Homework 1: An Ultrasound Problem

AMATH 482: Computational Methods for Data Analysis

Rosemichelle Marzan

January 24, 2020

Abstract **– This report demonstrates how the fast Fourier transform and Gaussian-based spectral filtering can be applied to a hypothetical, yet relatively realistic, scenario. The proposed solution denoises the given noisy dataset using the averaging method to determine the object's frequency signature, which is then used as the center frequency in the applied Gaussian filter. Applying the inverse fast Fourier transform to the filtered signal allows the object to be tracked spatially over time.**

1 Introduction and Overview

The scenario is as follows:

> A dog has swallowed a marble which has traveled into the intestines. Ultrasound data for 20 different measurements in time contain information about the spatial variations within the dog's intestines. However, due to the dog's movements, intestinal fluids add random noise to the data. The location and trajectory of the marble must be computed to determine where an intense acoustic wave should be focused to destroy the marble at the 20th data measurement.

To solve this problem, the spectrum must first be averaged to determine the frequency signature of the marble. The frequency signature will be used as the center frequency around which the data will then be filtered to remove the noise and determine the path of the marble. Finally, the marble's coordinates at the final measurement will be determined. These steps will be performed in MATLAB R2019b. The associated code can be found in **Appendix B**.

2 Theoretical Background

2.1 The Fourier Transform and the Fast-Fourier Transform (FFT)

The Fourier transform is an important technique used in the engineering and science fields to analyze signals. It works by decomposing a signal as a function of time into its frequency components. The Fourier transform equation essentially takes the integral of Euler's formula multiplied by the signal function over an infinite domain (1).

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

Whereas the Fourier transform transforms the signal from the time domain to the frequency domain, the inverse Fourier transform does the reverse (2).

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

The Fast-Fourier transform (FFT) is an algorithm for the discrete Fourier transform derived from the Fourier transform for continuous functions. The FFT is useful for signals obtained from the real world, in which data collection can be limited by the recording device's sampling frequency. The FFT operates on $\mathcal{O}(N \log N)$, N being the sampling frequency (hence, the FFT is *discrete*). This makes it considerably much faster than the discrete Fourier transform, which operates on $\mathcal{O}(N^2)$, for large N.

There is also an inverse fast Fourier transform (IFFT) that can transform the frequency function to its corresponding time domain signal. In the context of the problem presented, the IFFT can be useful for obtaining the filtered time domain signal where the marble's location can be determined spatially over time.

2.2 Denoising data by averaging

Averaging is a useful technique for filtering signals with white noise, i.e. noise that affects all frequencies equally. Since the noise in the data is random, it is appropriate to implement the averaging method to identify the marble's frequency signature. In doing so, we can apply what we know about modeling white noise: adding a normally distributed random variable with zero mean.

It can be helpful to understand why this method is useful by thinking about how synthetic noisy data can be created. Say our signal is a simple sine wave with a 1 Hz frequency. We can generate synthetic noise by adding a value from a standard normal distribution whose mean is zero. If the noise in the data is random, then as the number of measurements approaches infinity, the average of the noise in the measurements should be zero while the signal is preserved.

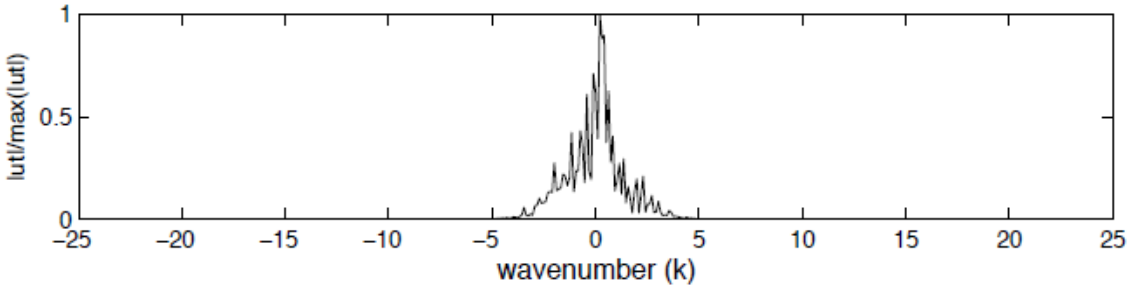

Figure 1: An example of a Gaussian filtered signal in the frequency domain. Note that because the center frequency is the frequency of interest, it has the highest amplitude; while frequencies higher than $k = 5$ and lower than $k =$ -5 are attenuated.

2.3 Spectral Filtering and the Gaussian Filter

Spectral filtering is a noise-filtering technique that is applied to the signal in the frequency domain to amplify frequencies of interest (e.g. the marble). There are generally three types of filters: low-pass, high-pass, and band-pass, whose names indicate that the frequencies below, above, or within a certain threshold are retained, while everything else is attenuated. The Gaussian filter is an example of a low-pass filter because higher frequencies, or those further away from the center frequency (the frequency of interest), are attenuated (Figure 1). Below is the equation for the Gaussian filter, where τ is the bandwidth of the filter, k is the frequency wavenumber, and k_0 is the center frequency (3).

$$F(k) = e^{-\tau(k-k_0)^2} \tag{3}$$

3 Algorithm Implementation and Development

3.1 Defining the domains

Code Lines 14-22

In the time domain, the problem occurs within a three-dimensional space, whose axes extend from -15 to 15 in the x, y, and z directions, with 64 points each. The frequency domain, k, is rescaled by $\frac{2\pi}{2L}$ because the FFT assumes 2π periodic signals. Applying the `meshgrid` command to the frequency space allows the filter function to have a three-dimensional form.

3.2 Averaging the spectrum

Code Lines 24-29

The ultrasound data in its raw form (`Undata`) is a 20 x 262144 matrix. It is reshaped into 20 rows of 3D matrices (64 x 64 x 64) that describe the intestinal spatial variation. The FFT is applied to the 3D matrix at each measurement. Averaging must occur in the frequency domain, not the time domain, because the white noise affects the signal in the frequency domain. The averaged signal data is then computed by taking the absolute value of the sum of each transformed signal, then divided by the number of samples (i.e. measurements). It is important that the absolute value of all the points is used to find the average because the maximum of this will be used to find the center frequency (i.e. frequency signature) of the marble.

3.3 Determining the frequency signature of the marble

Code Lines 31-36

The marble will generate a frequency signature distinct from other artifacts within the intestines. This can be identified by a significantly large peak in the averaged ultrasound data. The `max` command will yield the maximum amplitude value in the averaged spectrum as well as its index within the "flattened" (20 x 262144 matrix) data set. Since this index is in the un-reshaped (non-3D) form, it must be correlated to the x, y, and z dimensions (64 x 64 x 64 space). Recall that a 3D filter will be applied to the data. Thus, the spatial coordinates must be correlated to the x, y, and z dimensions of the k frequency space to yield the x, y, and z components of the center frequency value, k_0.

3.4 Filtering the data

Code Lines 38-53
The x, y, and z components of k_0 are each incorporated in the Gaussian filter function. Using a `for` loop, the marble's location at each measurement is saved as an (x,y,z) coordinate into each row of the `marbleloc` matrix. For each row in the raw data, `Undata`, the data is reshaped then applied with the FFT. Now in frequency space, the transformed signal is then multiplied by the Gaussian filter. The filtered signal, which contains the amplified marble frequency, is then reverted to the time domain using IFFT. By doing so, the marble's spatial coordinates at each time point can be obtained. To achieve this, first, the index of the maximum value in the filtered time-domain signal is determined, then, its corresponding (x,y,z) spatial coordinates is found using the `ind2sub` command.

4 Computational Results

4.1 The marble's frequency signature

The marble's frequency signature (center frequency), k_0, is 1.885 in the x-dimension, -1.0472 in the y-dimension, and 0 in the z-dimension.

4.2 The marble's path over time

Figure 2 shows that over the course of the 20 measurements, the marble had a spiral path within the dog's intestine. The marble's coordinates at each measurement were stored in the matrix, `marbleloc`. The values in the last row indicate that by the 20th measurement, the marble is located at x = -5.625, y = 4.219, z = -6.094, which is also where the intense acoustic wave should be focused to break up the marble.

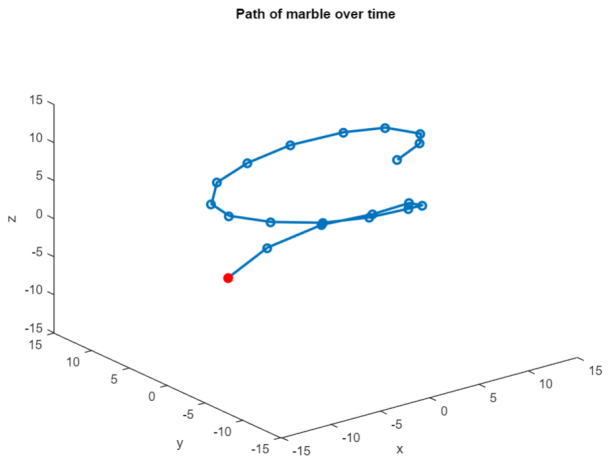

Figure 2: Marble location at each measurement. Final location (20th measurement) marked with a red filled circle.

5 Summary and Conclusions

There were four key steps to solving this problem: defining the domains, averaging the spectrum, finding the marble frequency signature, and filtering the data. Defining the domain ensured that there was agreement between the time and frequency domains such that the FFT and IFFT can be correctly applied. Averaging the spectrum allowed just enough of the random noise to be removed to determine the marble's frequency signature based on its large frequency peak. Determining the marble's frequency signature required reshaping the raw data and correlating indices to the spatial domain and its corresponding coordinates in the frequency domain. Finally, the data was filtered around the marble's frequency as the center frequency to further amplify the marble's signal, which helped pinpoint its location at each measurement back in the time domain.

The marble was able to be broken up at **(-5.625, 4.219, -6.094)** in the 20th measurement, saving the dog's life.

6 Appendix A. MATLAB functions used and brief implementation explanation

- **meshgrid**: Created 3D grids for the x, y, and z dimensions in both the time domain and frequency domain.
- **fftn**: Multidimensional fast Fourier transform algorithm.
- **reshape**: Mapped the points from the "flattened" `Undata` signal (raw noisy data) to a three-dimensional matrix.
- **ind2sub**: Correlated the "flattened" index (e.g. index of the maximum value in the averaged signal) to an x-, y-, and z-coordinate in the spatial domain.
- **ifftn**: Multidimensional inverse fast Fourier transform algorithm.
- **plot3**: Plots coordinates from the x, y, and z dimensions onto a 3D space.

7 Appendix B. MATLAB code

```
clear all; close all; clc;
load HW1_Testdata.mat

%%% Name: ROSEMICHELLE MARZAN
%%% Course: AMATH 482
%%% Homework 1, Due 1/24/2019

% VARIABLE NAMES FOR SIGNAL
% Undata = raw (noisy) data
% Unt = noisy data in the frequency domain
% Unft = filtered data in frequency domain
% Unf = filtered data in time domain

% setting up the domains
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];
[X,Y,Z] = meshgrid(x,y,z);
[Kx,Ky,Kz] = meshgrid(k,k,k);
samples = 20; % num of measurements

% averaging the spectrum
```

```
ave = zeros(n,n,n);
for i = 1:samples
    ave = ave + fftn(reshape(Undata(i,:),n,n,n));
end
ave = abs(ave)/samples;

% find the marble's frequency signature
[val,index] = max(ave(:));
[xind,yind,zind] = ind2sub([n,n,n],index);
K0x = Kx(xind,yind,zind);
K0y = Ky(xind,yind,zind);
K0z = Kz(xind,yind,zind);

% filtering noisy data at the center frequency
tau = 0.2;
filter = exp(-tau.*((Kx-K0x).^2 + (Ky-K0y).^2 + (Kz-K0z).^2));
% marble location as (x,y,z) coordinate
marbleloc = zeros(samples,3);

for i = 1:samples
    Unt = fftn(reshape(Undata(i,:),[n,n,n]));
    Unft = Unt.*filter;
    Unf = ifftn(Unft);
    [val,index] = max(Unf(:));
    [xind, yind, zind] = ind2sub([n,n,n],index);
    marbleloc(i,1) = X(xind,yind,zind); % x-coordinate
    marbleloc(i,2) = Y(xind,yind,zind); % y-coordinate
    marbleloc(i,3) = Z(xind,yind,zind); % z-coordinate
end

plot3(marbleloc(:,1),marbleloc(:,2),marbleloc(:,3),'-o','Linewidth',2);
title('Path of marble over time')
xlabel('x');
ylabel('y');
zlabel('z');
axis([-15 15 -15 15 -15 15])
hold on
plot3(marbleloc(end,1),marbleloc(end,2),marbleloc(end,3),'ro','MarkerFaceColor','r','
    Linewidth',2);

% marble location at 20th measurement
final = marbleloc(end,:);
```