Homework 1, AMATH 582

Kiana Mittelstaedt

Contents

Abstract

We have a noisy dataset containing the spatial variations of a marble inside of a dog's intestines. To save the dog, we need to calculate the trajectory of the marble so the vet knows where to focus an intense acoustic wave that will break up the marble. We apply the Fast Fourier Transform to decompose our noisy data into its frequency components. Next, we use an averaging process along with a Gaussian filter to denoise the data. Finally, we can transform our data back into its spatial domain to find the location of the marble in the dog's intestines.

1 Introduction and Overview

My dog Fluffy swallowed a marble and it has moved to his intenstines. Ultrasound data has been collected, showing the spatial variations in the area where the marble is located. However, since Fluffy is not stationary, the internal fluid movement has caused the data to be very noisy. To save Fluffy, I need to denoise this data, locate the marble, and calculate its trajectory. Reporting these results to Fluffy's vet will allow him to break up the marble by focusing an intense acoustic wave on the marble.

2 Theoretical Background

Integral to my process of locating the marble in Fluffy's intestines is the Fourier transform. This spectral transform will transform my spatial domain into a frequency domain via decomposition into sines and cosines. The Fourier transform on a function $f(x)$ is defined as $F(k) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{-ikx} f(x)dx$, where k is the frequency (or wave number). This integral gives the representation of $f(x)$ in all frequencies. The inverse Fourier Transform, which integrates over frequencies to get back to the spatial domain, is defined as $f(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{ikx} F(k)dk$.

Once the data is converted into the frequency domain, we will use an averaging technique to denoise our data. Our data has *white noise*, meaning that the noise affects all frequencies the same. Over multiple signal measurements, the *white noise* averages to zero. As such, we want to average over our frequency realizations to denoise and then produce a localized frequency signature.

With the localized signature in the frequency domain, we are able to locate where the signal was detected. Next, we will create a normally distributed Gaussian filter, $\mathcal{F}(k) = e^{(-\tau(k-k_0)^2)}$, where τ denotes the bandwidth and k_0 corresponds to the frequency signature.

We will take our data, Fourier transform it, and multiply it by the Gaussian filter. If there is a signal near the frequency signature, the Gaussian filter will isolate the signal and remove any undesired frequencies that were picked up in the ultrasound detection process. With our signal extracted in the frequency domain, we will then inverse Fourier transform back to the spatial domain so we can collect information about the location of the marble and save Fluffy.

3 Algorithm Implementation and Development

4 Computational Results

After averaging over our 20 frequency realizations, I produced the following graph:

The frequency signature (center frequency), detected through averaging over the spectrum, is $[1.8850, -1.0472, 0]$. This is the frequency that the marble is emitting in the Fourier domain.

After applying the Gaussian filter and filtering the data around the frequency signature above, I determined the path of the marble. The marble's path is shown in the graph below: Using this information, we can extract the location of the marble at the 20^{th} realization.

5 Summary and Conclusions

A MATLAB Functions Used

The MATLAB functions I utilized, defined according to the "*help*" feature in MATLAB:

- **abs**: returns the absolute value
- **fftn**: returns the N-dimensional discrete Fourier transform
- **fftshift**: swaps the first and third quadrants and the second and fourth quadrants of a matrix
- **ifftn**: returns the N-dimensional inverse discrete Fourier transform
- **ifftshift**: the inverse of the fast Fourier transform
- **ind2sub**: determines the equivalent subscript values corresponding to a given single index into an array
- **isosurface(X,Y,Z,V,ISOVALUE)**: computes isosurface geometry for data V at isosurface value ISOVALUE
- **linspace(X1,X2,N)**: generates N points between X1 and X2
- **max)**: returns the largest element
- **meshgrid(X)**: returns 3-dimensional grid coordinates with grid size length(X)-by-length(X)-by-length(X)
- **reshape**: reshapes an array
- **size**: returns the size of an array
- **zeros**: creates an array of zeros

B MATLAB Code

```
clear all; close all; clc;
load Testdata
%% Define Domain
L=15; % define computational spatial domain
n=64; % Fourier/frequency modes 2^n
x2=linspace(-L,L,n+1); % define spatial discretization
x=x2(1:n); y=x; z=x; % consider first n points:  periodic
k=(2*pi/(2*L))*[0:(n/2-1) -n/2:-1]; % wave #s of FFT, rescaled & shifted
ks=fftshift(k); % undo the FT shift, bring back to our domain
[X,Y,Z]=meshgrid(x,y,z); % measurement dimension
[Kx,Ky,Kz]=meshgrid(ks,ks,ks); % meshgrid frequency (wave #) domain
```

```
%% Average Over Data
Uave=zeros(n,n,n); % 3D matrix of zeros
for j=1:20
Un(:,:,:)=reshape(Undata(j,:),n,n,n);
UnT=fftn(Un); % Fourier transform
Uave=Uave+UnT; % add multi-dim ft (fftn) of current data to avg
end
Uave=fftshift(Uave)/20;
% extract max frequency signature
[maxavg,i]=max(abs(Uave(:)));
[p1,p2,p3]=ind2sub(size(Uave),i); % indexes
ksx=Kx(p1,p2,p3); % max k values
ksy=Ky(p1,p2,p3);
ksz=Kz(p1,p2,p3);
close all
figure(1) % plot in frequency domain
isosurface(Kx,Ky,Kz,abs(Uave)/max(abs(Uave(:))),0.6)
axis([-6 6 -6 6 -6 6]), grid on;
xlabel('Kx'); ylabel('Ky'); zlabel('Kz');
%% Apply 3D Gaussian Filter
% filter centered around max k values ksx, ksy, ksz
gfilter=exp(-0.5*((Kx-ksx).^2+(Ky-ksy).^2+(Kz-ksz).^2));
```