GP9505A Problem Set #5 - Filtering and Discrete Fourier Transform

Completed by Benjamin Consolvo

5 November 2015

Submitted to Dr. Gerhard Pratt at Western University

1 Causality, minimum phase, and Hilbert transformation

This report outlines the principles of causality, minimum phase, and Hilbert transforms. The first section is devoted to defining causality and giving an example of a causal time series. The second section defines minimum delay and minimum phase, and provides two illustrations of Argand planes. The third and final section defines a Hilbert transform, gives several graphical examples of Hilbert transform pairs, and describes how Hilbert transforms are related to minimum phase and causal functions.

1.1 Causality

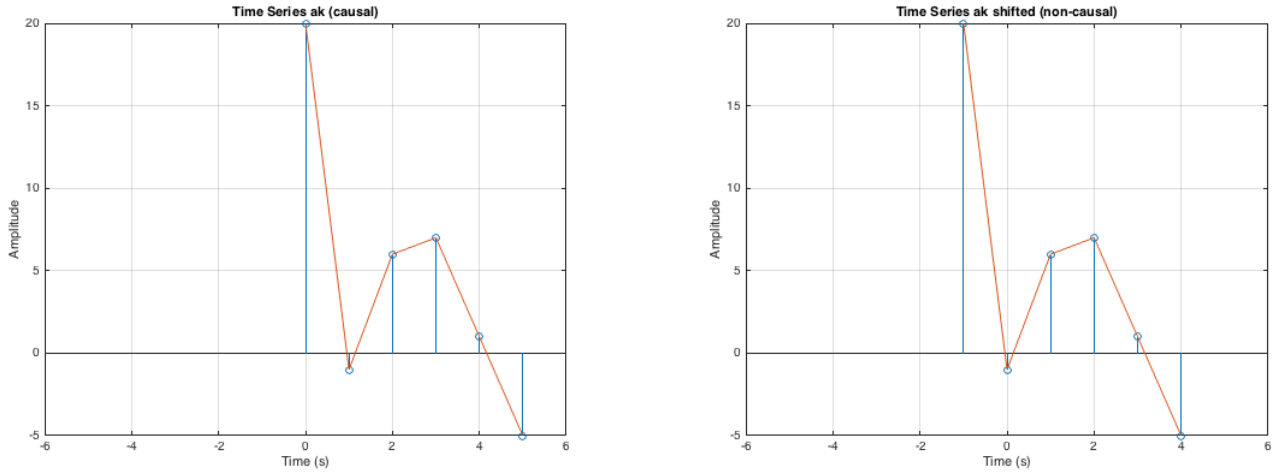
A time series is said to be causal if it has no negative time components. A series can likewise be constructed to be causal by making all negative time components zero (Pratt, 2015). Consider the finite time series (or sequence)

$$a_k \quad = \quad (a_0, a_1, a_2, a_3, a_4, a_5). \tag{1.1}$$

The subscript $k \geq 0$ indicates no negative time components. However, the actual values of the time series can be negative, eg.

$$a_k \quad = \quad (20, -1, 6, 7, 1, -5). \tag{1.2}$$

Figure 1.1 (a) shows a plot of the time series (1.2).



(a) (b)

Figure 1.1: (a) Time series from equation (1.2). Notice that there are no negative time components. (b) Time series from equation (1.4). The time series is shifted to the left by 1 second or by a multiplication of z^{-1}.

A time series can also be represented by the z-transform such that

$$\begin{aligned} A(z) &= a_0 + a_1 z + a_2 z^2 + a_3 z^3 + a_4 z^4 + a_5 z^5 \\ &= 20 + (-1)z + 6z^2 + 7z^3 + 1z^4 + (-5)z^5. \end{aligned} \tag{1.3}$$

If we multiply by z^{-1}, the time series shifts to the left (Figure 1.1 (b)) and is no longer causal:

$$\frac{A(z)}{z} = 20z^{-1} - 1 + 6z^1 + 7z^2 + z^3 - 5z^4. \tag{1.4}$$

We could then represent the new (non-causal) time series as

$$a_{k-1} = (a_{-1}, a_0, a_1, a_2, a_3, a_4) \tag{1.5}$$

$$= (20, -1, 6, 7, 1, -5). \tag{1.6}$$

The associated MATLAB code for Figure 1.1 is given below.

```
%% 1.1
% Define and plot time series
ak=[20,-1,6,7,1,-5]; % Define time series values
t=[0:1:5]; % Define time variable
figure;
stem(t,ak); % vertical line plot
grid on;
hold on;
```

```
plot(t,ak); % continuous line plot
title('Time Series ak (causal)');
xlabel('Time (s)');
ylabel('Amplitude');
xlim([-6 6]);

% shift the time series to the left
ak=[20,-1,6,7,1,-5]; % Define time series values
t=[-1:1:4]; % Define time variable
figure;
stem(t,ak); % vertical line plot
grid on;
hold on;
plot(t,ak); % continuous line plot
title('Time Series ak shifted (non-causal)');
xlabel('Time (s)');
ylabel('Amplitude');
xlim([-6 6]);
```

timeseriesplot.m

We have now come across the term 'non-causal.' A system is said to be non-causal if any nonzero negative time components exist. In the case of the z-transform, as soon as a nonzero z^{-1} term is introduced, the system becomes non-causal. There is an intuitive way to understand a non-causal system. A filter is non-causal if it produces some outputs before it receives any inputs. Finally, a filter is described as anti-causal if *all* of its non-zero time components are negative (all z components have negative powers). A causal filter also is realizable, that is, its inverse exists (Pratt, 2015).

1.2 Minimum delay and minimum phase

Minimum delay filters are a subgroup, or a more specific case, of causal filters. It can be stated that every minimum delay filter is necessarily causal. A causal filter is distinguished as minimum delay if most of its energy is front-loaded. To demonstrate this, if we consider a simple two-term time series,

$$C(z) \quad = \quad a + bz, \tag{1.7}$$

$C(z)$ is said to be minimum delay if and only if $|a| > |b|$. $C(z)$ is said to be maximum delay when $|b| > |a|$ (Pratt, 2015).

Another way of defining minimum delay is to consider the roots of the z-transform polynomial. A system is minimum delay if and only if all of the roots (or zeros) are outside the unit circle in the Argand plane. Let us take a look at the roots of our time series from equation (1.3) to determine if it is minimum phase. Let us transform it into z-transform polynomial:

$$A(z) \quad = \quad 20 - z + 6z^2 + 7z^3 + z^4 - 5z^5 \tag{1.8}$$

The five roots are

$$\begin{aligned} z &= 1.76, \\ z &= -1.162 - 0.787i, \\ z &= -1.162 + 0.787i, \\ z &= 0.382 - 1.004i, \text{ and} \\ z &= 0.382 + 1.004i, \text{ by Wolfram factoring.} \end{aligned}$$

The roots (orange stars) do indeed all lie outside of the unit circle (blue), as shown in Figure 1.2.

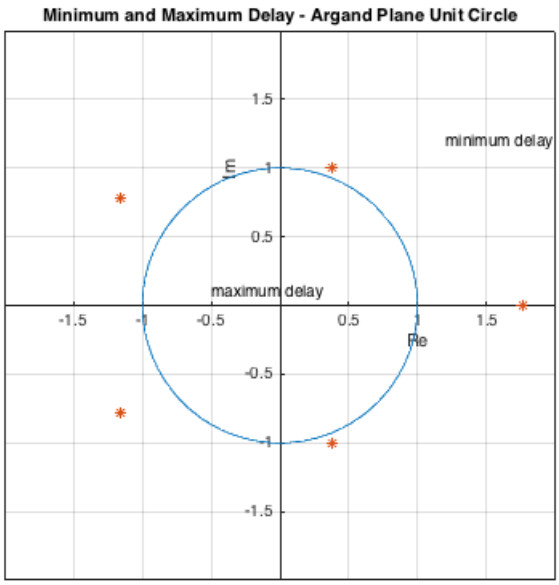


Figure 1.2: Argand plane with unit circle: a system is minimum delay if and only if all zeros lie outside the circle. The x-axis is the real component, and the y-axis is the imaginary component.

The associated code for Figure 1.2 follows.

```
%%1.2 drawing the unit circle in the Argand plane
angle=linspace(0,2*pi,360); % Linearly spaced vector generates 360 points
    between 0 and 2 pi
x=cos(angle); % define x
y=sin(angle); % define y

% defining the roots
zx=[1.7611  -1.16212  -1.16212  0.381574  0.381574] % Real component of root
```

```
zy=[0 -0.787121 0.787121 -1.00364 1.00364] % Imaginary component of root


figure;
plot(x,y); % plot the circle
axis('equal'); % making the axes equal size so that the circle really looks
    like a circle
grid on;
set(gca,'XAxisLocation','origin','YAxisLocation','origin'); %moving axes to
    the centre
text(-0.5,0.1,'maximum delay'); % displaying where maximum delay occurs
text(1.2,1.2,'minimum delay'); % displaying where minimum delay occurs
xlabel('Re');
title('Minimum and Maximum Delay - Argand Plane Unit Circle');
ylabel('Im');
xlim([-2 2]); % setting limits for x-axis
ylim([-2 2]); % setting limits for y-axis
hold on;
plot(zx,zy,'*');
```

argandunitcircle.m

We can then state that $A(z)$ is both causal, and minimum delay. A filter is said to be maximum delay if any of the roots lie inside the unit circle. For example, if a wavelet has a time series

$$f = (-256,\ 1536, -3840, 5376, -4704,\ 2688, -1008,\ 240, -33,\ 2), \tag{1.9}$$

then

$$\begin{aligned} F(z) &= -256 + 1536z - 3840z^2 + 5376z^3 - 4704z^4 + 2688z^5 - 1008z^6 \\ &\quad +240z^7 - 33z^8 + 2z^9 \end{aligned} \tag{1.10}$$

and

$$F(z) = (2z-1)(z-2)^8, \text{ by Wolfram factoring.} \tag{1.11}$$

$$\text{Then, the roots are } z = \frac{1}{2}, 2, 2, 2, 2, 2, 2, 2, \text{ and } 2.$$

While all the roots of $F(z)$ are real, not all are minimum delay. The roots of $z = 2$ are minimum delay, but the root of $z = \frac{1}{2}$ is maximum-delay. Because one root is maximum delay, $F(z)$ is maximum delay. However, if we wanted to make $F(z)$ minimum delay, we

must simply time-reverse the maximum delay component to be minimum delay:

$$(2z-1) \quad \text{becomes} \quad (2-z). \tag{1.12}$$

Then, we can define a new function with all minimum-delay components (zeros at $z=2$) such that

$$G(z) \quad = \quad (2-z)(z-2)^8. \tag{1.13}$$

Then,

$$\begin{aligned} G(z) \quad = \quad & 512 - 2304z + 4608z^2 - 5376z^3 + 4032z^4 - 2016z^5 \\ & +672z^6 - 144z^7 + 18z^8 - z^9. \end{aligned} \tag{1.14}$$

The roots are now

$$z \quad = \quad 2, 2, 2, 2, 2, 2, 2, 2, \text{ and } 2.$$

$G(z)$ is now minimum phase, with all roots outside of the unit circle. Interestingly, the autocorrelation of $F(z)$ is equal to the autocorrelation of $G(z)$,

$$F(z)F\left(\frac{1}{z}\right) \quad = \quad G(z)G\left(\frac{1}{z}\right). \tag{1.15}$$

Indeed, if we construct any minimum delay filter from its maximum delay 'counterpart' using this method, their autocorrelations are equal.

Describing a filter as minimum delay is equivalent to describing it as minimum phase; however, typically when we describe a filter as minimum phase, we are referring to the same filter but in the frequency domain. In order to go into the frequency domain, let us consider again equation (1.7) but replace z with a complex exponential, such that

$$C(\omega) \quad = \quad a + be^{i2\pi n/N}, \tag{1.16}$$

where ω represents frequency, n is the index $n = [0, 1, 2, 3, 4, ...]$ and N is the length of the filter. The phase of $C(\omega)$ is represented by $\phi_n = 2\pi n/N$. If we represent the phase angle (ϕ_n) on the Argand plane, when $|a| > |b|$, we know that ϕ_n exists only in $\left[\frac{-\pi}{2}, \frac{\pi}{2}\right]$. Figure 1.3 depicts the phase as it oscillates in frequency only between $\frac{-\pi}{2}$ and $\frac{\pi}{2}$.

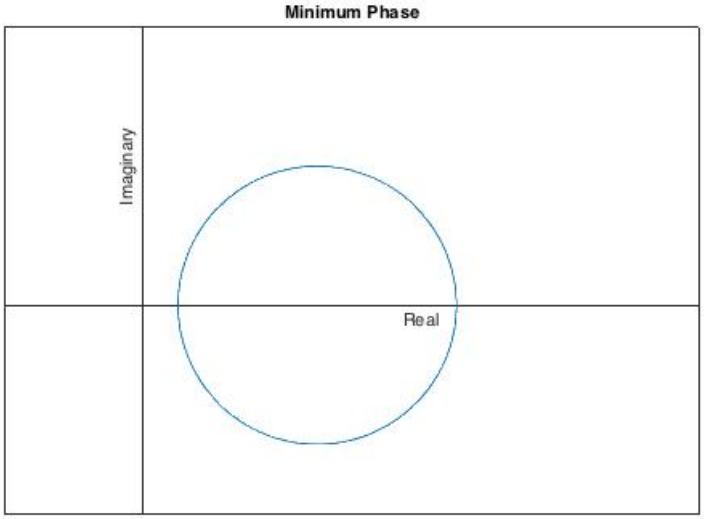


Figure 1.3: Minimum phase diagram in the Argand plane: ϕ_n is the phase angle, and ω is the frequency.

The associated MATLAB code for Figure 1.3 follows.

```
%%1.2 drawing a circle for the minimum phase diagram
clc; clear all; close all; % clearing command line, Clearing variables,
    closing figure windows
angle=linspace(0,2*pi,360); % Linearly spaced vector generates 360 points
    between 0 and 2 pi
x=2*cos(angle)+2.5; % define x
y=2*sin(angle); % define y


figure;
plot(x,y); % plot the circle
axis('equal'); % making the axes equal size so that the circle really looks
     like a circle
xlabel('Real');
ylabel('Imaginary');
set(gca,'XAxisLocation','origin','YAxisLocation','origin'); %moving axes to
     the centre
set(gca,'YTick',[]); % Turning numbers on grid off
set(gca,'XTick',[]); % Turning numbers on grid off
title('Minimum Phase');
xlim([-2 8]); % setting limits for x-axis
ylim([-3 4]); % setting limits for y-axis
```

minimumphasecircle.m

1.3 Hilbert transform

Now that we understand a bit more about causality and minimum phase, we can illustrate their importance in relation to Hilbert transforms. We must first define a Hilbert transform. The objective of performing a Hilbert transform is to perform a 90° phase-shift on a signal. In order to do this, we can multiply by the Quadrature function, defined by

$$Q(\omega) = -i\text{sgn}(\omega) \tag{1.17}$$

$$Q(\omega) = \frac{-i\omega}{|\omega|}, \tag{1.18}$$

where ω is frequency (Pratt, 2015). This operator is purely imaginary and only modifies the phase of the signal. We can then define the Hilbert transform as

$$HT\{F(\omega)\} = Q(\omega)F(\omega). \tag{1.19}$$

In the time domain, we can define the Hilbert transform as a convolution such that

$$HT\{f(t)\} = FT^{-1}\{Q(\omega)\} * f(t). \tag{1.20}$$

We then know from Pratt (2015) that

$$HT\{f(t)\} = \frac{-1}{\pi t} * f(t) \tag{1.21}$$

and

$$HT\{f(t)\} = \frac{1}{\pi}\int_{-\infty}^{\infty}\frac{f(\tau)}{\tau - t}. \tag{1.22}$$

The analytic signal is an important concept that relates the real part of a filter to its imaginary Hilbert transform pair. The analytic signal is defined as

$$a(t) = f(t) + iHT\{f(t)\}, \tag{1.23}$$

where the original time function is $f(t)$, and its Hilbert transform is $HT\{f(t)\}$. We could

then say that

$$a(t) \quad = \quad f(t) + i\frac{1}{\pi}\int_{-\infty}^{\infty}\frac{f(\tau)}{\tau - t}. \tag{1.24}$$

We still have a real part, the original function $f(t)$); and an imaginary part, the Hilbert transform.

To give an example, let us plot a sine function and its Hilbert transform pair. Figure 1.4 shows a sine wave, and multiplication by the quadrature function. Notice that the phase is shifted by $\frac{\pi}{2}$ or 90°. The associated MATLAB code follows Figure 1.5.

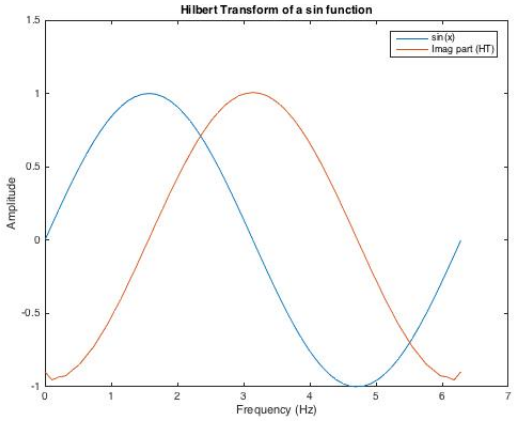
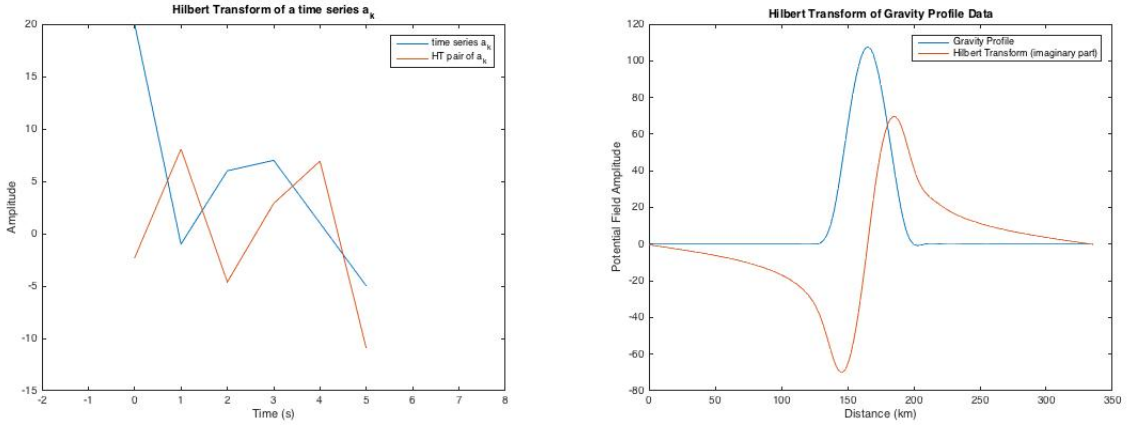


Figure 1.4: Blue line is the original sine function. Red line is the imaginary part of the analytic signal (Hilbert transform of the original sine function).

To further illustrate the relationship between a filter and its Hilbert transform, Figure 1.5 (a) shows the constructed time series, a_k (equation (1.2)), and its Hilbert transform pair. Figure 1.5 (b) shows a profile of gravity data and its Hilbert transform pair.



(a) (b)

Figure 1.5: (a) Original real time series a_k in blue and its Hilbert transform in red; (b) original gravity data profile in blue and its Hilbert transform in red.

MATLAB actually first calculates the whole analytic signal. I plot the original function and the imaginary part of the analytic signal. The associated MATLAB code follows.

```
%%1.2 drawing the unit circle in the Argand plane
angle=linspace(0,2*pi,360); % Linearly spaced vector generates 360 points
    between 0 and 2 pi
x=cos(angle); % define x
y=sin(angle); % define y

% defining the roots
zx=[1.7611 -1.16212 -1.16212 0.381574 0.381574] % Real component of root
zy=[0 -0.787121 0.787121 -1.00364 1.00364] % Imaginary component of root


figure;
plot(x,y); % plot the circle
axis('equal'); % making the axes equal size so that the circle really looks
     like a circle
grid on;
set(gca,'XAxisLocation','origin','YAxisLocation','origin'); %moving axes to
     the centre
text(-0.5,0.1,'maximum delay'); % displaying where maximum delay occurs
text(1.2,1.2,'minimum delay'); % displaying where minimum delay occurs
xlabel('Re');
title('Minimum and Maximum Delay - Argand Plane Unit Circle');
ylabel('Im');
xlim([-2 2]); % setting limits for x-axis
ylim([-2 2]); % setting limits for y-axis
hold on;
plot(zx,zy,'*');
```

argandunitcircle.m

Pratt (2015) states two important facts concerning causality, minimum phase, and Hilbert transforms:

1. The Fourier transform of a causal filter has real and imaginary components that are Hilbert transform pairs.

2. The Fourier transform of a minimum phase filter satisfies the causality condition and the log amplitude spectra and phase spectra are Hilbert transform pairs.

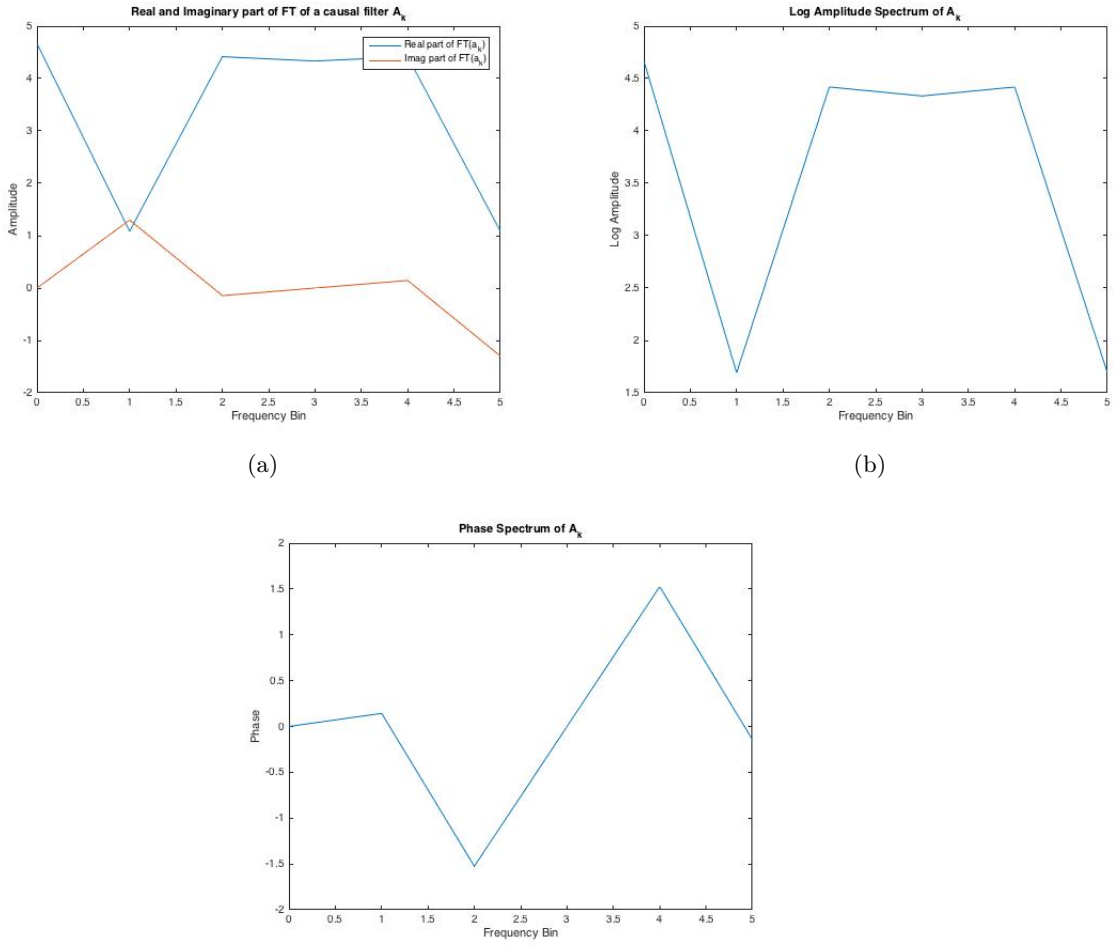
In particular for (2) stated above, the phase can be computed as the Hilbert transform of the log of the amplitude spectrum (Margrave, 2014). An understanding of Fourier transforms is assumed at this point. From Figure 1.2, we know that our constructed time series a_k (eq. (1.2)) is both causal and minimum phase. Let us first then take the discrete Fourier transform of a_k and plots its real and imaginary parts. The discrete Fourier transform of a_k can be represented by

$$A_n = \frac{1}{N}\sum_{k=0}^{N-1} a_k e^{i2\pi nk/N}, \tag{1.25}$$

where N is the length of A_n (which is 6), and $n = [0, 1, 2, 3, 4, 5]$. The Fourier transform is then calculated to be

$$A_n = [4.67, 1.08 + 1.30i, 4.42 - 0.14i, 4.33, 4.42 + 0.14i, 1.08 - 1.30i]. \tag{1.26}$$

MATLAB calculated the values for A_n displayed here. The code can be found below Figure 1.6. Figure 1.6 (a) shows a plot of the real and imaginary values of A_n. Figure 1.6 (b) is a plot of the log of the amplitude spectrum of A_n. And finally, (c) depicts the Hilbert transform of the log of the amplitude spectrum, which is also the phase.



(a) (b)

(c)

Figure 1.6: (a) Real part of A_k in blue, and imaginary part of A_k in red; (b) plot of the log of the amplitude spectrum; (c) plot of the Hilbert transform of the log of the amplitude spectrum (the phase).

The associated MATLAB code for Figure 1.6 follows.

```
%% Because ak is causal, its Fourier Transform should have real and imag
    parts that are Hilbert Transform pairs.
close all; clc; clear all;
ak=[20,-1,6,7,1,-5]; % Define time series values
t=[0:1:5]; % Define time variable
FTak=ifft(ak); % Taking the Fourier Transform of the time series ak. The
    IFFT "inverse fast fourier transform" is taken in our case because the
    convention in MATLAB is to put the normalizing factor in front of this
    function.
figure;
plot(t,real(FTak)); % Plotting the real part of the Fourier transform
```

```
hold on;
plot(t,imag(FTak)); % Plotting the imaginary part of the FT
legend('Real part of FT(a_k)','Imag part of FT(a_k)');
title('Real and Imaginary part of FT of a causal filter A_k');
xlabel('Frequency Bin');
ylabel('Amplitude');
%% Calculating the log amplitude spectrum of Ak and then the Hilbert
    Transform of it to get the phase
a1=abs(FTak); % log amplitude spectrum
h1=hilbert(a1); % calculating the analytic function from the log amplitude
    spectrum
p1=imag(h1); % The associated phase of Ak by taking the imaginary part of
    the analytic signal
figure;
plot(t,a1); % plotting the log of the amplitude spectrum
title('Log Amplitude Spectrum of A_k');
xlabel('Frequency Bin');
ylabel('Log Amplitude');

figure;
plot(t,p1); % plotting the Hilbert transform of the log of the amplitude
    spectrum (the phase)
title('Phase Spectrum of A_k by HT calculation');
xlabel('Frequency Bin')
ylabel('Phase');
```

FTplots.m

1.4 References

Pratt, G (2015). Time Series Analysis and Inverse Theory. Course Notes.

Margrave, G (2014). Methods of Seismic Data Processing. Course Lecture Notes.

2 Low-pass filter

Solutions for (a) and (b) on attached graph paper.

3 Low-pass Butterworth filter

a) Solution on attached graph paper.
b) For the mathematics of finding a formula for the poles of the Butterworth filter, refer to attached graph paper. The power spectrum of the Butterworth filter is

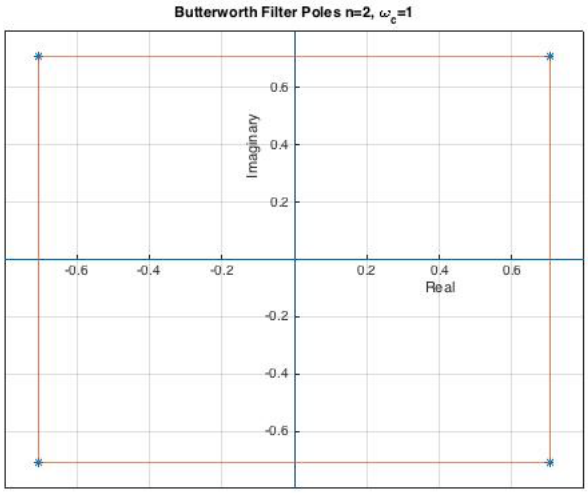
$$|B(i\omega)|^2 \quad = \quad B(i\omega)B(-i\omega),$$

where ω is angular frequency.

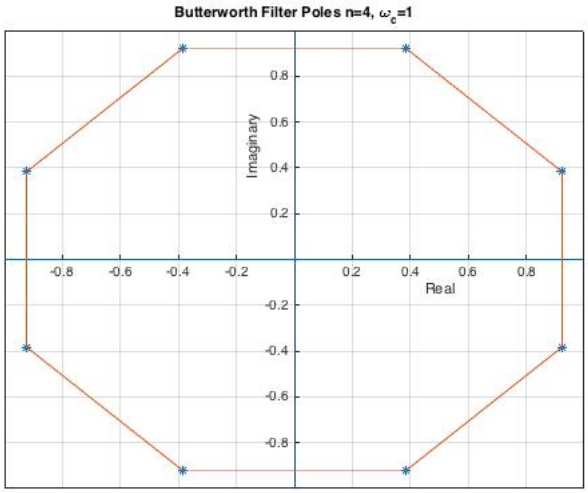
The poles can be represented in the Argand plane, using the formula:

$$u \quad = \quad \left[exp\left(\frac{i\pi(2m+1+n)}{2n}\right)\right]\omega_c,$$

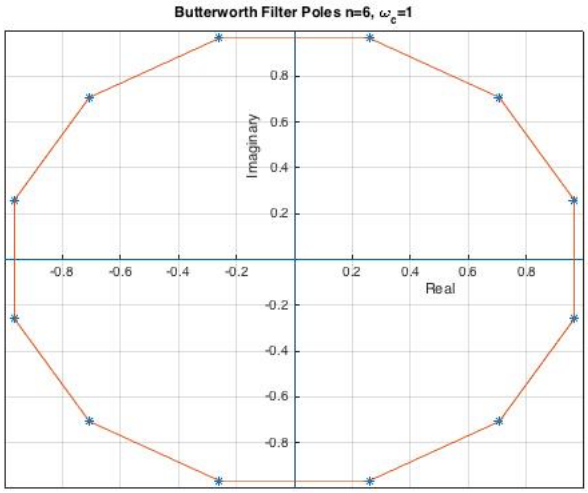
where u is a dummy variable for $i\omega$, $m = [1, 2, 3...]$, n is the number of poles, and ω_c is the cut-off frequency. The plots in Figure 3.1 show the location of the poles of the Butterworth filter, for $n = 2$, $n = 4$, and $n = 6$, keeping the cutoff frequency (ω_c) constant.



(a)



(b)



(c)

Figure 3.1: Butterworth filters in the ω plane for (a) 2 poles, (b) 4 poles, and (c) 6 poles.

The code used in MATLAB to generate the plots in Figure 3.1 follows.

```
%% 3. b)
% Plotting poles for n = 2,4,6
clear all;
n=2;
m=[-n:1:n];
omegac=1;
for k=1:length(m);
    u(k) = omegac*exp((i*pi*(2*m(k)+1+n)/(2*n)));
end


figure;
plot(real(u),imag(u),'*');
set(gca,'XAxisLocation','origin','YAxisLocation','origin');
title('Butterworth Filter Poles n=2, \omega_c=1');
xlabel('Real');
ylabel('Imaginary');
xL = xlim;
yL = ylim;
line([0 0], yL);  %x-axis
line(xL, [0 0]);  %y-axis
grid on;
hold on;
plot(real(u),imag(u));

clear all;
n=4;
m=[-n:1:n];
omegac=1;
for k=1:length(m);
    u(k) = omegac*exp((i*pi*(2*m(k)+1+n)/(2*n)));
end


figure;
plot(real(u),imag(u),'*');
set(gca,'XAxisLocation','origin','YAxisLocation','origin');
title('Butterworth Filter Poles n=4, \omega_c=1');
xlabel('Real');
ylabel('Imaginary');
xL = xlim;
yL = ylim;
line([0 0], yL);  %x-axis
```

```
line(xL, [0 0]); %y-axis
grid on;
hold on;
plot(real(u),imag(u));


clear all;
n=6;
m=[-n:1:n];
omegac=1;
for k=1:length(m);
    u(k) = omegac*exp((i*pi*(2*m(k)+1+n)/(2*n)));
end


figure;
plot(real(u),imag(u),'*');
set(gca,'XAxisLocation','origin','YAxisLocation','origin');
title('Butterworth Filter Poles n=6, \omega_c=1');
xlabel('Real');
ylabel('Imaginary');
xL = xlim;
yL = ylim;
line([0 0], yL); %x-axis
line(xL, [0 0]); %y-axis
grid on;
hold on;
plot(real(u),imag(u));
```

project53b.m