Application of Fourier Transforms to Filter Acoustic Data

Christine M. Baker, `cmbaker94`

Jan 27, 2021

Abstract

Fourier transforms are a powerful tool for analyzing and filtering time and spatial data. This technique is applied to detect a submarine's emitted acoustic frequency from a broad spectrum recording of acoustics. A Gaussian filter of the spectra is applied to data obtained over a 24-hour time period to remove noise and track the submarine's location. The path of the submarine is identified to decide where to send a P-8 Orion sub-tracking aircraft to follow the submarine in half-hour increments.

1 Introduction and Overview

Spectral transforms are powerful and efficient techniques to analyze scientific and engineering data. In addition to analyzing variance of measurements as a function of spatial or temporal frequency, spectral transforms and inverse transforms can be used as a robust method to filter data by frequency (Kutz, 2013). Fourier transforms, a type of spectral transform, represents data as sums of cosines and sines. A Fast-Fourier transform (FFT) is an efficient method to compute a spectrum of data and can be applied in multiple dimensions.

Here, FFTs are used to detect a moving submarine that emits an unknown acoustic frequency using noisy acoustic data. Data over a 24-hour period in half-hour increments is provided from a broad spectrum recording of acoustics. To locate and identify the trajectory of the submarine, the frequency signature (center frequency) generated by the submarine is identified by finding the peak of the time-averaged three-dimensional spectra computed from the acoustic signal. A Gaussian filter centered at the peak frequencies emitted by the submarine is applied in frequency-space to denoise the data, and the data is then inverse transformed to plot the trajectory of the submarine, chosen as the maximum of the data at each time step. The horizontal (x and y) coordinates of the submarine can be sent to my P-8 Orion sub-tracking aircraft to follow the submarine. The theoretical background is presented in Section 2, algorithm implementation and development in Section 3, computational results in Section 4, and summary and conclusions in Section 5. The MATLAB functions used and code are presented in Appendix A and B, respectively.

2 Theoretical Background

Fourier introduced a notion that a function, $f(x)$, can be represented by a trigonometric series of sines and consines, such that for $-\pi < x \leq \pi$:

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

which produces 2π-periodic functions. The Fourier Transform defined over an entire line $-\infty \leq x \leq \infty$ is defined as:

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

and the inverse Fourier Transform is defined as:

$$f(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{ikx} F(k)dk \tag{3}$$

The FFT, a routine developed to perform forward and backward Fourier transforms, requires $O(N \log N)$ operation count to solve a system (Kutz, 2013). The FFT routine has excellent accuracy properties and transforms over a finite interval interval $-L \leq x \leq L$ with periodic boundary conditions and are discretized into 2^n points. Band-pass filtering, noise attenuation via frequency filtering, can be applied to data by applying a filter to Fourier transformed data in frequency-space and then, inverse transforming the data back to time-space. Step-wise filters (*i.e.,* box-car filters) are simple but generate ringing in frequency-space due to their sharp edges, whereas Gaussian filters have much smoother spectra properties and can be generated as:

$$\mathcal{F}(k) = \exp(-\tau(k - k_0)^2) \tag{4}$$

where τ is the bandwidth of the filter and k is the wavenumber. When $\mathcal{F}(k)$ is applied to the frequencies centered around 0 (*i.e.,* $k_0 = 0$), the filter will act as a low-pass filter, where the high-frequency components in a system are eliminated. Similarly, the k_0 can be altered to filter around specified frequencies, such as the frequencies emitted by submarines. The bandwidth, τ, can be adjusted according to the range of frequencies around the center-frequency that are of interest. Filtering can significantly remove noise and improve the detection of a signal (Kutz, 2013).

Another technique to remove white-noise in a time series is to compute the FFT of small windows of data and average the spectra. If the signal is present throughout a data record, this will average out the white-noise in a signal and retain signals that are not noise when appropriate data lengths are chosen. There are drawbacks to this approach such as the requirement to have a long enough data set to have multiple realizations of the signal of interest, *i.e.,* low-frequency signals in the data may be missed if the subsection length of the data are not long enough to resolve these frequencies.

3 Algorithm Implementation and Development

A combination of time-averaging spectra and filtering in frequency-space are used to identify the submarine emitted frequencies and track the submarine path as shown in Appendix B. The data provided is from a broad spectrum recording of acoustics in the Puget Sound over a 24-hour period in half-hour increments (49 time steps). The spatial domain is set to $L = 10$ with the units are an unknown length scale and the Fourier modes is specified as 64. The x, y, z-Cartesian coordinates have a equivalent range from $-L$ to L and number of nodes (64), *i.e.,* the data is 64x64x64 per time step. The wavenumbers are defined and rescaled by $2\pi/L$, because the FFT assumes a 2π periodic signal. A three-dimensional grid is defined for the Cartesian and wavenumber x, y, z components (K_x, K_y, K_z).

The frequency signature (center frequency) generated by the submarine is identified by time-averaging the three-dimensional spectra and finding the x, y, z frequencies at the maximum of the spectra. At each time step, the data is reshaped to the spacial coordinates, a three-dimensional FFT is computed for a given time step, and the spectra is added to a cumulative spectra. After all the time steps are looped through, the cumulative spectra is divided by the number of time steps to obtain the average spectrum. The center frequency in all dimensions is chosen by picking the maximum of the spectrum in all dimensions and finding the related matrix index the maximum value. The related wavenumber in three-dimensions (K_x, K_y, K_z) is selected.

Now that the center frequency generated by the submarine is determined, the data can be filtered around the center frequency to denoise the data and determine the path of the submarine. A three-dimensional Gaussian filter is generated with the form:

$$\mathcal{F}(k) = \exp(-\tau \left[(k - k_{x,0})^2 + (k - k_{y,0})^2 + (k - k_{z,0})^2\right] \tag{5}$$

where $k_{x,0}, k_{y,0}, k_{z,0}$ is the center frequency and τ is the bandwidth of the filter that was selected based on visual identification through a trial-and-error approach. Once again, at each time step, the FFT of the data is computed and multiplied by the filter. Then, the spectra is inverse transformed and the Cartesian coordinates of the maximum data value are extracted. The Cartesian coordinates of the maximum data value are stored for all time steps to create a 3x49 matrix of the x, y, z path of the submarine over the 24-hour period.

Lastly, the path of the P-8 Orion sub-tracking aircraft is defined as the horizontal (x, y) positions of the submarine.

4 Computational Results

The time-averaged raw signal in Cartesian coordinates has high return in a spiral shape (Figure 1a, for normalized values greater than 0.75); however, the signal is noisy and spectral analysis is required to confirm that this frequency signal has the same characteristics at each time step. The time-averaged unfiltered spectra over the 24-hour period reduced the noise at other frequencies while retaining frequencies centered around acoustic admissions of the new class of submarine. The normalized time-averaged three-dimensional spectra had the maximum variance at $(K_{x,0}, K_{y,0}, K_{z,0}) = (5.34, -6.91, 2.20)\ 2\pi/L$, where L indicates some length unit (Figure 2 for normalized values greater than 0.5). At each time step, a three-dimensional Gaussian filter centered around the center frequencies with a bandwidth of $\tau = 0.2$ was applied to the spectra and an inverse Fourier transform was performed to reconstruct a cleaner data signal. The time-averaged band-passed data in Cartesian coordinates retains the spiral shape observed in the unfiltered data and removes the noise (Figure 1b, for normalized values greater than 0.75). The maximum filtered return at each time step represents the path of the submarine (Figure 3, time represented by circle color). My P-8 Orion sub-tracking aircraft will be sent to the same horizontal coordinates (x, y) as the submarine path (Table 1) at elevations above the water surface.

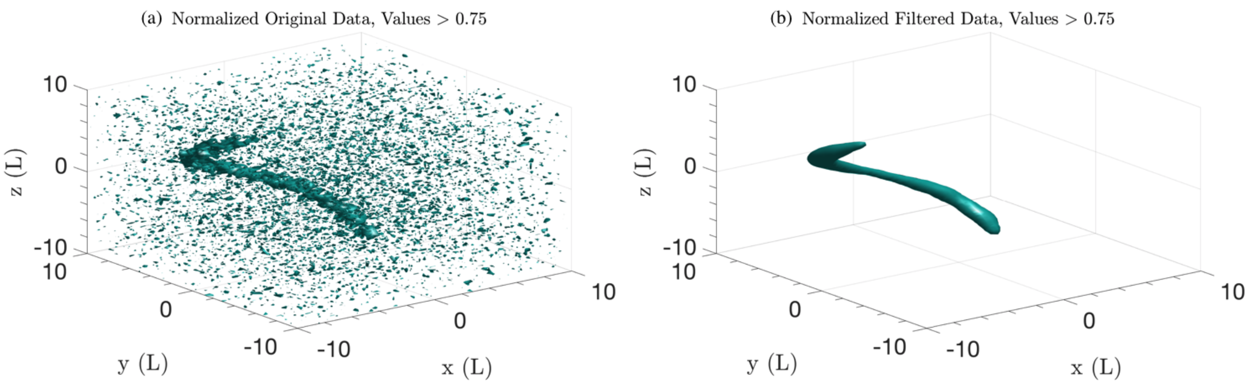

Figure 1: Normalized (a) original and (b) filtered time-averaged broad spectrum recordings of acoustic data over 24 hours in the horizontal (x, y) and vertical (z) dimension with unknown length units, L. The submarine path is evident by the denser region of the isosurface of normalized values greater than 0.75, which is noisy in the raw data (a) and clean in the filtered data (b) To produce the filtered data, a Gaussian filter centered at the peak frequency was applied in frequency-space.

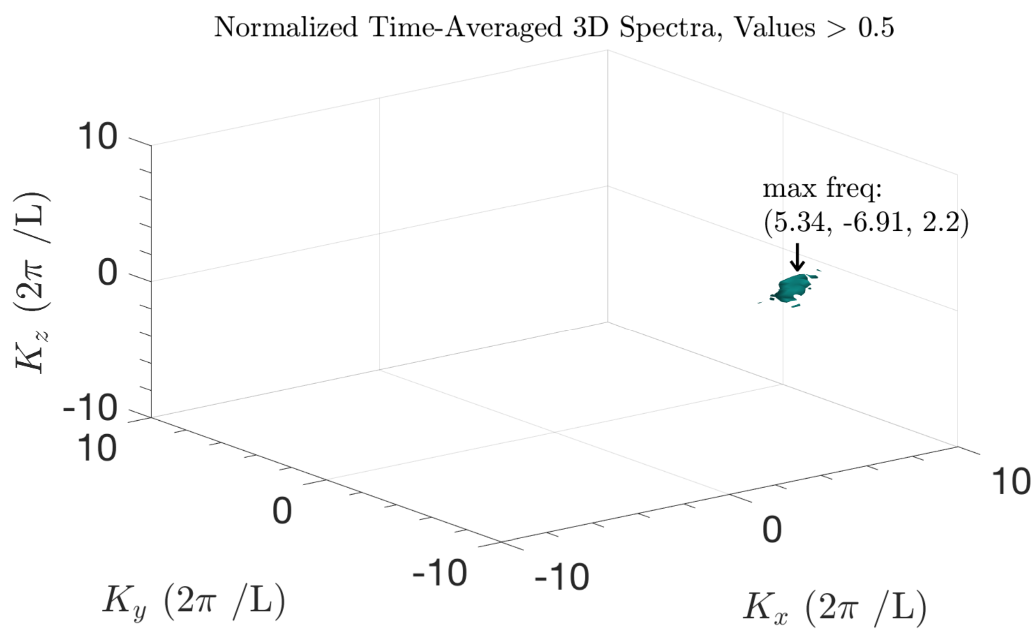

Figure 2: Normalized time-averaged three-dimensional spectra of 49 half-hour samples of broad spectrum recordings of acoustics. Normalized values greater that 0.5 are shown in the iso-surface in wavenumber space (K_x, K_y, K_z) with the peak frequencies of $(K_{x,0}, K_{y,0}, K_{z,0}) = (5.34, -6.91, 2.20)\ 2\pi/L$.

Table 1: The coordinates where I should send my P-8 Orion sub-tracking aircraft to follow the submarine. Time (hours) is the time at the beginning of each half hour of the data collection.

Time (hours)	**x**	**y**
0	3.1250	0
0.5	3.1250	0.3125
1	3.1250	0.6250
1.5	3.1250	1.2500
2	3.1250	1.5625
2.5	3.1250	1.8750
3	3.1250	2.1875
3.5	3.1250	2.5000
4	3.1250	2.8125
4.5	2.8125	3.1250
5	2.8125	3.4375
5.5	2.5000	3.7500
6	2.1875	4.0625
6.5	1.8750	4.3750
7	1.8750	4.6875
7.5	1.5625	5.0000
8	1.2500	5.0000
8.5	0.6250	5.3125
9	0.3125	5.3125
9.5	0	5.6250
10	-0.6250	5.6250
10.5	-0.9375	5.9375
11	-1.2500	5.9375
11.5	-1.8750	5.9375
12	-2.1875	5.9375
12.5	-2.8125	5.9375
13	-3.1250	5.9375
13.5	-3.4375	5.9375
14	-4.0625	5.9375
14.5	-4.3750	5.9375
15	-4.6875	5.6250
15.5	-5.3125	5.6250
16	-5.6250	5.3125
16.5	-5.9375	5.3125
17	-5.9375	5.0000
17.5	-6.2500	5.0000
18	-6.5625	4.6875
18.5	-6.5625	4.3750
19	-6.8750	4.0625
19.5	-6.8750	3.7500
20	-6.8750	3.4375
20.5	-6.8750	3.4375
21	-6.8750	2.8125
21.5	-6.5625	2.5000
22	-6.2500	2.1875
22.5	-6.2500	1.8750
23	-5.9375	1.5625
23.5	-5.3125	1.2500
24	-5.0000	0.9375

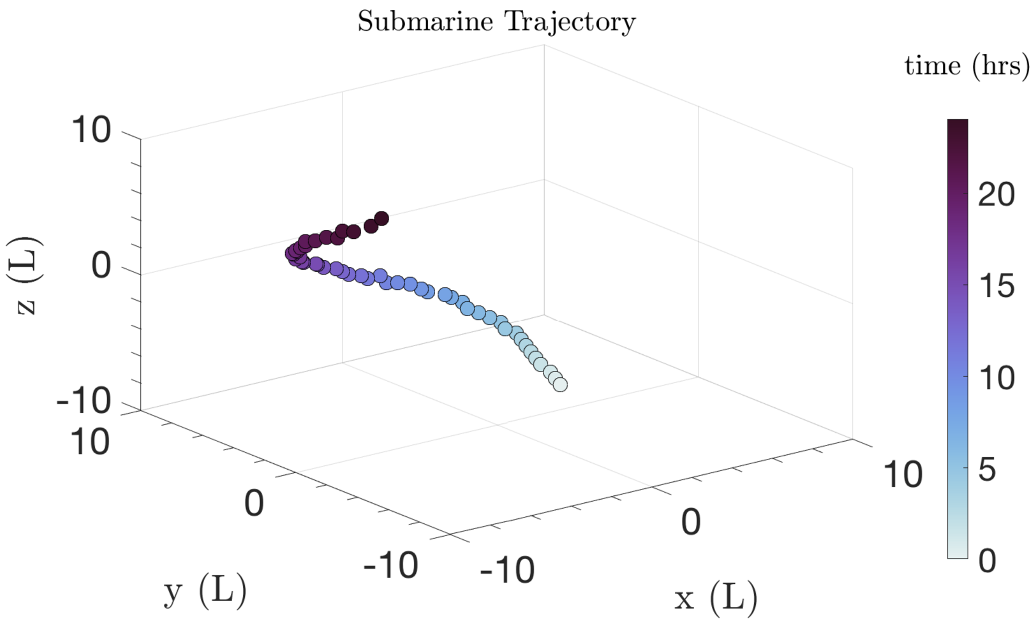

Figure 3: The submarine trajectory in the horizontal (x, y) and vertical (z) direction over a 24-hour time period (circle colors). The location is the maximum return of the filtered data.

5 Summary and Conclusions

Spectral analysis is a powerful tool to identify signals in time series that are stationary in frequency space from a noisy signal or to remove other frequencies, which could be noise or other underwater features for the submarine scenario. The data was filtered around the center frequency using a Gaussian filter. The filtered spectra was inverse transformed and the maximum returns can be identified to track the submarine path and send a tracking aircraft above the water.

Appendix

A MATLAB functions used and brief implementation explanation

- `abs`: returns the absolute value of the input element
- `axis`: indicate the limits for the current plotted axes
- `cmocean`: package of colormaps (used for the scatter plot)
- `exp`: the expoential e^x for each element in the array (used for the Gaussian filter)
- `fftn`: N-D fast Fourier transform, returns a multidimensional Fourier transform of an N-D array using a fast Fourier transform
- `fftshift`: used to rearrange zero-frequency component to the center of the spectrum

- `figure`: open new figure
- `grid`: add grid lines to a figure
- `ifftn`: compute the multidimensional discrete inverse Fourier transform
- `ind2sub`: convert linear indices to subscript (using the index of an array, find index in matrix)
- `isosurface`: plot extracted data about threshold from volume data
- `*label`: label the x, y, z axes of plots
- `length`: find the length of the largest array dimension (grab the time-dimension in loops)
- `linspace`: generate linearly spaced vector (used to create vector of length dimensions)
- `load`: load data from a .m file into workspace (load subdata.m)
- `max`: find the maximum value of an array or matrix
- `meshgrid`: create a 2D or 3D (in our case) grid coordinates of defined x, y, and z vectors.
- `nanmean`: find the average value without including nans
- `print`: export and save figure
- `reshape`: reshape array to set dimensions (64x64x64)
- `scatter3`: scatter plot in 3D
- `set`: manipulate a figure
- `size`: grab size of a matrix
- `text`: define text for a figure
- `quiver3`: plot a quiver in 3D
- `zeros`: create a matrix of zeros

MATLAB codes

```
clear all
close all
clc

addpath(genpath('/Users/cmbaker9/Documents/MTOOLS'))

%% STEP 0: Locate and load data

```

```
datapath = '/Users/cmbaker9/Documents/UW_Classes/AMATH_Data_Analysis/HW1/subdata/';
load([datapath,'subdata.mat']) % Imports the data as the 262144x49 (space by time) matrix called subdata 5

figfolder ='/Users/cmbaker9/Documents/UW_Classes/AMATH_Data_Analysis/HW1/figures/';

%% STEP 1: Define spatial and fft domain

L = 10; % spatial domain
n = 64; % Fourier modes
N = 3; % number of dimensions
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); % create meshgird of cartesian coordinates
[Kx,Ky,Kz]=meshgrid(ks,ks,ks); % create meshgrid of wavenumber space
ave = zeros(64,64,64); % create matrix of zeros

%% STEP 2: Loop through data to identify peak frequencies

for j=1:length(subdata(1,:))
    Un              = reshape(subdata(:,j),n,n,n); % reshape data at each time step
    S               = fftn(Un); % compute fft
    S4d(:,:,:,j)    = S; % storing spectrum
    Un4d(:,:,:,j)   = Un; % storing reshaped data
    ave             = ave+S; % summing spectra
end

%% STEP 3: Find peak frequencies

Savg    = abs(fftshift(ave))/size(S4d,4); % compute time-averaged spectra
Ms      = max(Savg,[],'all'); % find max variance

[val,id] = max(Savg(:)); % find max of spectra index in array
[Sr,Sc,Sp] = ind2sub(size(Savg),id); % find index of max in matrix
maxfreq = [Kx(Sr,Sc,Sp), Ky(Sr,Sc,Sp), Kz(Sr,Sc,Sp)]; % pick max frequencies

% generate isosurface figure of maximum frequency valeus
figure('units','inches','position',[1 1 10 6],'Color','w');
isosurface(Kx,Ky,Kz,Savg/Ms,.5)
hold on
```

```
quiver3(maxfreq(1),maxfreq(2),maxfreq(3)+3,0,0,-2,'AutoScale','off','LineWidth',2,'MaxHeadSize',6,'Color','k')
axis([-10 10 -10 10 -10 10])
grid on
text(maxfreq(1),maxfreq(2)+2,maxfreq(3)+6,'max freq:','interpreter','latex','fontsize',20);
text(maxfreq(1),maxfreq(2)+2,maxfreq(3)+3.5,['(',num2str(round(maxfreq(1),2)),', ',num2str(round(maxfreq(2),2)),', ',num2str(round(maxfreq(3),2)),')'],'interpreter','latex','fontsize',20);
h1=gca;
set(h1,'tickdir','out','xminortick','on','yminortick','on','zminortick','on');
set(h1,'ticklength',2*get(h1,'ticklength'));
set(h1,'fontsize',26);
xlabel('$K_x$ (2$\pi$ /L)','interpreter','latex','fontsize',26);
ylabel('$K_y$ (2$\pi$ /L)','interpreter','latex','fontsize',26);
zlabel('$K_z$ (2$\pi$ /L)','interpreter','latex','fontsize',26);
title('Normalized Time-Averaged 3D Spectra, Values $>$ 0.5','interpreter','latex','fontsize',20);
Sname1 = [figfolder,'Savg_isosurface'];
print(Sname1,'-dpng')

%% STEP 4: Generate filter at peak frequencies

alpha = 0.2; % width of filter
filter = exp(-alpha*((Kx-maxfreq(1)).^2+(Ky-maxfreq(2)).^2+(Kz-maxfreq(3)).^2)); % generate Gaussian filter

% %% STEP 4.5: Test Filter
% Savgtest= Savg.*filter;
% test2d = nanmean(Savg,3);
% test2dfilt = nanmean(Savgtest,3);
% figure
% pcolor(Kx(:,:,1),Ky(:,:,1),test2d); shading interp; colorbar
% figure
% pcolor(Kx(:,:,1),Ky(:,:,1),test2dfilt); shading interp; colorbar

%% STEP 5: Apply filter and inverse fft data

for j=1:length(subdata(1,:))
    Un=reshape(subdata(:,j),n,n,n); % reshape data
    Sfilt = fftshift(fftn(Un)).*filter; % compute fft and multiply by filter
    iUn = ifftn(fftshift(Sfilt)); % comptue inverse fft
    iUn4d(:,:,:,j)=iUn; % store in 4D matrix
    [mfilt, id] = max(abs(iUn(:))); % find maximum value in array
```

```
    [pathx, pathy, pathz] = ind2sub(size(iUn), id); % find index of max value in matrix
    subxyz(:,j) = [X(pathx, pathy, pathz), Y(pathx, pathy, pathz), Z(pathx, pathy, pathz)]; % store path
end

%% STEP 6: Compare unfiltered and filtered data

% compare with raw data
uplot_orig = nanmean(abs(Un4d),4); % comptue average value, original
M_orig = max(uplot_orig,[],'all'); % store maximum
% filtered data
uplot_filt = nanmean(abs(iUn4d),4); % compute average value, filtered
M_filt = max(uplot_filt,[],'all'); % store maximum

% create figure of unfiltered data
figure('units','inches','position',[1 1 10 6],'Color','w');
isosurface(X,Y,Z,uplot_orig/M_orig,0.75)
axis([-10 10 -10 10 -10 10])
grid on
h1=gca;
set(h1,'tickdir','out','xminortick','on','yminortick','on','zminortick','on');
set(h1,'ticklength',2*get(h1,'ticklength'));
set(h1,'fontsize',26);
xlabel('x (L)','interpreter','latex','fontsize',26);
ylabel('y (L)','interpreter','latex','fontsize',26);
zlabel('z (L)','interpreter','latex','fontsize',26);
title('Normalized Original Data, Values $>$ 0.75','interpreter','latex','fontsize',20);
Sname1 = [figfolder,'Un_orig'];
print(Sname1,'-dpng')

% create figure of filtered data
figure('units','inches','position',[1 1 10 6],'Color','w');
isosurface(X,Y,Z,uplot_filt/M_filt,0.75)
axis([-10 10 -10 10 -10 10])
grid on
h1=gca;
set(h1,'tickdir','out','xminortick','on','yminortick','on','zminortick','on');
set(h1,'ticklength',2*get(h1,'ticklength'));
set(h1,'fontsize',26);
xlabel('x (L)','interpreter','latex','fontsize',26);
ylabel('y (L)','interpreter','latex','fontsize',26);
zlabel('z (L)','interpreter','latex','fontsize',26);
```

```matlab
title('Normalized Filtered Data, Values $>$ 0.75','interpreter','latex','fontsize',20);
Sname1 = [figfolder,'Un_filtered'];
print(Sname1,'-dpng')

%% STEP 7: Generate figure showing x,y,z location

timevec = 0:0.5:24; % time array

% figure of the submarine path
figure('units','inches','position',[1 1 10 6],'Color','w');
axes1 = axes('Position',[0.13 0.14 0.68 0.78]);
scatter3(subxyz(1,:),subxyz(2,:),subxyz(3,:),100,timevec,'fill','MarkerEdgeColor','k')
colormap(cmocean('dense'));
cb = colorbar('Position', [0.9 0.1 0.02 0.7],'Location','east');
text(14.6, -7.3, 14.6,'time (hrs)','interpreter','latex','fontsize',20);
axis([-10 10 -10 10 -10 10])
grid on
h1=gca;
set(h1,'tickdir','out','xminortick','on','yminortick','on','zminortick','on');
set(h1,'ticklength',2*get(h1,'ticklength'));
set(h1,'fontsize',26);
xlabel('x (L)','interpreter','latex','fontsize',26);
ylabel('y (L)','interpreter','latex','fontsize',26);
zlabel('z (L)','interpreter','latex','fontsize',26);
title('Submarine Trajectory','interpreter','latex','fontsize',20);
Sname1 = [figfolder,'xyz_trajectory'];
print(Sname1,'-dpng')
```

References

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