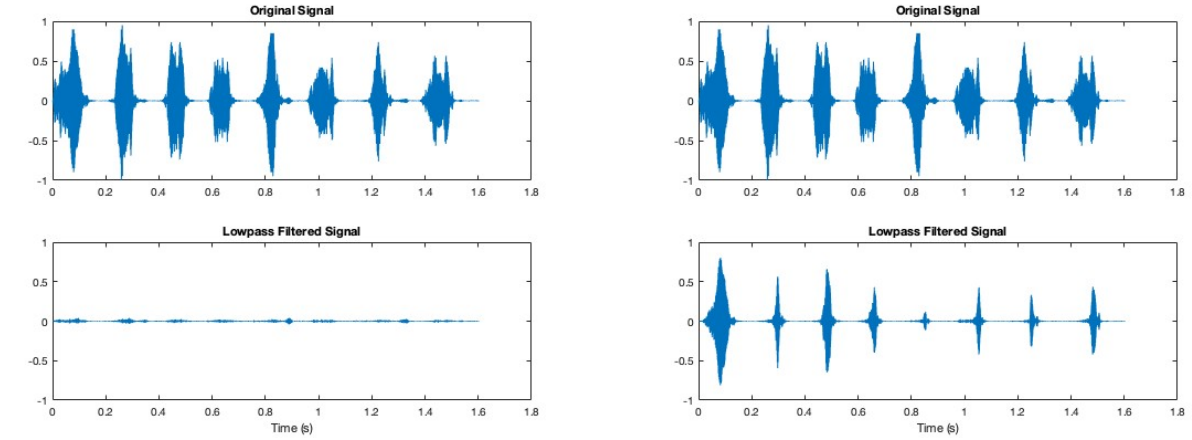
1 Laboratory 2: The $\mathcal{Z}$-transform and filter design

First consider some visualizations on FIR filters using the function `fir1` which directly sets the window to the Hamming filter. Let us define two different low pass filters following the commands:

1. `fir1(48, [0.35 0.65])`: low pass filter with 48^{o} and $0.35\pi \leq \omega \leq 0.65\pi$ rad with the Hamming window
2. `fir1(34, 0.48, chebwin(35, 30))`: low pass filter with 34^{o} and split frequency of 0.48 rad applying a Chebychev window of 30db curvature

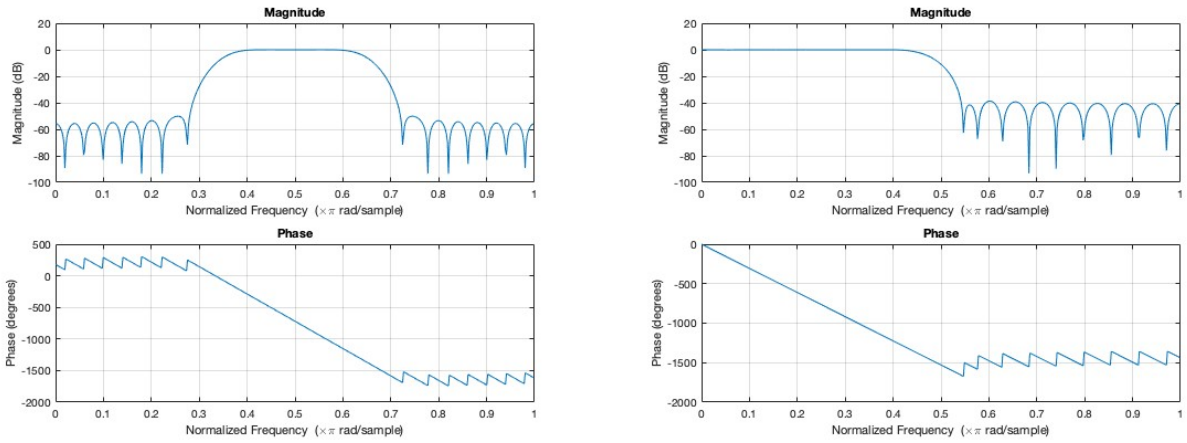
Then, by applying these two filters to a given input signal we get the output shown on the Figures 1.



(a) Applying low pass filter 1 to a sample signal. (b) Applying low pass filter 1 to a sample signal.

Figure 1: Applying low pass filter `fir1` to a sample signal.

The filters defined are represented on Figures 2



(a) Filter 1 (b) Filter 2.

Figure 2: Filter representation.

Similar code can be used for Fast Fourier Transform filters using the function `fir2`.

1.1 Exercise 1

We seek to design a simple second order FIR filter (Finite Impulse Response Filters) with the two zeros on the unit circle. In order for the filter's impulse response to be real valued, the two zeros must be complex conjugates of one another $z_{\pm} = e^{\pm i\theta}$. The transfer function for this filter is given by

$$H(z) = 1 - 2\cos\theta z^{-1} + z^{-2}.$$

Let's use this transfer function to determine the recursion formula (or difference equation) for this filter and compute the filter's impulse response $h[n]$.

In order to get the recursion formula, consider the transfer function in terms of the rational function as follows

$$H(z) = \frac{P(z)}{Q(z)} = \frac{\sum^M b_k z^{-k}}{\sum^N a_k z^{-k}}$$

Observe therefore that it is direct that $Q(z) = 1$, since the numerator is already a polynomial on z^{-1}. Therefore, we have

$$Y(z) = X(z)H(z),$$

where we have denoted $X(z) = \mathcal{Z}\{x[n]\} = \sum_{k\in\mathbb{Z}} x[k]z^{-k}$. Therefore, we have

$$Y(z) = X(z)(1 - 2\cos\theta z^{-1} + z^{-2}).$$

Recall now that multiplying by z^{-1} corresponds to a shift of the sequence, therefore the $\mathcal{Z}$ transform corresponding expression is given as follows:

$$y[n] = x[n] - 2\cos\theta x[n-1] + x[n-2].$$

Observe therefore that on the recursive form the output does not depend on the past outputs.

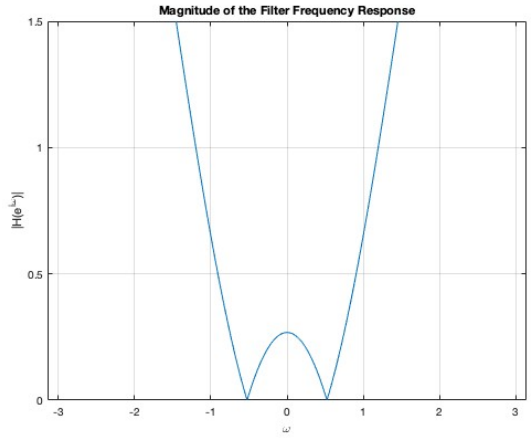
Therefore, since $H(z) = \mathcal{Z}\{h[n]\} = \sum_{k\in\mathbb{Z}} h[k]z^{-k}$, we have that the impulse response $h[n]$ is given by:

$$h[0] = 1, \quad h[1] = -2\cos\theta, \quad h[2] = 1$$

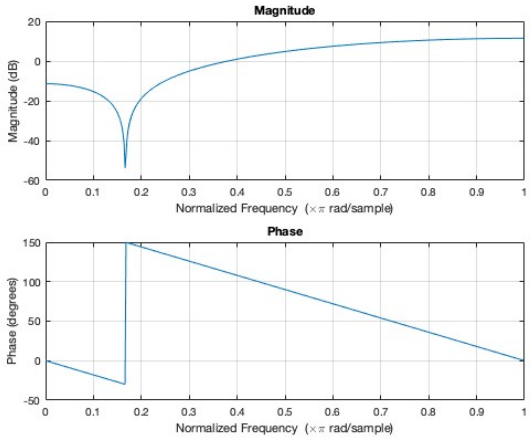
and $h[n] = 0$ for any other n value.

Observe that the impulse response $h[n]$ has finite duration, and therefore we are considering a FIR filter. Any filter with only zeros and no poles other than those at 0 and ∞ is a FIR filter.[1]
Zeros in the transfer function represent frequencies that are not passed through the filter. This can be useful for removing unwanted frequencies in a signal. The fact that $H(z)$ has zeros at $e^{i\theta}$ means that the filter will not pass pure sine waves at frequency θ. We seek to use Matlab to compute and plot the magnitude of the filter's frequency response $|H(e^{i\omega})|$ as a function of ω on the interval $-\pi < \omega < \pi$, when $\theta = \pi/6$. See Figure 3a.

[1]Recall that when we have a signal $s(t)$ with $\mathcal{Z}$-transform $\mathcal{Z}[s](z) = \frac{N(z)}{D(z)}$, then the roots of $D(z)$ are called the poles of the signal, and the roots of $N(z)$ are called the zeros of the signal.



(a) $|H(e^{i\omega})|$ as a function of ω on the interval $-\pi < \omega < \pi$, when $\theta = \pi/6$.



(b) Impulse response function representation.

Figure 3: Representation on the plane t, x, of the numerical continuation of periodic orbits for different ω ranges.

In the next experiment, we will use the filter $H(z)$ to remove an undesirable sinusoidal interference from a speech signal. To run the experiment, first read the file `easy.wav` that is provided with the instruction:

```
[x,FS] = audioread('easy.wav');
```

The vector x represents the audio signal and FS is an integer that denotes the frequency at which it has been sampled. We can play the sound with the instruction

```
player = audioplayer (0.8*x, FS);
play (player);
```

Now, we add some sinusoidal noise to the signal

```
t = (1:length(x))/FS;
y = 0.8*x + 0.1*sin(35000* t')
```

and we hear the noisy signal again with

```
player = audioplayer (y, FS);
play (player);
```

We seek to design a filter H with two zeros, that attenuates the noise, filter the signal and hear it again. In order to choose the value of θ we want to find the recursive equation of the filter, which observe that is given as follows:

$$y = T(x) = 0.8x + 0.1 * sin(35000 * t')$$

Therefore, we have $Y(z) = X(z)H(z)$, where Y is the noisy signal, due to the sinusoidal noise added. Observe that the added noise has frequency of $35000Hz$ which is really high frequency. Therefore, one way to filter this noise could be considering a `lowpass` filter in order to let just the lower frequencies get through the filter, and removing the high frequency noise added to the original signal, for example:

```
y_filtered2 = lowpass(y, 350, FS, ImpulseResponse= "fir");
y_filtered3 = lowpass(y, 15000, FS, ImpulseResponse= "fir");
```

Both low pass filters remove the sinusoidal noise, since in both cases the specified passband frequency filters out the added high frequency noise.

Notice that this filters should be relatively equivalent to the aforementioned filter $H(z)$ with zeros at $e^{i\theta}$. In the case of this alternative filter, recall that given that it filters out pure sine waves at frequency θ. Hence, in this case, the sinusoidal noise has a frequency of 35 kHz, which is a high frequency component. Therefore, we want to design a filter with two zeros on the unit circle that are located at frequencies close to 35 kHz, in order to attenuate this frequency component.
One way to do this is to choose θ such that the two zeros are located at frequencies that are symmetrically located around 35 kHz on the unit circle. Specifically, we want to choose θ such that the two zeros are located at $\omega = \pm\Delta\omega$, where $\Delta\omega$ is the frequency offset from 35 kHz.

We can compute the value of $\Delta\omega$ using the formula: $\Delta\omega = \frac{f}{FS} \approx 0.79$, where f is the frequency we want to remove (35 kHz in this case), and FS is the sampling frequency of the signal (given as the variable `FS` in the code).

Therefore, in this case, we could use a filter as described above:

$$h[0] = 1, \quad h[1] = -2\cos\theta, \quad h[2] = 1$$

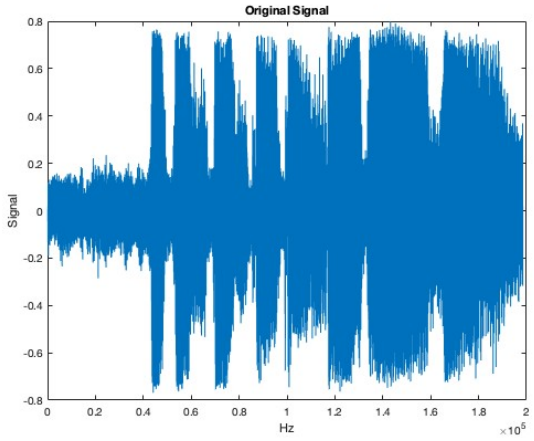
with $\theta = \Delta\omega$ the noise with the given frequency is removed. The corresponding code could be used by implementing one of these, either using the `filter` function from the `MATLAB` libraries:

```
theta_ = (35000) / FS;
h_filter = [1, -2*cos(theta_), 1];
y_filtered_H = filter(h_filter, 1, y);
```

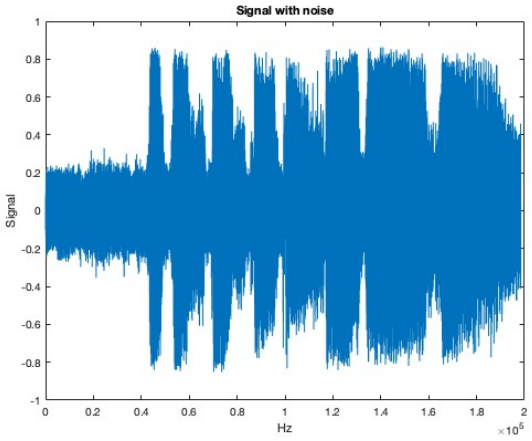
or either defining a custom function with the fitler and then apply it by using the convolution function:

```
function h_ = filter_h(theta_)
    h_ = [1, -2*cos(theta_), 1];
end
y_filtered_H2 = conv(y, filter_h(theta_));
```

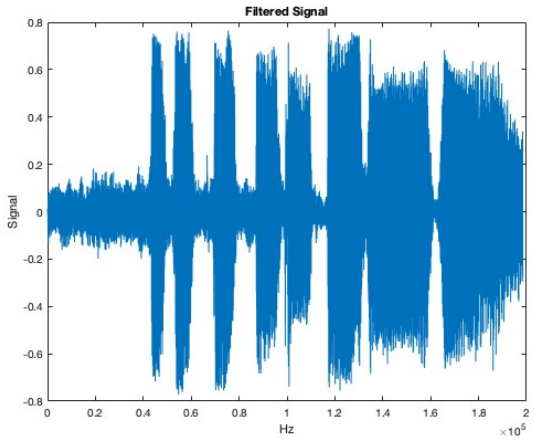
By running the corresponding code, `Lab2_1.m` the filtered signal using the studied filter which removes the specific frequency noise is played. The corresponding representations of the noises are presented on Figures 4



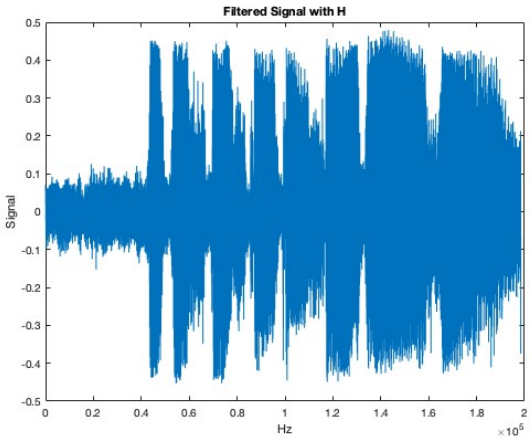
(a) Original signal.



(b) Noisy signal



(c) Filtered signal using the lowpass filter.



(d) Filtered signal using H.

Figure 4: Representations of the signal.