HW1_An Ultrasound Problem

Yanjie Liu

Abstract:

The goal of this report is to locate the specific target from noisy data signal. Specifically, the Fourier transform is used to transform from time domain to frequency domain and the noise of the data signal is averaged. Then the frequency of the target will be specified. Then the Gaussian filter is used to remove noise and strengthen the frequency of the target. Then the specific location of the target and its path will be determined in time domain.

Introduction and Overview

Suppose a dog swallowed a marble. By using ultrasound, data is obtained concerning the spatial variations in a small area of the intestines where the marble is suspected to be. Since the dog keeps moving, the internal fluid movement through the intestines generates highly noisy data.
In order to eliminate the effect of noisy data, we can average the frequency in frequency domain to strengthen the frequency generated by the marble. Then the Gaussian filter can be applied on the data to center around the target frequency. Finally, the converting frequency domain into time domain will lead to the specific location and path of the marble.

Theoretical Background

Fourier Transform

The Fourier transform converts a function of time (a signal) into its constituent frequencies which is consisted by a set of cosine and sine function. For a given function f(x), the Fourier transform of f(x) is

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

In this function, the function of frequency, F(x), is generated by the integral of the product of the Euler's formula and f(x) on an infinite domain. And its inverse function is

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

Although the transform is over the entire real line x $\in$ [− ∞ , ∞], whereas our computational domain is only over a finite domain x $\in$ [−L,L]. Further, the Kernel of the transform, exp($\pm$ikx), describes oscillatory behavior. Thus the Fourier transform is essentially an eigenfunction expansion over all continuous wavenumbers k. And once we are on a finite domain x $\in$ [−L,L], the continuous eigenfunction expansion becomes a discrete sum of eigenfunctions and associated eigenvalues.
In this report, we use Fast Fourier Transform, since fft has complexity O(NlogN) which is much faster than the previous algorithm with complexity O(N^2) and the accuracy properties of fft is better than the standard discretization schemes.

Gaussian Filter

In order to remove noise in frequency domain, we apply Gaussian filter in this case. The function of Gaussian filter is

$$e^{-\tau (k-k_0)^2} \qquad (3)$$

In this function, tau measures the bandwidth of the filter, and k is the wavenumber, and k0 is the center-frequency of the desire signal field. In this case, the data signal is three dimensional. Then the Gaussian filter is

$$e^{-\tau \ ((k-k_{x0})^2 + (\mathrm{k}-k_{y0})^2 + (k-k_{z0})^2)} \qquad (4)$$

In previous function, the kx0, ky0, kz0 represent the the center-frequency of the desire signal filed in each axis.

Algorithm Implementation and Development

Set-up

Before the start of algorithm, we need to load the ultrasound data into Matlab and define the spacial and frequency domain. In this case, we set the interval over x,y,z is [-15,15] in time domain. And we transform the time domain interval into the frequency domain interval by scaling by $\frac{2\pi}{2\mathrm{L}}$, which provides the interval from -2π to 2π . Then we use fftshift to swap the frequency domain, which serves to eliminate the effect of fft on axis.

Average the Spectrum

In order to discover the center frequency(kx0,ky0,kz0) for filter, we need to remove the noise from our ultrasound signal. After converting the raw data by fftn function for each detection, we need to sum up all of these frequencies and divide them by the detection time to get the average spectrum. Then we also need to normalize the data to make sure they are on a scale of 0 to 1.

Determine the Center Frequency

Since we just remove noise from the data and strengthen the frequency of the marble, the target frequency make up the largest portion of the time-averaged signal. Then we can obtain the target frequency by searching its indexes in each axis and mapping its indexes to ks.

Filter the data and collect the location of the target

In order to filter the data, we need to convert the raw signal from time domain into frequency domain with fftshift function. Then we need to apply the Gaussian filter equation to denoise the signal data. After applying inverse Fourier transform function (ifftn), we can convert the filtered data back to time domain. Due to the filter, the target frequency is the strongest, which allows us to find the position of the marble by finding the max signal.

Computational Result

After removing noise, applying Gaussian filter, and converting back to time domain, we can determine the final location of the marble:

$$x = -5.625 \quad y = 4.2188 \quad z = -6.0938$$

And the following figure is the position of the marble at each detection.

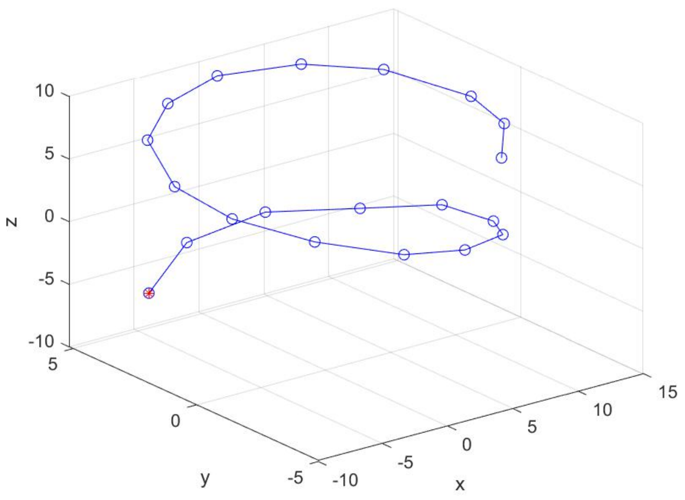

Figure 1: trajectory of the marble in spacial domain

Summary and Conclusions

In this report , we use Fourier transform, time-averaging method, and Gaussian filter to denoise the ultrasound data in frequency domain and locate the position of the marble in spacial domain.

Appendix A

[X,Y,Z] = meshgrid(x,y,z) returns 3-D grid coordinates defined by the vectors x, y, and z.

B = reshape(A,sz1,...,szN) reshapes A into a sz1-by-...-by-szN array where sz1,...,szN indicates the size of each dimension.

Y = fftn(X) returns the multidimensional Fourier transform of an N-D array using a fast Fourier transform algorithm.

X = ifftn(Y) returns the multidimensional discrete inverse Fourier transform of an N-D array using a fast Fourier transform algorithm.

Y = fftshift(X) rearranges a Fourier transform X by shifting the zero-frequency component to the center of the array.

[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.

plot3(X,Y,Z) plots coordinates in 3-D space.

description from https://www.mathworks.com/help/matlab/

Appendix B

```
clear; 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);

utsum = zeros(n,n,n);
for j=1:20
   Un(:,:,:)=reshape(Undata(j,:),n,n,n);
   utsum = utsum + fftn(Un);
end
utave = abs(fftshift(utsum))/20;
utnorm = utave/max(utave(:));


[ix,iy,iz] = ind2sub(size(utnorm), find(utnorm==max(utnorm(:))));
kx0 = Kx(ix,iy,iz);
ky0 = Ky(ix,iy,iz);
kz0 = Kz(ix,iy,iz);

tau=0.2;
filter  = exp(-tau * ((Kx - kx0).^2 + (Ky - ky0).^2 + (Kz - kz0).^2));
path = zeros(20,3);
for j=1:20
   Un(:,:,:)=reshape(Undata(j,:),n,n,n);
   final_fft = fftn(Un);
   final_unft = fftshift(final_fft);
   final_utf = filter.*final_unft;
   final_uf = abs(ifftn(final_utf));
   marble = final_uf/max(final_uf(:));
   [x,y,z] = ind2sub(size(marble),find(marble==max(marble(:))));
   path(j,1) = X(x,y,z);
   path(j,2) = Y(x,y,z);
   path(j,3) = Z(x,y,z);
end

plot3(path(:,1),path(:,2),path(:,3),'bo-');
xlabel('x')
ylabel('y')
zlabel('z')
grid on
hold on
plot3(path(20,1),path(20,2),path(20,3),'r*');
saveas(gcf,'marble.png')
```