REPORT

Zajęcia: Analog and digital electronic circuits

Teacher: prof. dr hab. Vasyl Martsenyuk

Lab 8

Date: 12.04.2024

Topic: "FIR filtering"

Variant: 13

Agnieszka Białecka

Informatyka II stopień,

stacjonarne,

1 semestr,

Gr.1a

1. Abstract

The objective was to investigate FIR filtering technique for different parameters of filtering.

2. Theoretical introduction

2.1. Filter Fundamentals

The transfer function of digital filters can be generally expressed in the z-domain as

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{m=0}^{M} b_m z^{-m}}{\sum_{n=0}^{N} a_n z^{-n}} = \frac{b_0 z^0 + b_1 z^{-1} + b_2 z^{-2} + \ldots + b_M z^{-M}}{a_0 z^0 + a_1 z^{-1} + a_2 z^{-2} + \ldots + a_N z^{-N}}$$

with input X(z) and output Y (z). Real input signals x[k] that should end up as real output signals y[k] (in terms of signal processing fundamentals this is a special case, though most often needed in practice) require real coefficients b, a ∈ R. This is only achieved with single or multiple real valued - single or multiple complex conju- gate pairs of zeros and poles.

Furthermore, in practice we most often aim at (i) causal and (ii) bounded input, bound output (BIBO) stable LTI systems, which requires (i) M ≤ N and (ii) poles inside the unit circle. If all poles and zeros are inside the unit circle then the system is minimum-phase and thus H(z) is straightforwardly invertible.

Further concepts related to the transfer function are:

- Analysis of the transfer characteristics is done by the DTFT H(z = $e^{j\Omega}$),i.e. evaluation on the unit circle.
- We use a0 = 1 according to convention in many textbooks.
- The convention for arraying filter coefficients is straightforward with Python index starting at zero: b0 = b[0], b1 = b[1], b2 = b[2], ..., a0 = a[0] = 1, a1 = a[1], a2 = a[2].

2.1.1. Filtering Process

A non-recursive system with a1, a2, ..., aN = 0 always exhibits a finite impulse response (FIR), note: a0 = 1 for output though. Due to the finite length impulse response, a non-recursive system is always stable.

The output signal of a non-recursive system in practice can be calculated by linear convolution.

$$y[k] = \sum_{m=0}^{M} h[m]x[-m+k]$$

of the finite impulse response h[m] = [b0, b1, b2, ..., bM] and the input signal x[k].

A recursive system exhibits at least one $a_{n\geq1} \neq 0$. Because of the feedback of the output into the system, a potentially infinite impulse response (IIR) and a potentially non-stable system results.

For a recursive system, in practice the difference equation needs to be implemented:

$$y[k] = b_0x[k] + b_1x[k-1] + b_2x[k-2] + \ldots \\ + b_Mx[k-M] - a_1y[k-1] - a_2y[k-2] - a_3y[k-3] - \ldots - a_Ny[k-N]$$

-A pure non-recursive system is obtained by ignoring the feedback paths, i.e. setting a_1, a_2, ..., a_N = 0.

A pure recursive system is obtained by ignoring the forward paths, i.e. setting b_0, b_1, ..., b_M = 0. Then, the values of the state variables z^{-1}, z^{-2}, ..., z^{-M} alone determine how the system starts to perform at k = 0, since the system has no input actually. This system type can be used to generate (damped) oscillations.

2.1.2. Signal Flow Chart of Direct Form I

For example, the signal flow for a second order (M = N = 2), system with - a non-recursive part (feedforward paths, left z−-path) - a recursive part (feedback paths, left z−-path) is depicted below (graph taken from Wikimedia Commons) as straight- forward direct form I, i.e. directly following the difference equation.

2.2. FIR Filter

If all coefficients $a_{1,\ldots,N}$ = 0, the feedback paths are not existent in the signal flow chart above. This yields a non-recursive system and has transfer function

$$H(z) = \frac{Y(z)}{X(z)} = \sum_{m=0}^{M} b_m z^{-m} = b_0 z^0 + b_1 z^{-1} + b_2 z^{-2} + \ldots + b_M z^{-M}.$$

with the difference equation

$$y[k] = b_0x[k] + b_1x[k-1] + b_2x[k-2] + \ldots + b_Mx[k-M],$$

from which we can directly observe that the impulse response (i.e. for x[k] = δ[k]) is

$$h[k] = b_0\delta[k] + b_1\delta[k-1] + b_2\delta[k-2] + \ldots + b_M\delta[k-M].$$

This constitutes h[k] as the coefficients bk at sample instances k.

The impulse response for this non-recursive system has always finite length of M + 1 samples.

Usually this filter type is referred to as finite impulse response (FIR) filter in literature.

Very special recursive systems/filters can produce FIRs as well. This is however so rare, that the common link FIR filter == non-recursive system is predominantly made.

The filter order is M , the number of coefficients b is M + 1. Be cautious here and consistent with the naming, it sometimes gets confusing. Especially for linear phase filters (see below) it is really important if M or M + 1 is either even or odd.

Sometimes the number of taps M +1 is stated (rather than calling this number of coefficients). This refers to tapping M + 1 delayed instances of the signal input signal x to calculate the filtered output signal y. Note however, that tapping the signal for the first coefficient b_0 involves non-delayed x[k].

For FIR filters the magnitude at Ω = 0 and Ω = π can be straightforwardly evaluated with the following equations.

The DC magnitude is obtained by

$$g_0 = \sum_{k=0}^{M} h[k].$$

The magnitude at Ω = π (i.e. at half the sampling frequency) is obtained by

$$g_\pi = \sum_{k=0}^{M} (-1)^k h[k].$$

2.2.1. Poles / Zeros of FIR Filter

To calculate poles and zeros of the transfer function H(z) it is meaningful to rewrite

$$H(z) = \sum_{m=0}^{M} b_m z^{-m} \frac{z^M}{z^M} = (b_0 z^0 + b_1 z^{-1} + b_2 z^{-2} + \ldots + b_M z^{-M}) \frac{z^M}{z^M}.$$

This reveals that FIR filters have an M -fold pole at $z_\infty = 0$. This is always the case for non-recursive systems and besides the finite impulse response explains why these systems are always stable: poles in the origin are harmless, since they equally contribute to all frequencies.

For the zeros, the equation

$$\sum_{m=0}^{M} b_m z^{-m} z^M = b_M z^M + b_1 z^{M-1} + b_2 z^{M-2} + \ldots + b_M = 0$$

needs to be solved. The M -th order polynomial has M zeros. Recall from above, that for b ∈ R only real or complex conjugate zeros can occur, but never single complex zeros.

2.2.2. Essence of FIR Filter Design

The fundamental concept (just as was it with window design for the DFT) is to place the M available zeros in the z-plane such that a target magnitude and phase response results, which suits the desired filter characteristics.

It is important to note that (contrary to IIR filters) the magnitude and phase response can be separately controlled with FIR filters.

In the referenced [english monographs](../index.ipynb) we can find information on specific FIR design techniques such as:

- FIR design with windowing method
- FIR design as minimax least-squares optimization problem
- FIR design of frequency sampling a DTFT spectrum

These are well covered in Python’s Scipy and Matlab. We will later discuss the windowing method.

2.2.3. A Note on FIR Filtering vs. DFT Windowing

Consider an input signal x[k].

An FIR filter h[k] is used for convolution (i.e. filtering process)

$$x[k] * h[k] \circ\!-\!\bullet\, X(\mathrm{e}^{\mathrm{j}\Omega}) \cdot H(\mathrm{e}^{\mathrm{j}\Omega}),$$

whereas the DFT windowing process involves a multiplication with a window w[k]

$$x[k] \cdot w[k] \circ\!-\!\bullet\, \frac{1}{2\pi} X(\mathrm{e}^{\mathrm{j}\Omega}) \circledast_{2\pi} W(\mathrm{e}^{\mathrm{j}\Omega}).$$

In the DTFT domain this results in multiplication and circular convolution, respectively.

So, for the finite-length sequences h[k] and w[k] the same design fundamentals hold: we must put zeros at suitable locations in the z-plane to realize a certain desired DTFT spectrum, either H(ejΩ) or W (ejΩ). Depending on the application, filtering or windowing, the DTFT design criteria might be very different, since the DTFT spectrum acts as multiplication or convolution onto the input signal's DTFT spectrum.

3. Input data (Variant)

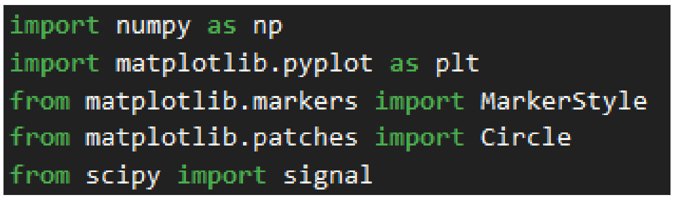
This report was created with base of variant 13:

M	b_0	b_1	b_2
2	0	1	1

GitHub repository:

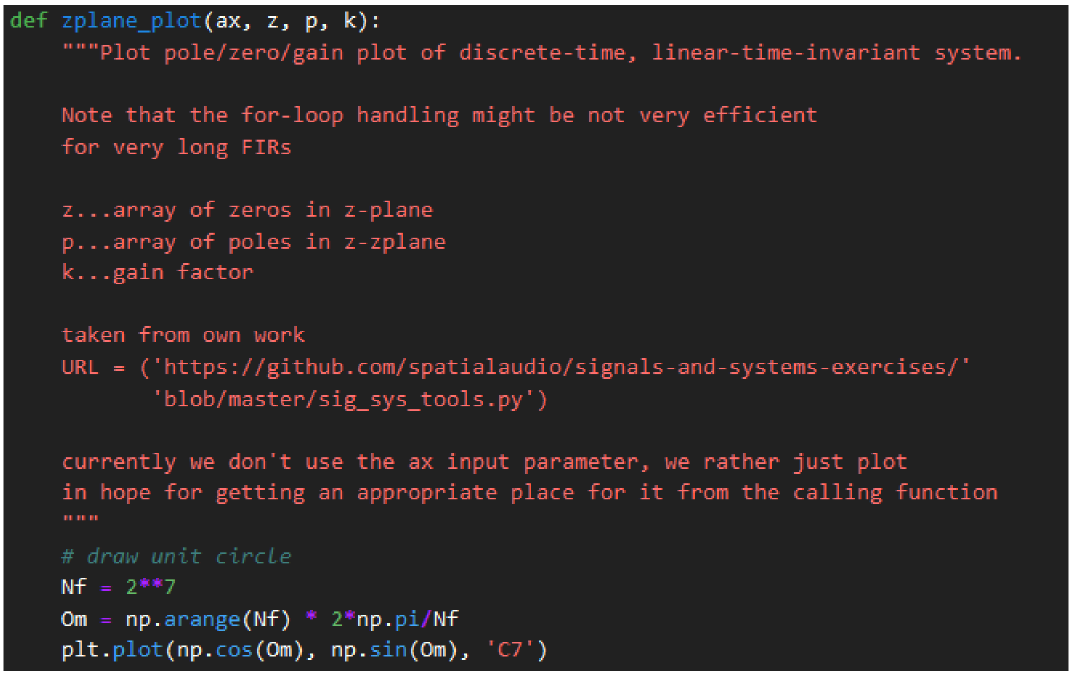
https://github.com/Delisolara/AaDEC

4. Course of actions



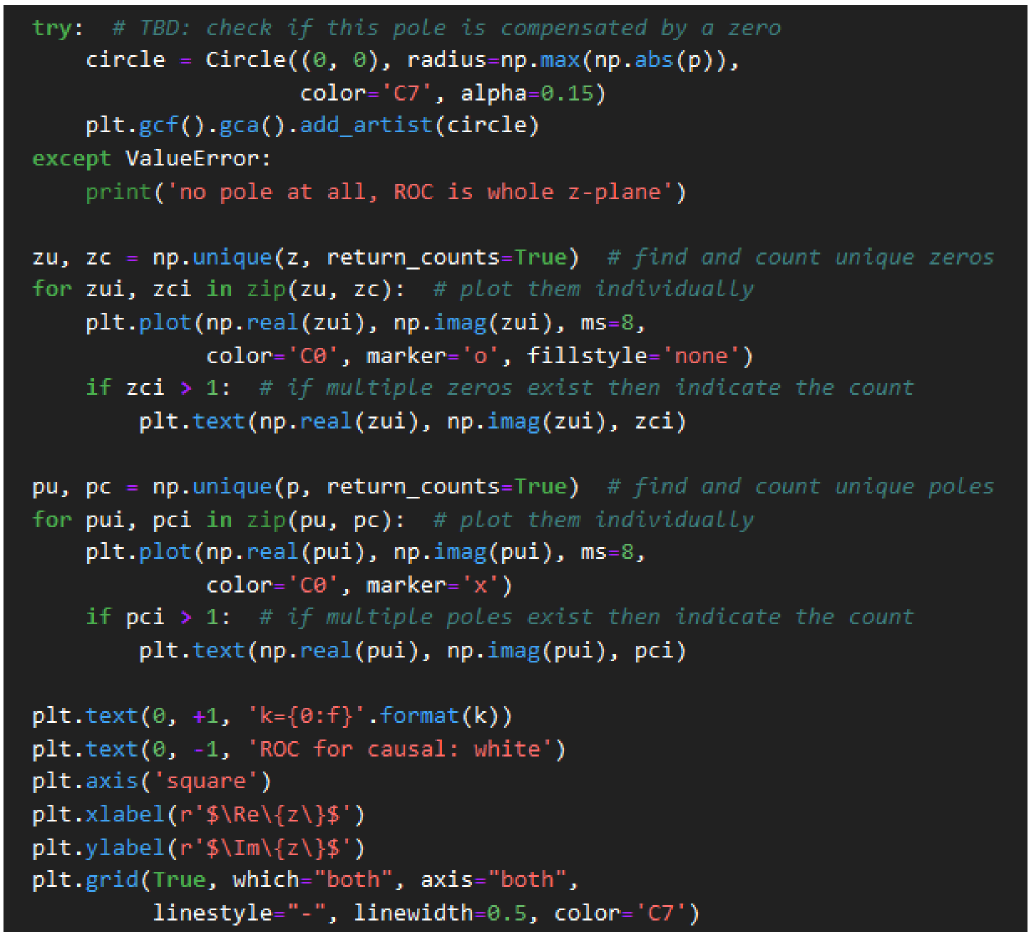
```python
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.markers import MarkerStyle
from matplotlib.patches import Circle
from scipy import signal
```

Picture 1. Uploaded libraries



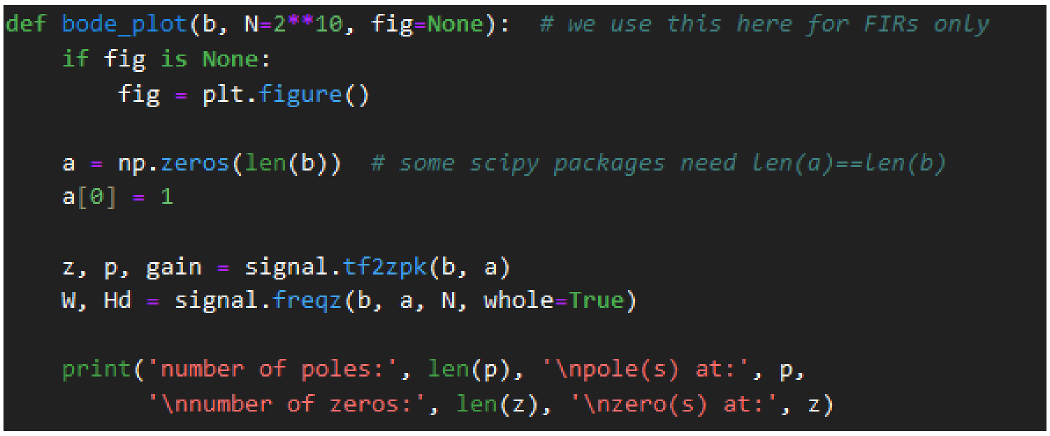
```python
def zplane_plot(ax, z, p, k):
    """Plot pole/zero/gain plot of discrete-time, linear-time-invariant system.

    Note that the for-loop handling might be not very efficient
    for very long FIRs

    z...array of zeros in z-plane
    p...array of poles in z-zplane
    k...gain factor

    taken from own work
    URL = ('https://github.com/spatialaudio/signals-and-systems-exercises/'
           'blob/master/sig_sys_tools.py')

    currently we don't use the ax input parameter, we rather just plot
    in hope for getting an appropriate place for it from the calling function
    """
    # draw unit circle
    Nf = 2**7
    Om = np.arange(Nf) * 2*np.pi/Nf
    plt.plot(np.cos(Om), np.sin(Om), 'C7')
```

Picture 2. Implemented code



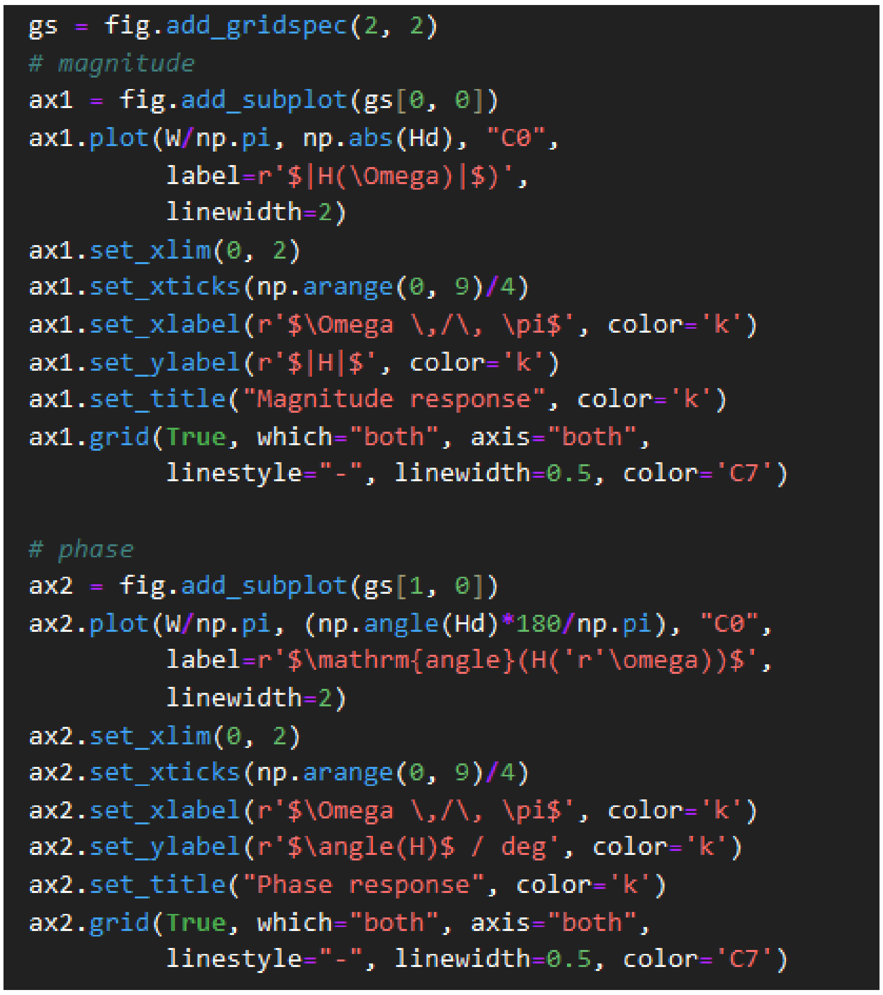
```python
try:  # TBD: check if this pole is compensated by a zero
    circle = Circle((0, 0), radius=np.max(np.abs(p)),
                    color='C7', alpha=0.15)
    plt.gcf().gca().add_artist(circle)
except ValueError:
    print('no pole at all, ROC is whole z-plane')

zu, zc = np.unique(z, return_counts=True)  # find and count unique zeros
for zui, zci in zip(zu, zc):  # plot them individually
    plt.plot(np.real(zui), np.imag(zui), ms=8,
             color='C0', marker='o', fillstyle='none')
    if zci > 1:  # if multiple zeros exist then indicate the count
        plt.text(np.real(zui), np.imag(zui), zci)

pu, pc = np.unique(p, return_counts=True)  # find and count unique poles
for pui, pci in zip(pu, pc):  # plot them individually
    plt.plot(np.real(pui), np.imag(pui), ms=8,
             color='C0', marker='x')
    if pci > 1:  # if multiple poles exist then indicate the count
        plt.text(np.real(pui), np.imag(pui), pci)

plt.text(0, +1, 'k={0:f}'.format(k))
plt.text(0, -1, 'ROC for causal: white')
plt.axis('square')
plt.xlabel(r'$\Re\{z\}$')
plt.ylabel(r'$\Im\{z\}$')
plt.grid(True, which="both", axis="both",
         linestyle="-", linewidth=0.5, color='C7')
```

Picture 3. Implemented code



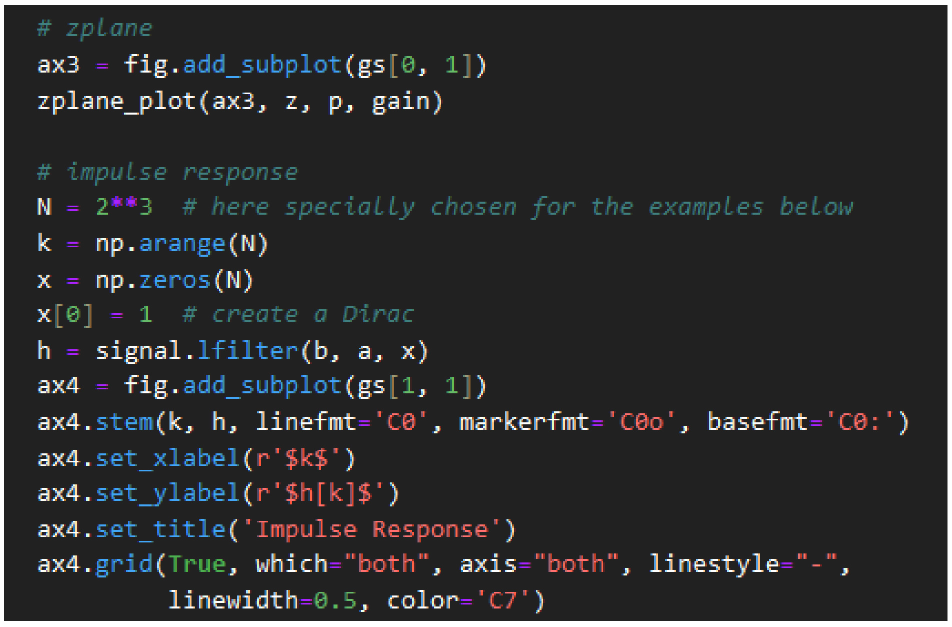
```python
def bode_plot(b, N=2**10, fig=None):  # we use this here for FIRs only
    if fig is None:
        fig = plt.figure()

    a = np.zeros(len(b))  # some scipy packages need len(a)==len(b)
    a[0] = 1

    z, p, gain = signal.tf2zpk(b, a)
    W, Hd = signal.freqz(b, a, N, whole=True)

    print('number of poles:', len(p), '\npole(s) at:', p,
          '\nnumber of zeros:', len(z), '\nzero(s) at:', z)
```

Picture 4. Implemented code



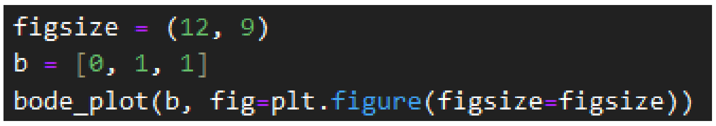
```python
gs = fig.add_gridspec(2, 2)
# magnitude
ax1 = fig.add_subplot(gs[0, 0])
ax1.plot(W/np.pi, np.abs(Hd), "C0",
         label=r'$|H(\Omega)|$)',
         linewidth=2)
ax1.set_xlim(0, 2)
ax1.set_xticks(np.arange(0, 9)/4)
ax1.set_xlabel(r'$\Omega \,/\, \pi$', color='k')
ax1.set_ylabel(r'$|H|$', color='k')
ax1.set_title("Magnitude response", color='k')
ax1.grid(True, which="both", axis="both",
         linestyle="-", linewidth=0.5, color='C7')

# phase
ax2 = fig.add_subplot(gs[1, 0])
ax2.plot(W/np.pi, (np.angle(Hd)*180/np.pi), "C0",
         label=r'$\mathrm{angle}(H('r'\omega))$',
         linewidth=2)
ax2.set_xlim(0, 2)
ax2.set_xticks(np.arange(0, 9)/4)
ax2.set_xlabel(r'$\Omega \,/\, \pi$', color='k')
ax2.set_ylabel(r'$\angle(H)$ / deg', color='k')
ax2.set_title("Phase response", color='k')
ax2.grid(True, which="both", axis="both",
         linestyle="-", linewidth=0.5, color='C7')
```

Picture 5. Implemented code



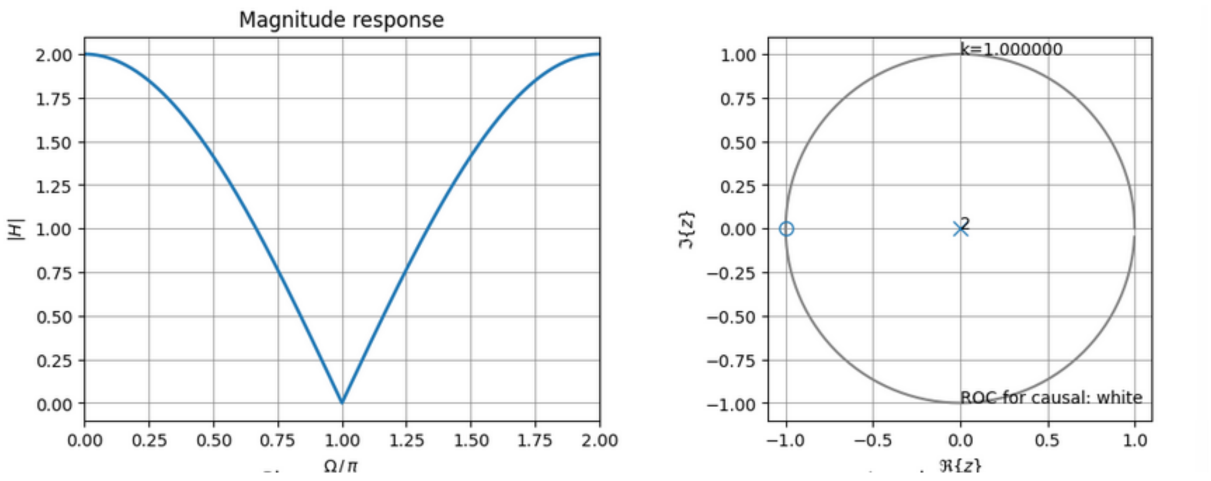
```python
# zplane
ax3 = fig.add_subplot(gs[0, 1])
zplane_plot(ax3, z, p, gain)

# impulse response
N = 2**3  # here specially chosen for the examples below
k = np.arange(N)
x = np.zeros(N)
x[0] = 1  # create a Dirac
h = signal.lfilter(b, a, x)
ax4 = fig.add_subplot(gs[1, 1])
ax4.stem(k, h, linefmt='C0', markerfmt='C0o', basefmt='C0:')
ax4.set_xlabel(r'$k$')
ax4.set_ylabel(r'$h[k]$')
ax4.set_title('Impulse Response')
ax4.grid(True, which="both", axis="both", linestyle="-",
         linewidth=0.5, color='C7')
```

Picture 6. Implemented code

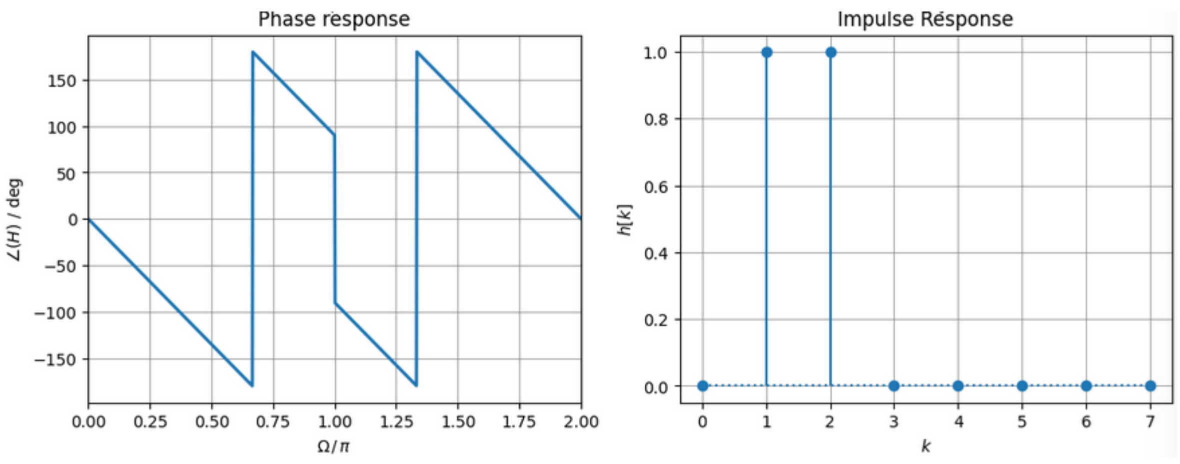


```
figsize = (12, 9)
b = [0, 1, 1]
bode_plot(b, fig=plt.figure(figsize=figsize))
```

Picture 7. Implemented code



Picture 8. The result



Picture 9. The result

5. Conclusions

In this lab, we investigated the Finite Impulse Response (FIR) filtering technique under various filtering parameters. We explored how different filter orders, window functions, and cutoff frequencies impact the performance and characteristics of FIR filters. Through a series of practical exercises, we gained hands-on experience in designing and implementing FIR filters, analyzing their frequency and impulse responses, and evaluating their effectiveness in signal processing applications.