, and is written as:

$$\mathcal{F}(k) = e^{(-\tau((k_x-k_{x0})^2+(k_y-k_{y0})^2+(k_z-k_{z0})^2)} \quad (E.6)$$

In which τ is the bandwidth of the filter, positively related to the decay rate of the filter and k_{x0}, k_{y0} and k_{z0} are the location of the signature frequency.

Section III – Algorithm Implementation and Development

Signature Frequency Identification

The spatial domain, $2L$, is specified to be from -15 to 15, and discretized into 64 data points, which corresponds to the cubic root of 262144, the number of columns in the data. Upon loading the data, the spatial coordinates (x) are generated as $(n+1)$ points on the defined domain, as the last point is discarded due to the periodic assumption of Fourier transform. Similarly, the remaining two-dimensional coordinates (y and z) are generated. Corresponding frequency domain coordinates (k) are generated by creating a vector of 0 to $\frac{n}{2}-1$ and $-\frac{n}{2}$ to -1 and scaling it by a factor of $\frac{2\pi}{2L}$, as the default order of Fourier coefficients in MATLAB starts from 0 and MATLAB assumes the frequency domain is from $-\pi$ to π. The k coordinates are shifted for plotting, and both spatial and frequency coordinates are meshed for plotting in 3D space.

Subsequently, a for loop iterates through all 20 measurement to extract and reshape the data back to three-dimensional space. Then n dimensional FFT command execute FFT on all 3 dimensions to create a new 3D matrix in the frequency domain (E.4). The absolute values of resulting 20 matrices are averaged and a threshold is applied to plot the data in 3D. After a few iterations, an appropriate threshold (0.6) is set, and the script can generate an isosurface plot to identify signature frequency region.

Filtering

A Gaussian filter in 3D is generated in the shifted frequency domain based on the location of the previously identified signature frequency. The script then shifts the frequency data to align it with the Gaussian filter and perform array multiplication to reduce the magnitude outside the signature region (E.6). The filtered frequency data is then converted back to spatial domain using n dimensional inverse FFT (E.5) for interpretation.

The Gaussian filter uses an arbitrarily determined bandwidth ($\tau = 0.3$) which can be adjusted based on the accuracy required for the results. This filtering process is carried out in all 20 measurements, and the results are plotted using *isosurface* command at preset 0.4 magnitude to illustrate the location of the marble at each time frame.

Trajectory Calculation

The script extracts the vertices of the plot from *isosurface* command and average their coordinates to calculate the geometric center of the marble. The consequent 20 coordinates are plotted using plot3 command the demonstrate the trajectory of the marble and determine the focal point for the intense acoustic wave to break up the marble.

Section IV – Computational Results

Signature Frequency

Based on the averaged frequency data, the threshold is set to 0.6 and normalized to the amplitude of the data for isosurface plotting. The location of the signature frequency is then determined at:

$$(kx, ky, kz) = (2, -1, 0)$$

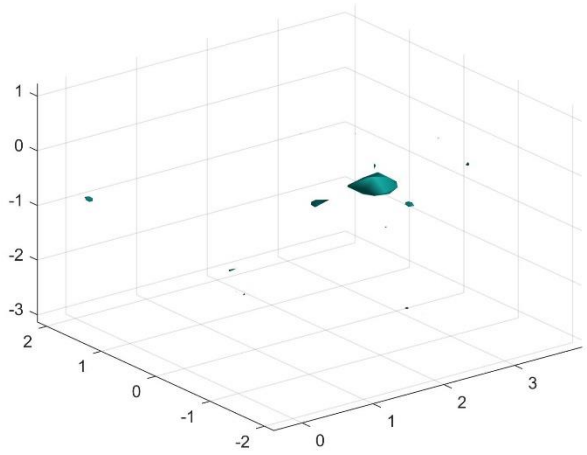

Fig. 2 Location of signature frequency at 0.6 threshold (left: x-axis, vertical: y-axis, right: z-axis)

Path of Marble

The filtered spatial data throughout 20 measurements are plotted at the given 0.4 magnitude using isosurface and stacked together to demonstrate the path of the marble.

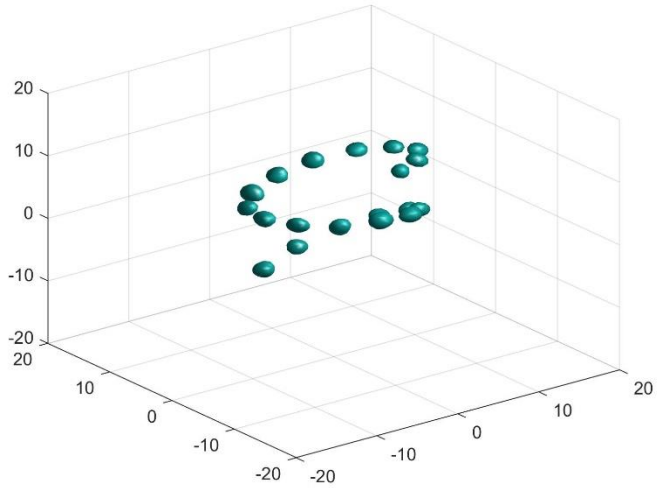

Fig. 3 Filtered spatial data showing the marble at 20 different locations

The trajectory of the marble is plotted as the pathway of the geometric center at each frame.

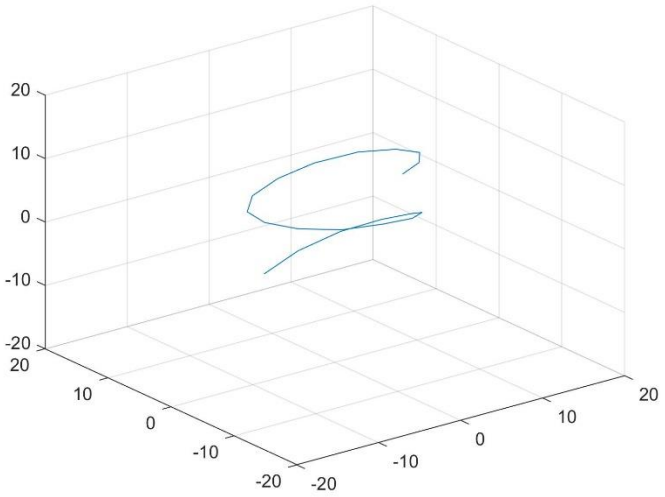

Fig. 4 Trajectory of the marble (from top to bottom)

Focal Point

The focal point at the last timeframe is determined from the trajectory to be:

$$(x, y, z) = (-5.4139, 4.1969, -6.0915)$$

Section V – Summary and Conclusions

The script developed for this homework successfully demonstrates the usage of Fourier transform in signal processing and noise reduction. From the original data, shown in Fig. 1, noisy data obscures the location of the marble and it is challenging to interpret the data. By using FFT, the script transforms the data into frequency domain, in which through averaging, noisy components are damped down, revealing the signature frequency of the marble, demonstrated in Fig. 2. After filtering, the data is converted back to time domain through inverse FFT to show the location of marble and its trajectory at each timeframe, as seen in Fig. 3 and Fig. 4.

Further improvement of the code can focus on a more automated process to identify the signature frequency location, as current script relies on manual interpretation. This improvement will also allow more precise positioning of the signature center, lowering the bandwidth required in filtering, and can potentially yield more accurate representation of the true marble location.

Appendix A – MATLAB functions used

- $meshgrid$: generate Cartesian grid from axial coordinates
- $fftn$: n dimensional Fourier transform (Fourier transform along all n directions)
- $ifftn$: n dimensional inverse Fourier transform
- $fftshift$: shift the zero-frequency component of Fourier coefficients to the center
- $ifftshift$: shift the coefficient back to MATLAB default (start from zero-frequency)
- $isosurface$: generate surface geometry at given value

Appendix B – MATLAB codes


```
clear; close all; clc;
```

Initialization

```
load Testdata
L=15; % spatial domain
n=64; % Fourier modes
x2=linspace(-L,L,n+1); x=x2(1:n); y=x; z=x; % create spatial coordinates
k=(2*pi/(2*L))*[0:(n/2-1) -n/2:-1]; ks=fftshift(k); % create frequency coordinates and shift
[X,Y,Z]=meshgrid(x,y,z); % meshgrid into 3D
[Kx,Ky,Kz]=meshgrid(ks,ks,ks);
```

Averaging spectrum

```
sum = zeros(size(n,n,n)); % preallocate summation matrix
for j=1:20
    Un(:,:,:)=reshape(Undata(j,:),n,n,n); % reshape data back to 3D form
    Unt = fftn(Un); % n-dimensional fft
    sum = sum+Unt;  % adding all measurements
end
ave = abs(sum)/j; % average the sum
aves = fftshift(ave); % fftshift to match freq coordinates

thresh_perc = 0.6; % determine the normalized threshold for signature ID

% plot the volume surface at preset threshold
close all, isosurface(Kx,Ky,Kz,abs(aves),thresh_perc*max(max(max(abs(aves)))))
axis([-10 10 -10 10 -10 10]), grid on,
drawnow
pause(2)

% estimate the location of signature frequency
kx0 = 2; ky0 = -1; kz0 = 0; % (2,-1,0)
```

Filtering

```
tau = 0.3; % determine Gaussian bandwidth
filter = exp(-((Kx-kx0).^2+(Ky-ky0).^2+(Kz-kz0).^2)*tau); % generate Gaussian filter centered at
signature frequency
traj = zeros(20,3); % preallocate marble trajectory
for j = 1:20
    Un(:,:,:)=reshape(Undata(j,:),n,n,n);
```

```
    Unt = fftn(Un);
    Unts = fftshift(Unt); % shift the transformed data to align w/ filter
    Untfs = filter.*Unts; % apply filter
    Untf = ifftshift(Untfs); % shift back to matlab default
    Unf = ifftn(Untf); % inverse fft back to spatial domain
    close all, isosurface(X,Y,Z,abs(Unf),0.4) % surface plot the filtered data
    [f,v] = isosurface(X,Y,Z,abs(Unf),.4);
    traj(j,:) = mean(v,1);
    axis([-20 20 -20 20 -20 20]), grid on,
    drawnow
    pause(1)
end
plot3(traj(:,1),traj(:,2),traj(:,3));
axis([-20 20 -20 20 -20 20]), grid on
```

ID focal point

```
focus = traj(end,:); % acoustic wave focus at the 20th measurement location
```

Published with MATLAB® R2019b