stability-1

Dan White

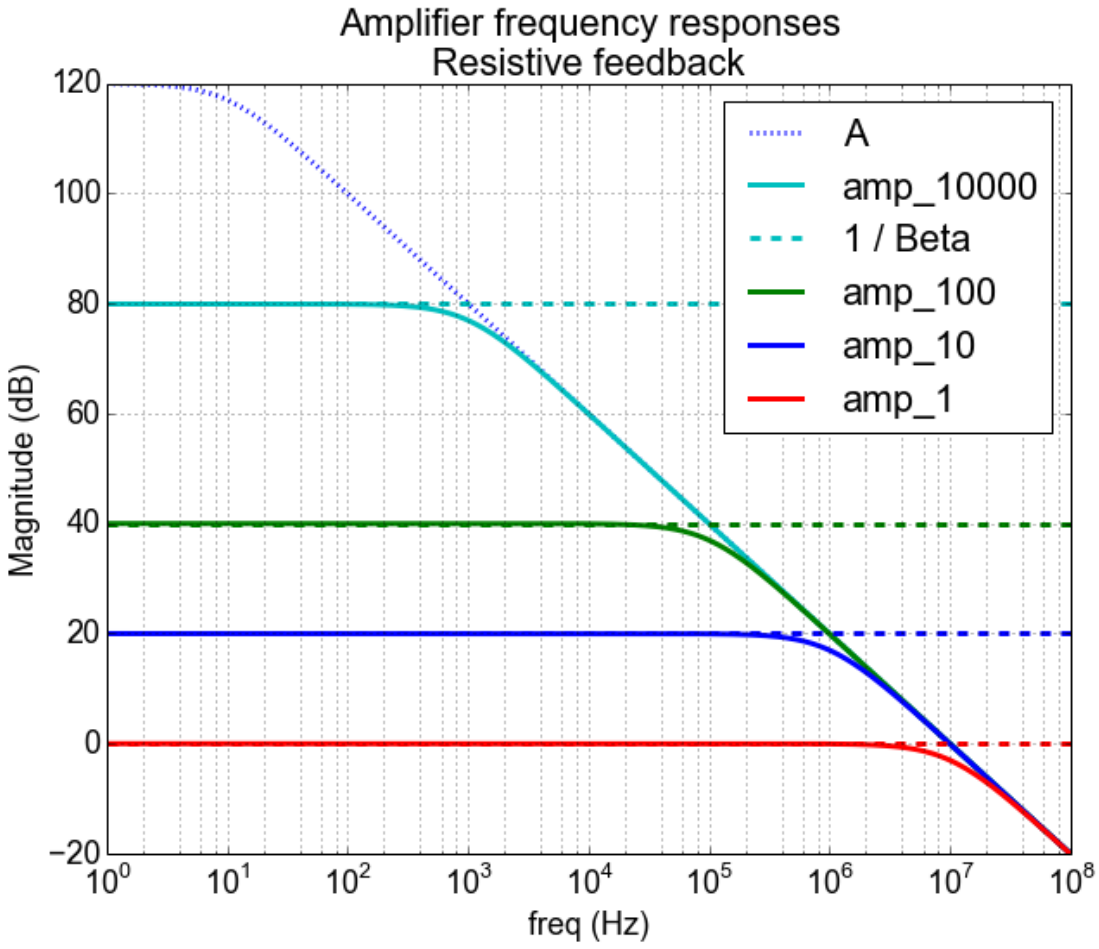
February 19, 2016

In [1]:
```
# this turns on "hold on" mode in the interactive notebook
%config InlineBackend.close_figures=False

import numpy as np
import matplotlib as mpl

mpl.rcParams['axes.grid'] = True
mpl.rcParams['lines.linewidth'] = 3.0

close('all')

def fbamp(A, B):
    def response(s):
        return (A(s)/(1 + A(s)*B(s)))
    return response


def A_onepole(Avol, fb):
    def response(s):
        return (Avol * fb / (s + fb))
    return response


def dB(x):
    return 20*log10(abs(x))
```

In [2]:
```
f = np.logspace(0, 8, 1e3)

Avol = 1e6
fb = 10

A = A_onepole(Avol, fb)


def Beta(b):
    """Return Beta-versus-s that is just the constant b."""
    return lambda s: b


def amp(s, gain):
    av = fbamp(A, Beta(1.0/gain))
    return av(s)



figure()  #needed since we are in "hold on" mode
semilogx(f, dB(A(1j*f)), ':b', label='A')

semilogx(f, dB(amp(1j*f, 10000)), '-c', label='amp_10000')
semilogx(f, dB(10000*ones_like(f)), '--c', label='1 / Beta')
```

```
semilogx(f, dB(amp(1j*f, 100)), '-g', label='amp_100')
semilogx(f, dB(100*ones_like(f)), '--g')

semilogx(f, dB(amp(1j*f, 10)), '-b', label='amp_10')
semilogx(f, dB(10*ones_like(f)), '--b')

semilogx(f, dB(amp(1j*f, 1)), '-r', label='amp_1')
semilogx(f, dB(1.0*ones_like(f)), '--r')


title('Amplifier frequency responses\nResistive feedback')
ylabel('Magnitude (dB)')
xlabel('freq (Hz)')
ylim([-20, dB(Avol)])
legend()

print 'Opamp GBW: %.2e Hz' % (Avol*fb)
```

```
Opamp GBW: 1.00e+07 Hz
```

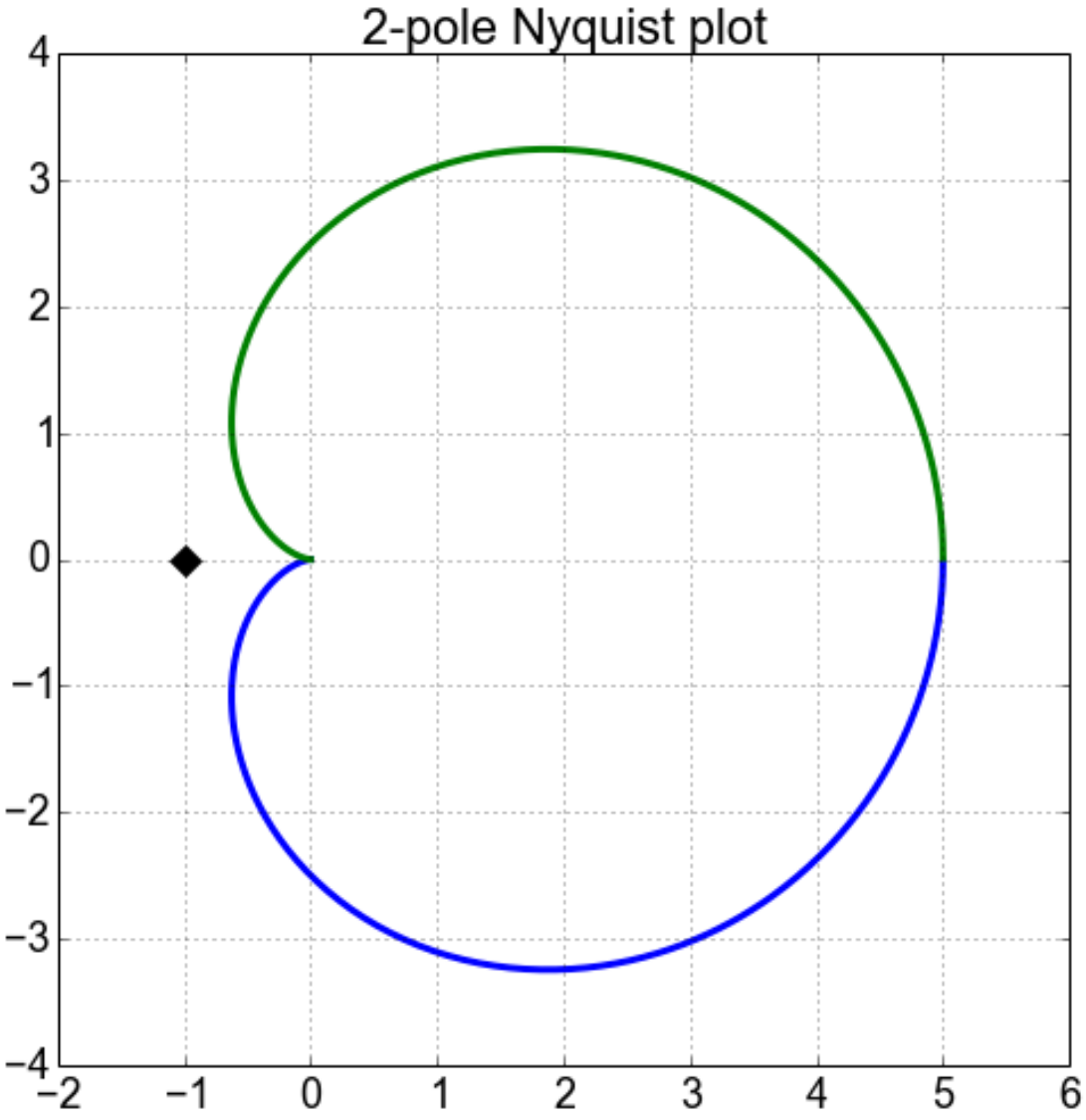


In [3]:
```
np.math.atan2(-10000, 1) *180/pi
```

Out [3]:
```
-89.99427042206779
```

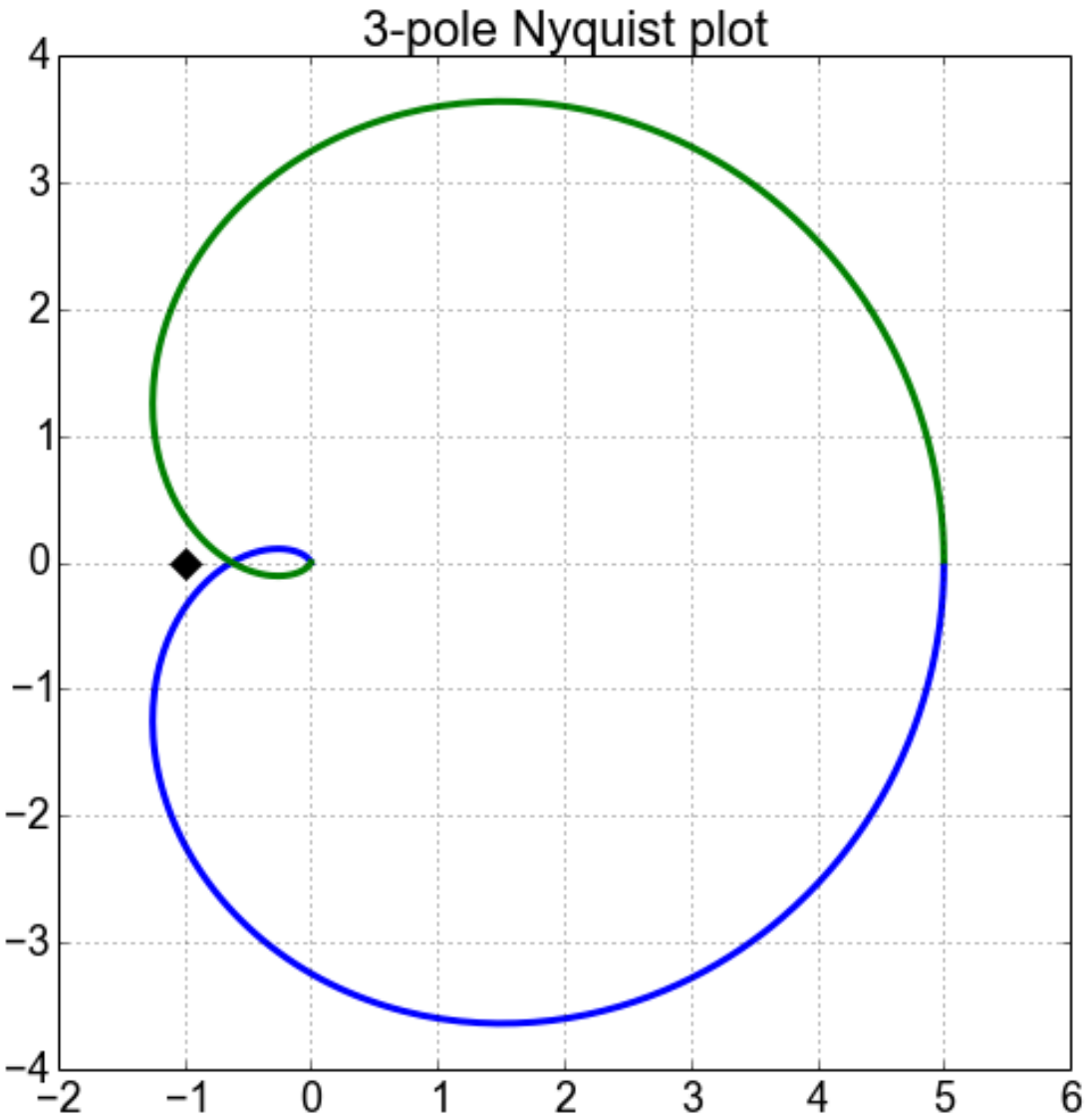
In [33]:
```
def A_2pole(Avol, f1, f2):
    def response(s):
        return (Avol / ((1 + s/f1) * (1 + s/f2)))
    return response
```

```
f = logspace(0, 7, 1e3)

amp = A_2pole(5, 1e3, 1e3)

Tp = amp(1j*f)
Tm = amp(-1j*f)

close('all')
fig=figure()
title('2-pole Nyquist plot')
plot(real(Tp), imag(Tp))
plot(real(Tm), imag(Tm))
plot(-1, 0, 'Dk', markersize=10)
xlim([-2, 6])
gca().set_aspect('equal')
```



In [34]:

```
def A_3pole(Avol, f1, f2, f3):
    def response(s):
        return (Avol / ((1 + s/f1) * (1 + s/f2) * (1 + s/f3)))
    return response


f = logspace(0, 7, 1e3)

amp = A_3pole(5, 1e3, 1e3, 1e3)

Tp = amp(1j*f)
Tm = amp(-1j*f)

close('all')
fig=figure()
title('3-pole Nyquist plot')
plot(real(Tp), imag(Tp))
plot(real(Tm), imag(Tm))
plot(-1, 0, 'Dk', markersize=10)
xlim([-2, 6])
gca().set_aspect('equal')
```



In []: