Systems and Signals 414 Practical 4: Poles and zeros

Aim: Understand the effect of poles and zeros on the responses of LTI systems, both in the time and frequency domain.

Handing in: Hand in via your `stnumber@sun.ac.za` email address. (To restate, *use your student number email address*! Do not use a special alias email address such as `johnsmith@sun.ac.za`.) Attach both your Jupyter `.ipynb` notebook file, and an `.html` copy of your output (`File` → `Download as` → `.html`). Email this to `sfstreicher+ss414@gmail.com` with subject as `prac4`. In summary, your email should look like this:

```
Recipient : sfstreicher+ss414@gmail.com
Subject   : prac4
Attachment: c:\...\filename.ipynb
Attachment: c:\...\filename.html
```

You may send your work multiple times; only the last submission will be marked. The process is automated (so we will not actually read the emails; i.e. trying to contact us using `sfstreicher+ss414@gmail.com` is futile). Make sure your `.ipynb` file returns your intended output when run from a clean slate!

Task: Do the following assignment using Jupyter. Document the task indicating your methodology, theoretical results, numerical results and discussions. Graphs should have labelled axes with the correct units indicated.

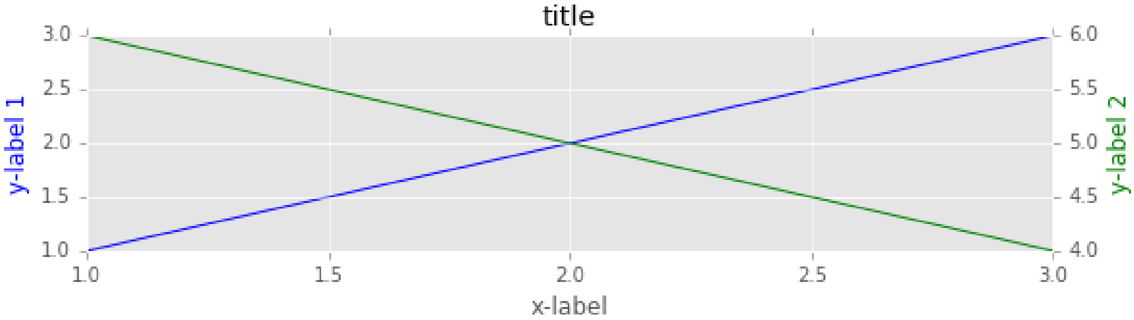
Hints: Plot "continuous-time" signals with Matplotlib's `plot` function, and discrete-time signals with `stem` instead. Here is some useful Jupyter preamble code that you may use for this practical:

In [5]:

```
#All the necessary imports
%matplotlib inline
import pylab as pl
pl.style.use('ggplot') #pretty plots
import numpy as np
from scipy import signal

pl.rcParams['figure.figsize'] = (9,2)

def prepare_plot(title, y_label, x_label):
    pl.figure()
    pl.title(title)
    pl.ylabel(y_label)
    pl.xlabel(x_label)

def twin_plot(xy_data1, xy_data2, title='', xlabel='', ylabel1='', ylabel2=''):
    pl.title(title)
    pl.plot(*xy_data1, color='b')
    pl.ylabel(ylabel1, color='b')
    pl.xlabel(xlabel)

    ax2 = pl.gca().twinx()
    pl.plot(*xy_data2, color='g')
    pl.ylabel(ylabel2, color='g')
```

In [6]:

```
#Example of twin plot
twin_plot(([1,2,3],[1,2,3]), ([1,2,3], [6,5,4]), 'title', 'x-label','y-label 1','y-labe
l 2')
```



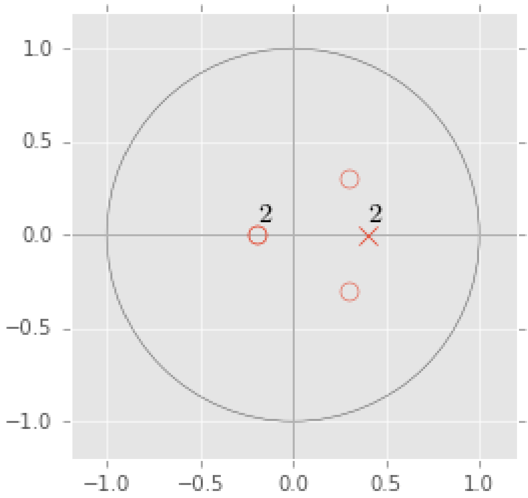
In [7]:

```
#Adapted from https://gist.github.com/endolith/4625838
def zplane(zeros, poles):
    """
    Plot the complex z-plane given zeros and poles.
    """
    zeros=np.array(zeros);
    poles=np.array(poles);
    ax = pl.gca()

    # Add unit circle and zero axes
    unit_circle = pl.matplotlib.patches.Circle((0,0), radius=1, fill=False,
                             color='black', ls='solid', alpha=0.6)
    ax.add_patch(unit_circle)
    pl.axvline(0, color='0.7')
    pl.axhline(0, color='0.7')

    #Rescale to a nice size
    rscale = 1.2 * np.amax(np.concatenate((abs(zeros), abs(poles), [1])))
    pl.axis('scaled')
    pl.axis([-rscale, rscale, -rscale, rscale])

    # Plot the poles and zeros
    polesplot = pl.plot(poles.real, poles.imag, 'x', markersize=9)
    zerosplot = pl.plot(zeros.real, zeros.imag,  'o', markersize=9, color='none',
                         markeredgecolor=polesplot[0].get_color(),
                         )

    #Draw overlap text
    overlap_txt = []
    def draw_overlap_text():
        for txt in overlap_txt:
            try:    txt.remove()
            except: txt.set_visible(False)
        del overlap_txt[:]

        poles_pixel_positions = ax.transData.transform(np.vstack(polesplot[0].get_data
()).T)
        zeros_pixel_positions = ax.transData.transform(np.vstack(zerosplot[0].get_data
()).T)

        for (zps_pixels, zps) in [(poles_pixel_positions, poles), (zeros_pixel_position
s, zeros)]:
            superscript = np.ones(len(zps))
            for i in range(len(zps)):
                for j in range(i+1,len(zps)):
                    if superscript[i]!=-1:
                        if np.all(np.abs(zps_pixels[i] - zps_pixels[j]) < 0.9):
                            superscript[i]+=1;
                            superscript[j]=-1;
            for i in range(len(zps)):
                if superscript[i] > 1:
                    txt = pl.text(zps[i].real, zps[i].imag,
                            r'${}^{%d}$'%superscript[i], fontsize=20
                            )
                    overlap_txt.append(txt)
    draw_overlap_text()

    #Reset when zooming
    def on_zoom_change(axes): draw_overlap_text()
```

```
    ax.callbacks.connect('xlim_changed', on_zoom_change)
    ax.callbacks.connect('ylim_changed', on_zoom_change)
```

In [8]:

```
#Example of poles-zero plot
pl.figure(figsize=(4, 4))
zplane([0.3+0.3j, 0.3-0.3j, -0.2, -0.2], [0.4, 0.4])
```



Scipy provides a good selection of signal processing tools in scipy.signal. Note the following functions important for this practical:

`signal.freqz(b, a, ...)`
We use this function for plotting purposes only. It returns x-y coordinates to help illustrate the frequency response of a filter of type:

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + \dots}{a_0 + a_1 z^{-1} + a_2 z^{-2} + \dots}$$

`signal.lfilter(b, a, x, ...)`
Given a signal $x[n]$, this function will apply the input filter on the input signal, i.e. $x[n] * h[n]$, and return the output signal. Note the input filter is of type:

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + \dots}{a_0 + a_1 z^{-1} + a_2 z^{-2} + \dots}$$

`signal.tf2zpk(b, a)`
This returns poles and zeros parameters $\mathbf{z}$, $\mathbf{p}$, and k as output from filter parameters $\mathbf{b}$ and $\mathbf{a}$ as input, with accordance to:

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + \dots}{a_0 + a_1 z^{-1} + a_2 z^{-2} + \dots} = k\frac{(z - z_0)(z - z_1)\dots}{(z - p_0)(z - p_1)\dots}$$

`signal.zpk2tf(z, p, k)`
The reverse conversion as provided by `signal.tf2zpk`.

`np.unwrap(p, ...)`
Use this function to remedy angular wrapping such as the sawtooth effect you would see with linearly-increasing angles wrapped between $-\pi$ and π.

1. Consider the LTI system (filter) with transfer function $H(z)$ given by

$$H(z) = \frac{1 - 2\cos(\theta_1) r_1 z^{-1} + r_1^2 z^{-2}}{1 - 2\cos(\theta_2) r_2 z^{-1} + r_2^2 z^{-2}},$$

with the following default parameter values:

$$\theta_1 = \frac{3}{8} 2\pi, \quad \theta_2 = \frac{1}{8} 2\pi, \quad r_1 = 0.95, \quad r_2 = 0.95,$$

1. Determine the locations of the poles and zeros of $H(z)$ in terms of θ_1, θ_2, r_1, r_2 by hand. Substitute the parameters with their default values and plot a pole-zero diagram. You may use the given `zplane` function to accomplish this.
2. Keeping the other parameters at their default values, vary r_1 over the intervals $r_1 = \{0.0, 0.5, 0.8, 1.0, 1.05\}$.
   A. Investigate the effect of this variation on the placement of poles and zeros using `zplane`. Note that the next sub-questions can be plotted on the same figure by tinkering with the figure size (such as `pl.figure(figsize=(12,4))`), setting up subplots with `pl.subplot`, and ensuring the labels of a subplot are within its borders with `pl.tight_layout`.
   B. Investigate the effect of this variation on both the magnitude and phase responses of the LTI system with `signal.freqz`. Use linearly scaled axes for frequencies and phases, and decibels $(20 \log_{10}(|A|))$ for amplitudes. Remember to use `np.log10` and not `log`; take note of the functions `np.unwrap` and the provided `twin_plot`.
   C. Investigate the effect of this variation on the impulse response of the system using `signal.lfilter`.
   D. Explain your observations in view of the locations of the poles and zeros of $H(z)$.
3. Repeat Question 2 but now only vary r_2 over the intervals $r_2 = \{0.0, 0.5, 0.8, 1.0, 1.05\}$, and default all other parameters (including r_1).
4. Repeat Question 2 but now only vary θ_1 over the intervals $\theta_1 = \{0, \frac{1}{8}2\pi, \frac{1}{4}2\pi, \frac{3}{8}2\pi, \frac{1}{2}2\pi\}$, and default all other parameters.
5. Repeat Question 2 but now only vary θ_2 over the intervals $\theta_2 = \{0, \frac{1}{8}2\pi, \frac{1}{4}2\pi, \frac{3}{8}2\pi, \frac{1}{2}2\pi\}$, and default all other parameters.
6. Now let $r_2 = 1.0$, while the other parameters take on their default values.
   A. Using `signal.lfilter`, determine the output of the system when sinusoids with frequencies $\omega_1 = 0.11(2\pi)$, $\omega_2 = 0.125(2\pi)$ and $\omega_3 = 0.135(2\pi)$ are applied to it (separately).
   B. Explain your observations in view of a plot of the poles and zeros of $H(z)$.

2. Now consider the LTI system (filter) with transfer function $H(z)$ given by

$$H(z) = \frac{0.0038 + 0.0001z^{-1} + 0.0051z^{-2} + 0.0001z^{-3} + 0.0038z^{-4}}{1 - 3.2821z^{-1} + 4.2360z^{-2} - 2.5275z^{-3} + 0.5865z^{-4}}.$$

1. Determine the system's magnitude and phase response using `signal.freqz`.
2. What type of filter is this system? Verify your answer by filtering a few sinusoids at appropriately chosen frequencies.
3. Sketch the pole-zero diagram for this system. Can you explain the frequency response of the system from this plot?