AMATH 582 Homework 1: Fluffy Goes To The Vet

Sara L. Daley

1/24/2020

Abstract

The goal of this paper is to isolate the frequency of an object that is moving and surrounded by noise. To do this we will average the signals in signal space (or frequency domain), and find the maximum to create a Gaussian filter. We then apply the filter to the signals in the time domain to isolate the signal of the object. We track the position of the object and report out it's final location.

1 Introduction and Overview

Everyone that has had the luxury of owning a pet dog knows at least once they do something (good or bad) that will stick with you forever. For Fluffy, it was when he swallowed a marble because he thought his toddler brother had dropped food. We took him to the vet so he could get an ultrasound; praying the vet bill did not exceed our mortgage that month (Fluffy was worth it though). The vet couldn't quite see the marble as Fluffy was a bit agitated. I offered to help for the sake of saving Fluffy. But if we're completely honest, it was to keep the cost of this adventure from exceeding half my yearly salary. The vet passed data to me that she knew contained the marble but couldn't pinpoint where the marble was.

In order to find the marble, I applied a filter to the data to isolate its frequency signature. With that I was able to trace it's path in time and provide exact coordinates to the vet. She put those into the machine to find the marble and focus acoustic waves to break it up.

2 Theoretical Background

The ultrasound emits wave pulses at a specific frequency. Those waves bounce off the marble and from this reflection we can find where the marble is. We do this using Fourier Transforms[1],

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

where $e^{\pm ikx}$ describes the oscillation. Fourier transforms can work in space or time by decomposing into Fourier modes (sines and cosines). We know from the Heisenberg Uncertainty Principle that we cannot know the exact position *and* velocity of a particle, but with FFT we can decompose a signal to isolate an item's position at a given time t [1]. There are two things with Fluffy that should be highlighted:

1. The data that came from the ultrasound is noisy.
2. The marble will be moving around inside Fluffy.

Thus, the noise needs to be filtered. Because we have more than one measurement (more than one signal over a length of time), we can average the measurements to make the noise average to zero where there is no object. The white noise will have zero mean; the noise will drop when the signals are averaged. In other words, we can average all the signals we have to find the center frequency of the marble. We then use the center frequency to create a filter. We have to create a filter in the signal space and apply it in the time space as our signal is moving. Once the filter is applied, we can extract the position of the marble at each time t and tell the vet where it is at the last time step. I chose to use the Gaussian filter as it is an ideal filter for the time domain[2], and is represented in 3D as,

$$\mathcal{F}(k) = \exp(-\tau((k(x) - k(x)_0)^2 + (k(y) - k(y)_0)^2 + (k(z) - k(z)_0)^2) \tag{2}$$

where τ measures the bandwidth of the filter, k is the wavenumber, and $k_0 = k$ is the center frequency.

3 Algorithm Implementation and Development

Please refer to Algorithm 1

Algorithm 1: Filter Signal Algorithm

Import data from `Testdata.mat`
Set *Computational/Spatial Doman*, *Fourier Modes*
Define domain discretization while considering periodicity (shifted)
fftshift domain to provide unshifted counterpart
for $j = 1 : length(Testdata)$ **do**
  reshape *Undata* (signal) to Fourier modes
  fft signal
end for
Normalize signal by taking absolute value of the shift
Isolate maximum of normalized signal
Translate the maximum into Frequency Space (now Frequency Center)
Create filter with center (x_0, y_0, z_0) in Frequency Space coordinates
for $j = 1 : length(Testdata)$ **do**
  reshape again
  fft signal again
  shift signal
  apply filter: filter * signal
  unshift filtered signal
  inverse ft to gain true signal, U
  Isolate maximum of U and translate to Time Space coordinates
  Save the positions
end for
plot

4 Computational Results

Frequency signature of the marble has center (kx, ky, kz):

$(1.884955592153876, 1.047197551196598, 0)$. Where (kx, ky, kz) are coordinates in the signal space.

Marble at 20^{th} (final) position: (x, y, z): $(-5.625000, 4.218750, -6.093750)$. Where (x, y, z) are coordinates in the time space.

5 Summary and Conclusions

In summary we were able to find the position of the marble at the final time-step given by the vet. We thus passed those coordinates on and with that isolation they could focus the acoustic waves at that particular point to break up the marble. We were able to help save Fluffy and keep the vet bill to be below the cost of my yearly salary! Through this process we took the noisy data, averaged the signals, and found the center frequency from the maximum of the averaged signals in the signal domain. We then applied that filter in the time domain to isolate the frequency of the marble and track it's position.

References

[1] Jose Nathan Kutz. *Data-driven modeling & scientific computation: methods for complex systems & big data*, Oxford University Press, 2013.

[2] `https://en.wikipedia.org/wiki/Gaussian_filter`.

[3] `https://www.mathworks.com/help/matlab/index.html`

Appendices

Please refer to my Github repo for all code and related documents.

A MATLAB Functions

- `load Testdata`

  loads file `Testdata` into an array

- `x2 = linspace(-spatial domain, spatial domain, Fourier modes + 1)`

  creates lineraly spaced vector from `-spatial domain` to `spatial domain` with `Fourier modes+1 points`. The endpoints are included [3]

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

  replicates grid vectors `(x,y,z)` to produce coordinates on 3D rectangular grid `(X,Y,Z)`. columns of X, rows of Y, pages of Z [3]

- `ii = zeros(1,n)`

  creates a vector of 0's of length n (size 1xn). Speeds up processing of loops as it sets the size of the vector.

- `Un(:,:,:)  = reshape(Undata(j,:),n,n,n)`

  reshapes elements `Undata(row j, all columns)` to be matrix of size n by n by n. n*n*n must be = NUMEL(Undata(j,:)) [3]

- `ks = fftshift(k)`

  shift the frequency component to the center of the spectrum. [3]

  To perform this in domain ≥ 2, fftshift(k,domain).

- `Utn = fftn(x)`

  N^{th} dimensional discrete Fourier Transform.

  For 2D, fft(x). If x is a vector, output contains same orientation.

- `[V I] = max(U(:))`

  returns the indices of the maximum values in vector I. V is the value. [3]

- `[kxx, kyy, kzz] = ind2sub(size(U), I)`

  this transforms the indices passed from the vector I into coordinates within space `[kxx,kyy,kzz]` [3]

- `Uf = ifftshift(Utf,dimension)`

  performs the inverse shift. can omit dimension if in 2D.

- `U = ifftn(Uf)`

  performs the inverse Fourier transform. if in 2D can use ifft(Uf).

B MATLAB Codes

Table of Contents

Load and Initialize

```
clear all; close all; clc;

% 'https://drive.google.com/drive/
folders/1SQ77P5t5RUWCSucmk4jPFbufFMX8VrJG'
load Testdata

L=15; % spatial domain
n=64; % Fourier modes
x2=linspace(-L,L,n+1); % Include +1 beyond modes
x=x2(1:n); % Exclude last point due to periodicity
y=x; z=x;
k=(2*pi/(2*L))*[0:(n/2-1) -n/2:-1]; % Freq space rescaled to 2pi
 domain
ks=fftshift(k); % Unshift k

[X,Y,Z]=meshgrid(x,y,z); % Time Domain
[Kx,Ky,Kz]=meshgrid(ks,ks,ks); % Transform Space

Utave = zeros(n,n,n);
ii = zeros(1,length(Undata(:,1)));
jj = zeros(1,length(Undata(:,1)));
kk = zeros(1,length(Undata(:,1)));
```

Average Signals (as the target is moving)

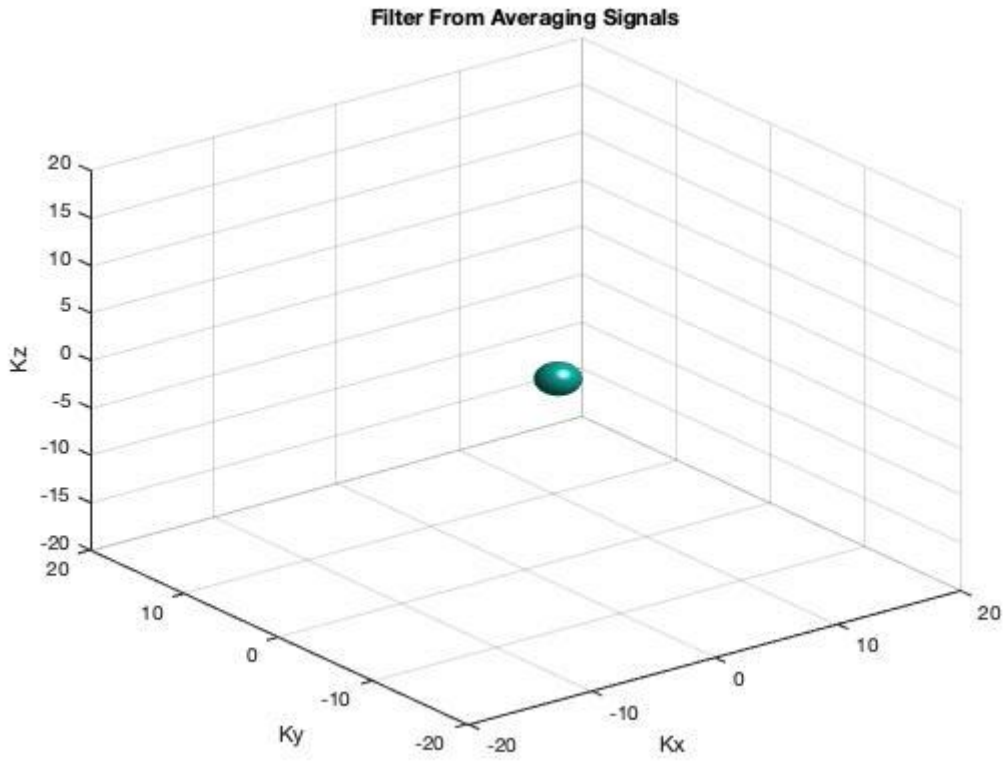
```
for j=1:length(Undata(:,1))
    Un(:,:,:)=reshape(Undata(j,:),n,n,n);
    Utn = fftn(Un);
    Utave = Utave+Utn;
end

Uave = abs(fftshift(Utave))/length(Undata(:,1));
[M, I] = max(Uave(:));
[kxx, kyy, kzz] = ind2sub(size(Uave),I);
Sig_cent = [Kx(kxx,kyy,kzz),Ky(kxx,kyy,kzz),Kz(kxx,kyy,kzz)]; % Freq
 Center

filter = exp(-0.6*((Kx-Sig_cent(1)).^2 + ...
    (Ky-Sig_cent(2)).^2 + (Kz-Sig_cent(3)).^2)); % 3D Gaussian
```

Plot Filter

```
isosurface(Kx,Ky,Kz,filter)
xlabel('Kx');ylabel('Ky');zlabel('Kz');
title('Filter From Averaging Signals');
axis([-20 20 -20 20 -20 20]), grid on; drawnow
```



Transform back to the Time Domain, Isolate target trajectory

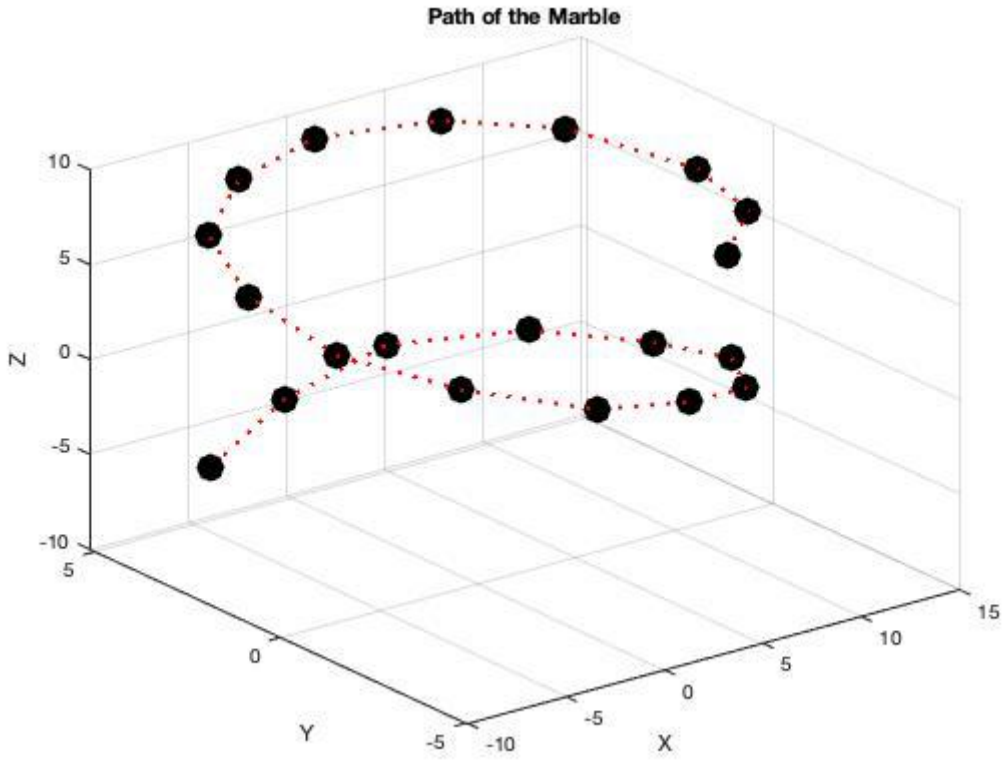
```
for j=1:length(Undata(:,1))
    Un(:,:,:)=reshape(Undata(j,:),n,n,n);
    Utn = fftn(Un);
    Utns = fftshift(Utn);
    Utf = filter .* Utns;
    Uf = ifftshift(Utf,3);
    U = ifftn(Uf);
    [N, K] = max(U(:));
    [a, b, c] = ind2sub(size(Utn),K);
    ii(j) = X(a,b,c);
    jj(j) = Y(a,b,c);
    kk(j) = Z(a,b,c);
end
```

Plot target and state position at final time

```
figure(2);
plot3(ii,jj,kk,'r:','linewidth',[2]);
hold on
plot3(ii,jj,kk,'k.','MarkerSize',[48]);
title('Path of the Marble')
xlabel('X');ylabel('Y');zlabel('Z'); grid on;

marble = [ii(length(Undata(:,1))), jj(length(Undata(:,1))), ...
    kk(length(Undata(:,1)))];
fprintf('Focus acoustic wave (x,y,z): %f %f %f \n',marble)
```

```
Focus acoustic wave (x,y,z): -5.625000 4.218750 -6.093750
```



Published with MATLAB® R2018b