

```

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

np.set_printoptions(precision=16)

def zplane_plot(ax, z, p, k):
    # draw unit circle
    Nf = 2**7
    Om = np.arange(Nf) * 2*np.pi/Nf
    plt.plot(np.cos(Om), np.sin(Om), 'C7')

    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'$\text{Re}\{z\}$')
    plt.ylabel(r'$\text{Im}\{z\}$')
    plt.grid(True, which="both", axis="both",
            linestyle="-", linewidth=0.5, color='C7')

def bode_plot(b, a, N=2**10, fig=None): # for IIR if length of b and
a are the same
    if fig is None:
        fig = plt.figure()

```

```

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)

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'$\omega \backslash, \pi$', 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'$\omega \backslash, \pi$', color='k')
ax2.set_ylabel(r'$\angle(H) \text{ / deg}', color='k')
ax2.set_title("Phase response", color='k')
ax2.grid(True, which="both", axis="both",
         linestyle="-", linewidth=0.5, color='C7')

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

# impulse response
N = 2**4 # here specially chosen for the examples below
k = np.arange(0, N)
x = np.zeros(N)
x[0] = 1
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')

# some defaults for the upcoming code:
figsize = (12, 9)

# taken from lecture's repository
def bilinear_biquad(B, A, fs):
    A0, A1, A2 = A
    B0, B1, B2 = B
    fs2 = fs**2

    a0 = A2 + 2*A1*fs + 4*A0*fs2
    b0 = B2 + 2*B1*fs + 4*B0*fs2

    b1 = 2*B2 - 8*B0*fs2
    a1 = 2*A2 - 8*A0*fs2

    b2 = B2 - 2*B1*fs + 4*B0*fs2
    a2 = A2 - 2*A1*fs + 4*A0*fs2

    b = np.array([b0, b1, b2]) / a0
    a = np.array([a0, a1, a2]) / a0

    return b, a

def f_prewarping(f, fs):
    return 2*fs*np.tan(np.pi*f/fs)

def pz_placement(zr, za, pr, pa):
    z = zr * np.exp(+1j*za)
    p = pr * np.exp(+1j*pa)

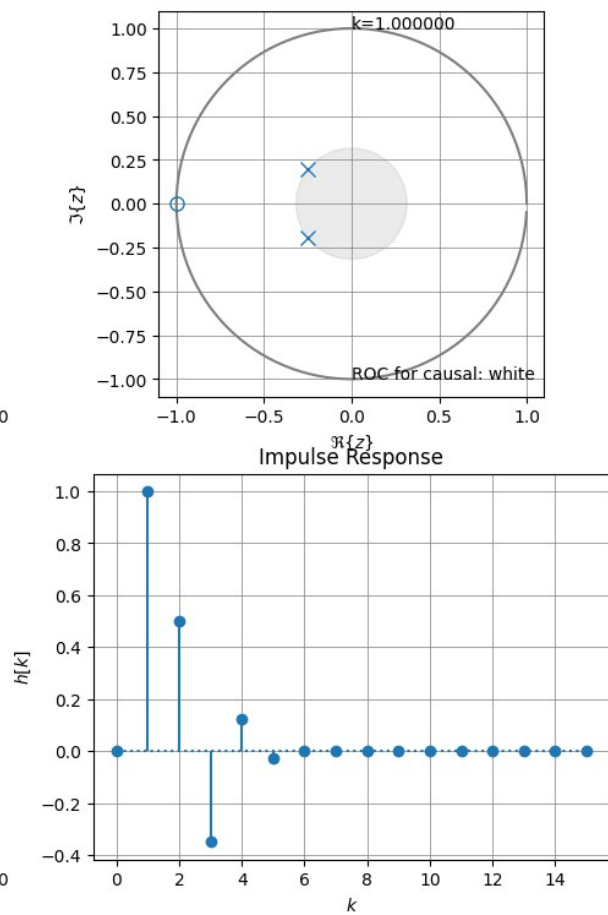
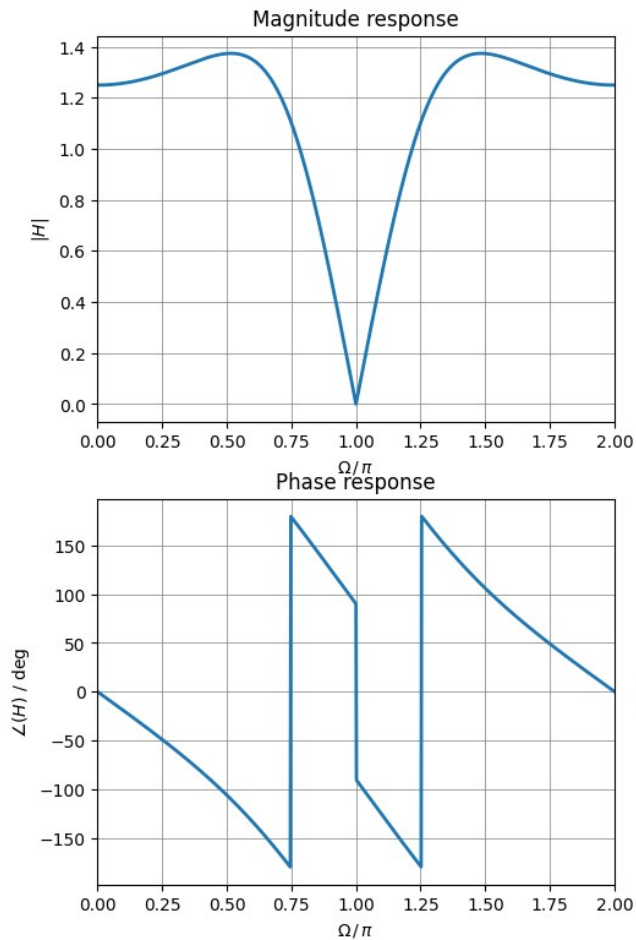
    b = [0, 1, 1]
    a = [1, 0.5, 0.1]

    bode_plot(b, a, fig=plt.figure(figsize=figsize))

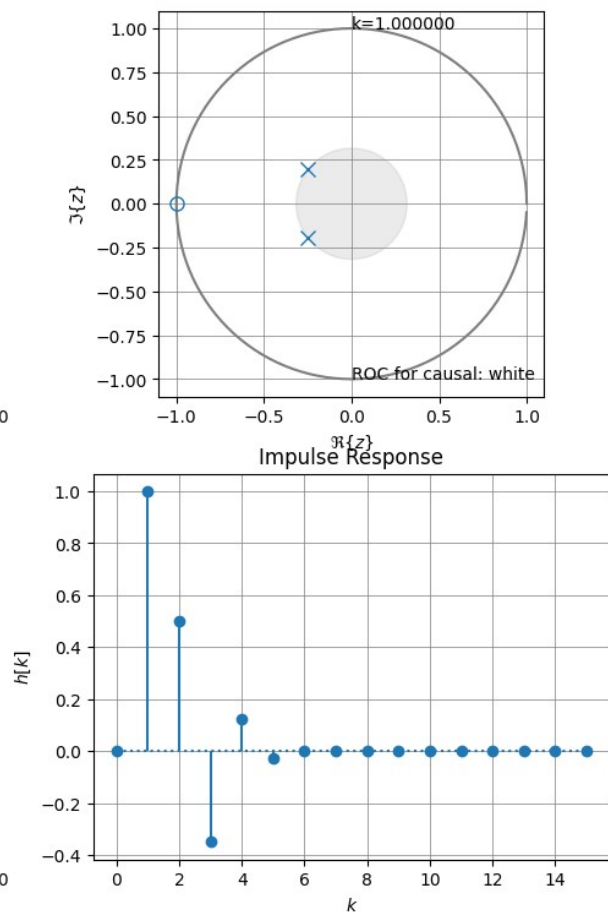
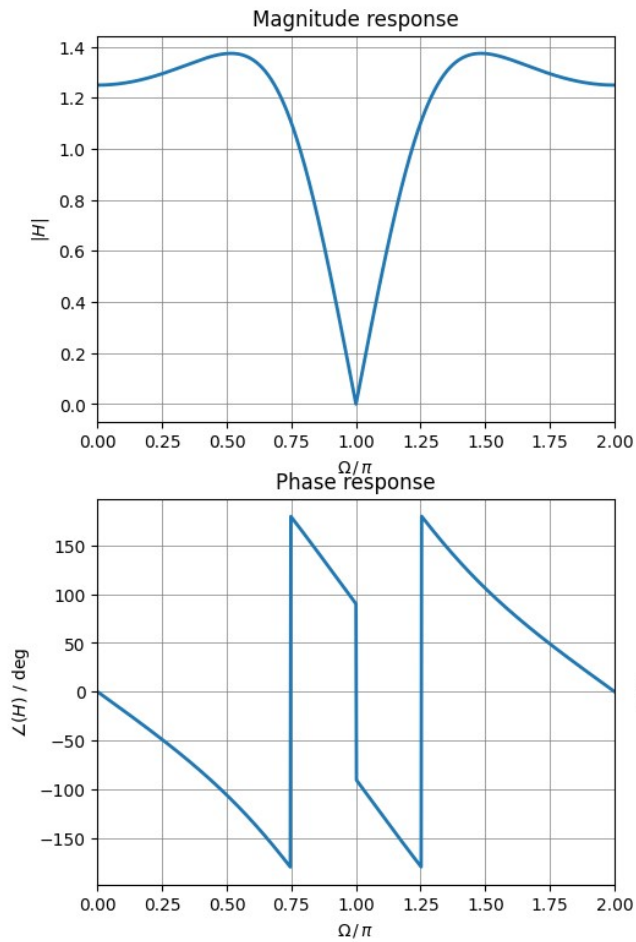
# no IIR actually, but rather FIR just to make a point:
# put poles into origin
pz_placement(zr=1/2, za=np.pi/2, pr=0, pa=0)

C:\Users\aga54\AppData\Local\Programs\Python\Python312\Lib\site-
packages\scipy\signal\_filter_design.py:1109: BadCoefficients: Badly
conditioned filter coefficients (numerator): the results may be
meaningless
    b, a = normalize(b, a)

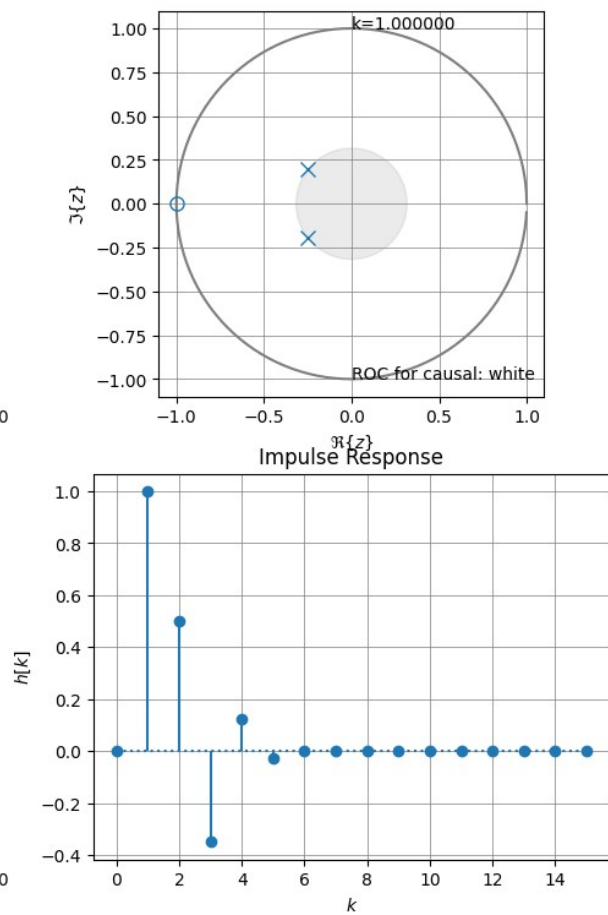
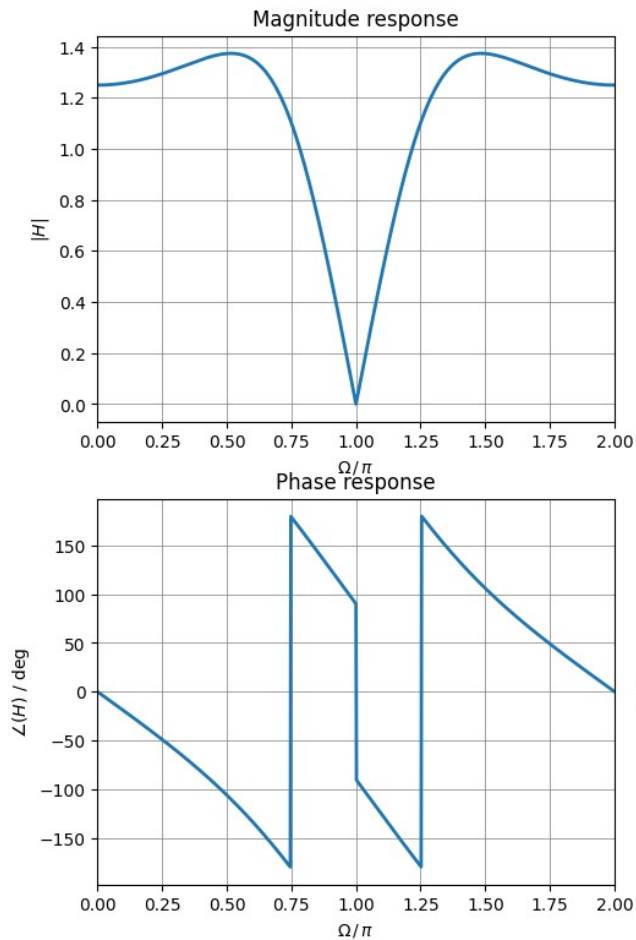
```



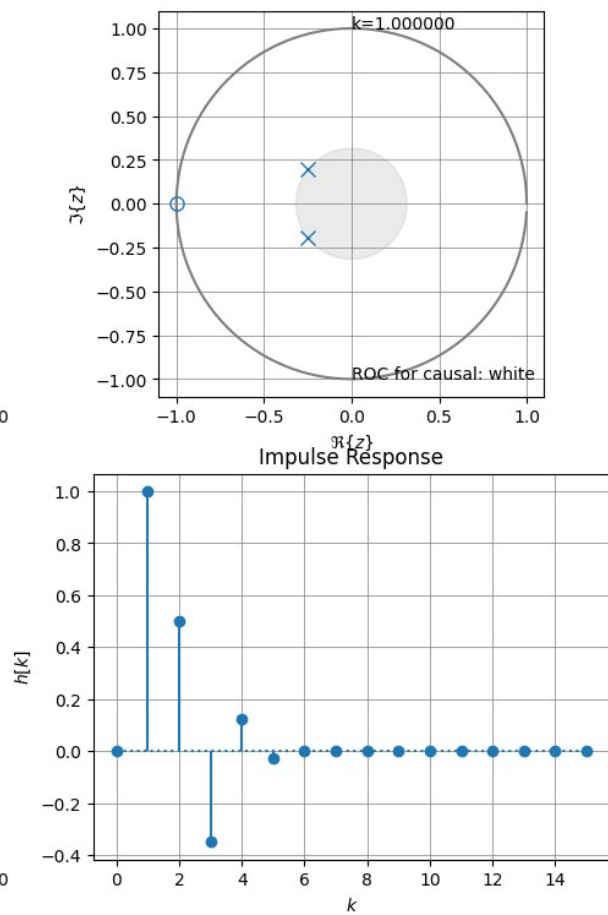
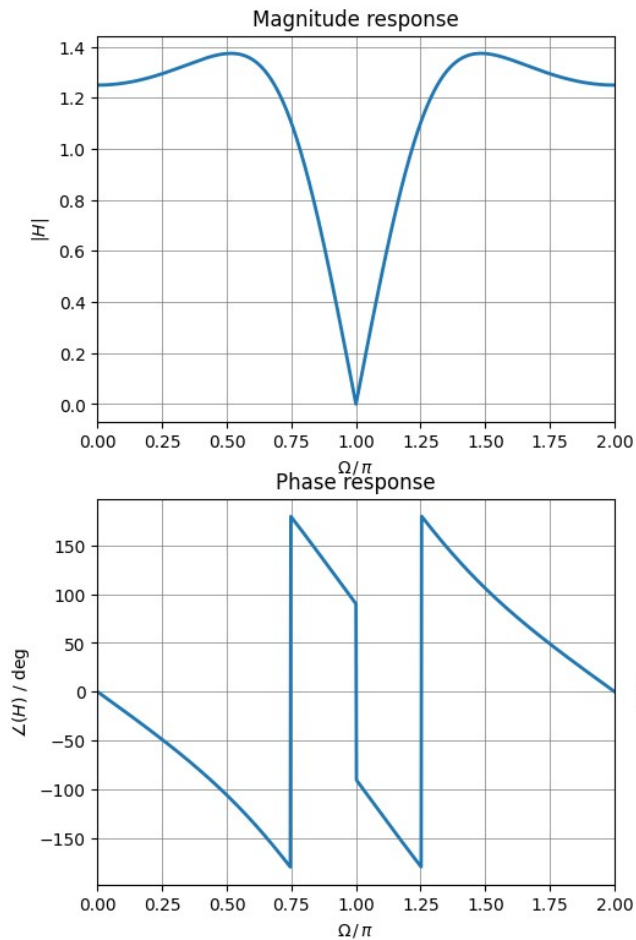
```
# filter transfer function from above can be inverted
# this yields a stable IIR filter, since its poles are inside unit
# and thus white ROC includes the unit circle
pz_placement(zr=0, za=0, pr=1/2, pa=np.pi/2)
```



```
# shift zeros closer to poles
# less ripple in magnitude response
# note that y-axis has changed in comparison to above example
pz_placement(zr=1/3, za=np.pi/2, pr=1/2, pa=np.pi/2)
```



```
# 2nd order lowpass filter
# special here is:
# zero at -1 thus amplitude 0 at  $f_s/2$ , phase -180 deg at  $f_s/2$ 
# two real poles at same location
pz_placement(zr=1, za=np.pi, pr=1/2, pa=0)
```



```
# 2nd order lowpass filter
# zero at -1 thus amplitude 0 at fs/2
# this time complex conjugate pole pair to yield about the same
# magnitude
# at DC
# check the differences between the two filters
# you might create an own plot where both filters can be overlayed
pz_placement(zr=1, za=np.pi, pr=3/5, pa=np.pi/8)
```

