

1. Find step response of $\dot{y} + y = x$

$$\mathcal{L}\{\dot{y} + y = x\} = sY + Y = X \quad \mathcal{L}\{u(t)\}$$

$$sY + Y = \frac{1}{s}$$

$$Y(s+1) = \frac{1}{s} \quad Y = \frac{1}{s(s+1)} = \frac{A}{s} + \frac{B}{(s+1)} \quad \begin{matrix} A(s+1) + Bs = 1 \\ \text{if } A=1, B=-1 \end{matrix}$$

$$Y = \frac{1}{s} - \frac{1}{s+1} \rightarrow \mathcal{L}^{-1}\{Y = \frac{1}{s} - \frac{1}{s+1}\} \rightarrow y(t) = u(t) - u(t)e^{-t}$$

$$\boxed{y(t) = (1 - e^{-t})u(t)}$$

2A. Find DC gain of $Y(s)/Y_{sp}(s)$ if $K(s) = \frac{K_I}{s}$. Does it depend on K_I ?

$$\text{DC gain} = \lim_{s \rightarrow 0} \frac{Y(s)}{Y_{sp}(s)} = \lim_{s \rightarrow 0} \frac{KH}{1+KH} = \lim_{s \rightarrow 0} \frac{\frac{K_I H}{s}}{1 + \frac{K_I H}{s}} = \lim_{s \rightarrow 0} \frac{K_I H}{s + K_I H} = \boxed{1}$$

It doesn't depend on K_I !

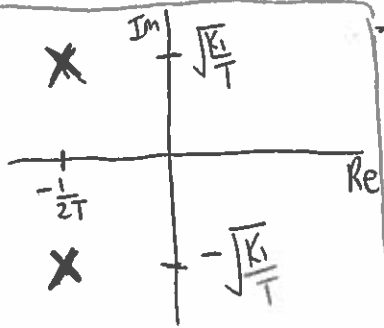
2B. Assume $H(s) = \frac{Y_T}{s + Y_T}$. Find poles if $K_I \gg Y_T$. Draw zero pole diagram.

$$\frac{Y(s)}{Y_{sp}(s)} = \frac{\frac{K_I}{s} \cdot \frac{Y_T}{s + Y_T}}{1 + \frac{K_I}{s} \cdot \frac{Y_T}{s + Y_T}} = \frac{\frac{K_I Y_T}{s}}{s(s + \frac{1}{T}) + K_I/T}$$

→ no zeros since K_I is $\gg \frac{1}{T}$

→ Poles when $s^2 + s\frac{1}{T} + \frac{K_I}{T} = 0$
 $a=1, b=\frac{1}{T}, c=\frac{K_I}{T}$

Poles when $s = \frac{-\frac{1}{T} \pm \sqrt{\frac{1}{T^2} - 4\frac{K_I}{T}}}{2}$ since $K_I \gg T \rightarrow s \approx \frac{-\frac{1}{T} \pm \sqrt{-\frac{4K_I}{T}}}{2} \approx \frac{-\frac{1}{2T} \pm i\sqrt{\frac{K_I}{T}}}{2}$ poles



← pole-zero diagram

4B. Proportional Control

$$H(s) = \frac{1}{s^2 - 0.01s + 1}$$

Calculations for #4

$$K = K_p$$

$$\frac{Y(s)}{Y_{sp}(s)} = \frac{KH}{1+KH} = \frac{\frac{K_p}{s^2 - 0.01s + 1}}{1 + \frac{K_p}{s^2 - 0.01s + 1}} \cdot \frac{s^2 - 0.01s + 1}{s^2 - 0.01s + 1} = \boxed{\frac{K_p}{s^2 - 0.01s + 1 + K_p}}$$

4C. Integral Control $K = \frac{K_I}{s}$

$$\frac{Y(s)}{Y_{sp}(s)} = \frac{KH}{1+KH} = \frac{\frac{K_I}{s^3 - 0.01s^2 + s}}{1 + \frac{K_I}{s^3 - 0.01s^2 + s}} \cdot \frac{s^3 - 0.01s^2 + s}{s^3 - 0.01s^2 + s} = \boxed{\frac{K_I}{s^3 - 0.01s^2 + s + K_I}}$$

4D. Differential Control $K = sK_D$

$$\frac{Y(s)}{Y_{sp}(s)} = \frac{KH}{1+KH} = \frac{\frac{sK_D}{s^2 - 0.01s + 1}}{1 + \frac{sK_D}{s^2 - 0.01s + 1}} \cdot \frac{s^2 - 0.01s + 1}{s^2 - 0.01s + 1} = \frac{sK_D}{s^2 - 0.01s + 1 + sK_D}$$

$$= \boxed{\frac{sK_D}{s^2 + (K_D - 0.01)s + 1}}$$

```
In [3]: %matplotlib inline
from __future__ import print_function
import numpy as np
import matplotlib.pyplot as plt
from scipy import signal
import convenience
np.set_printoptions(precision=2,suppress=True) # numpy output options
pi=np.pi
j=1j
```

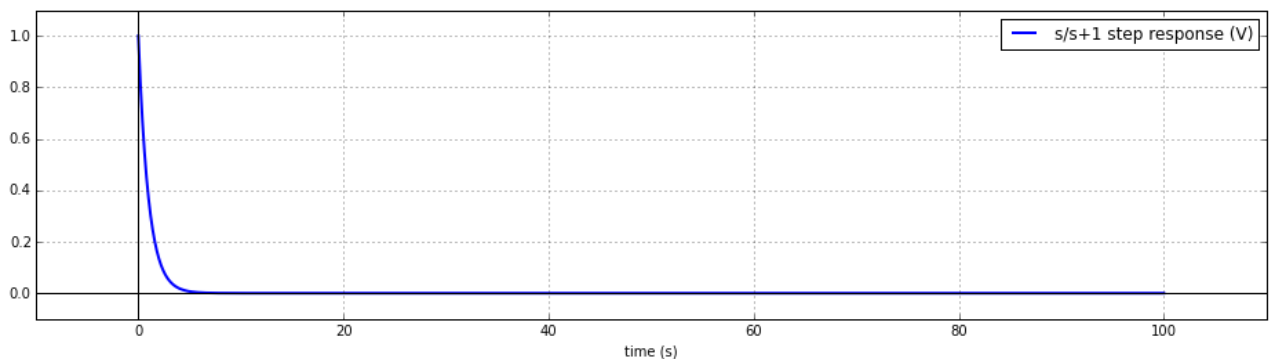
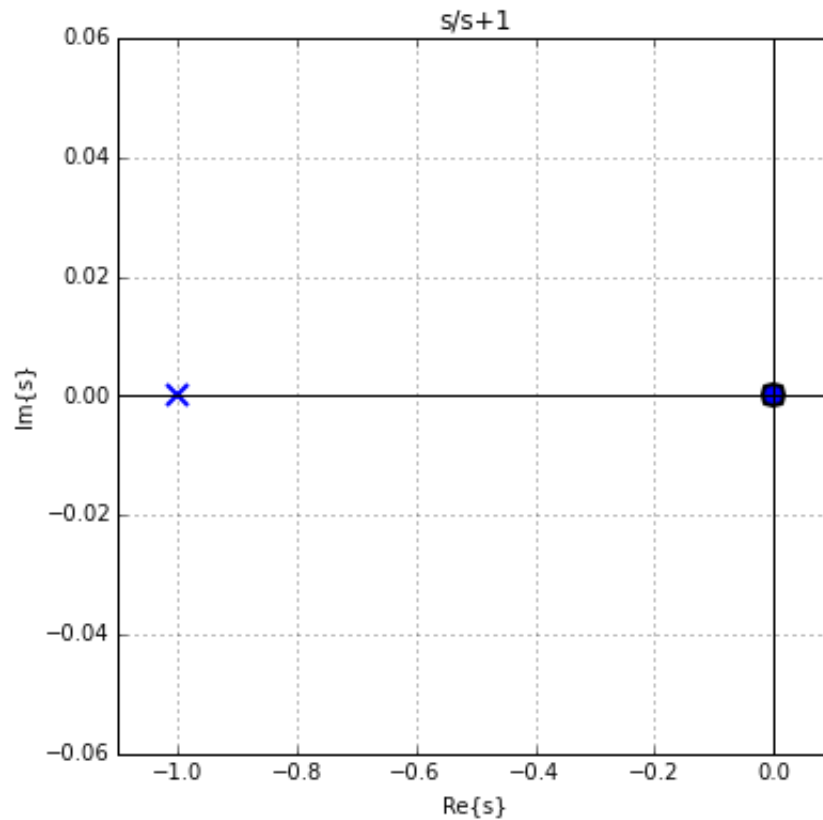
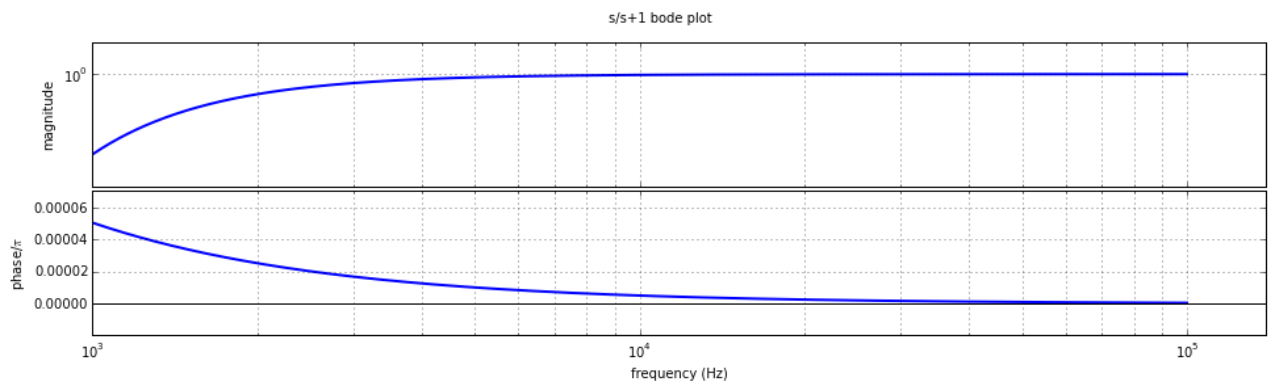
```
In [4]: def get_system(lti_args):
        num_coeffs = lti_args[0]
        denom_coeffs = lti_args[1]
        return signal.lti(num_coeffs, denom_coeffs)

def plot_bode(title, system, freqs=np.logspace(3,5,500)):
    _w, mag, phase = signal.bode(system,w=2*pi*freqs)#Bode takes frequ
encies in rad/s and returns magnitudes in dB
    convenience.logplot(freqs,10**((mag/20),phase,title=title+' bode pl
ot',margins=0.4)
    plt.show()

def plot_step_response(title, system, t=np.linspace(0,100,500)):
    _t,s=signal.step(system,T=t)
    convenience.timeplot(t,s,label=title+' step response (V)')
    plt.show()

def plot_pole_zeros(title,system):
    convenience.pzmap(system, title=title)
    plt.show()
```

```
In [29]: #3A
title = 's/s+1'
lti_args = ([1,0],[1,1])
system = get_system(lti_args)
plot_bode(title,system)
plot_pole_zeros(title,system)
plot_step_response(title,system)
```



3A: This first-order system acts as a high pass filter. There is one pole and one zero, both on the real axis. The step response looks like a flipped and smoothed out version of the step function, since the system is a high pass.

In [35]:

#3B

```
title = 's/(s^2+100s+1)'
```

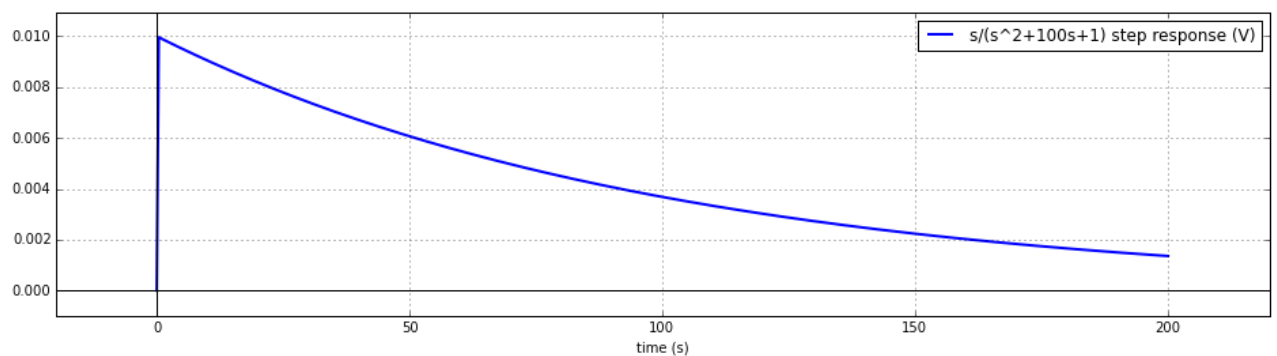
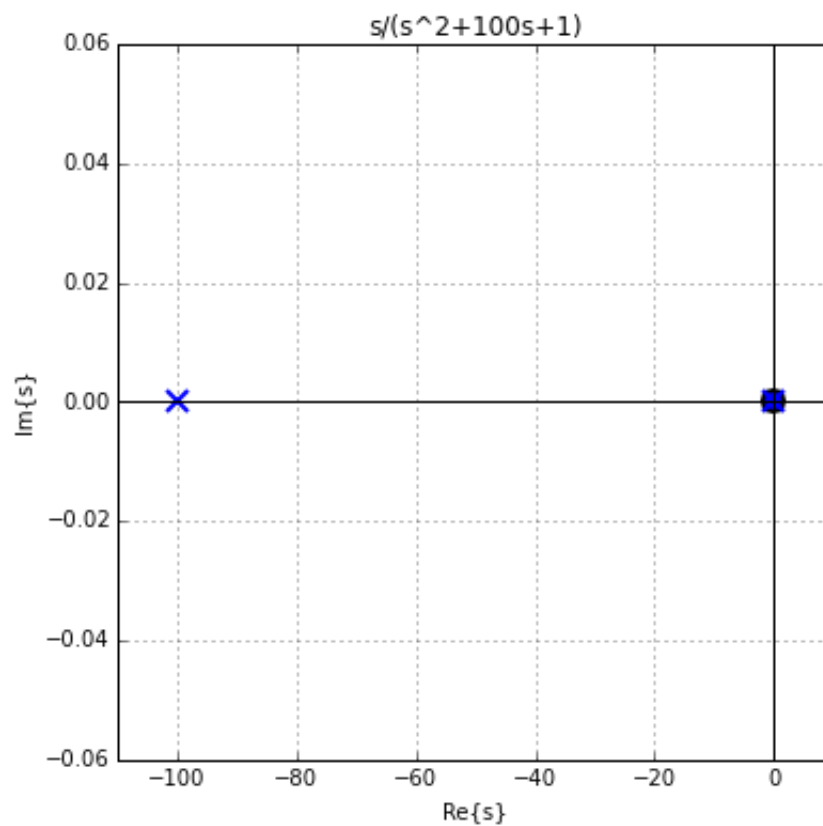
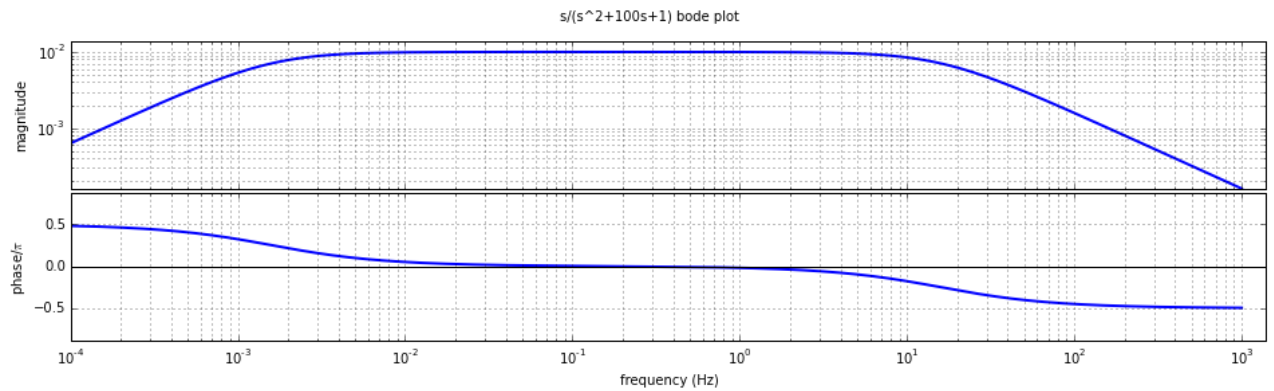
```
lti_args = ([1,0],[1,100,1])
```

```
system = get_system(lti_args)
```

```
plot_bode(title,system,np.logspace(-4,3,500))
```

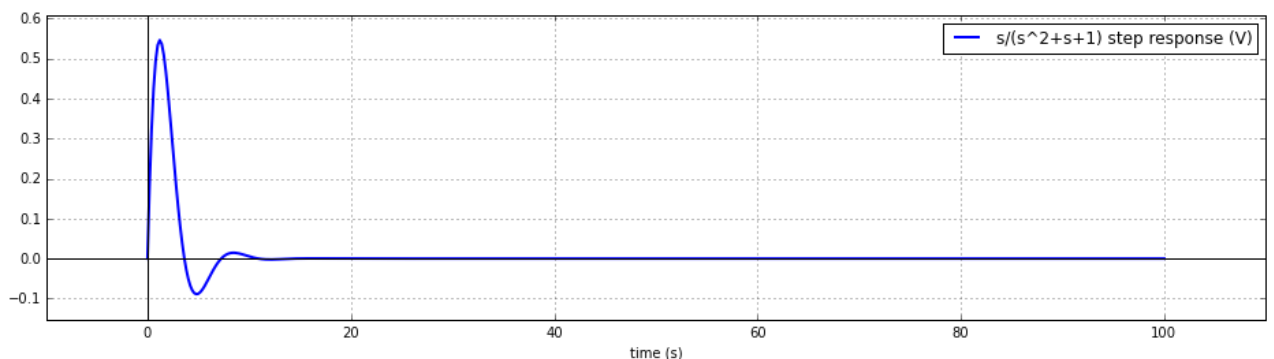
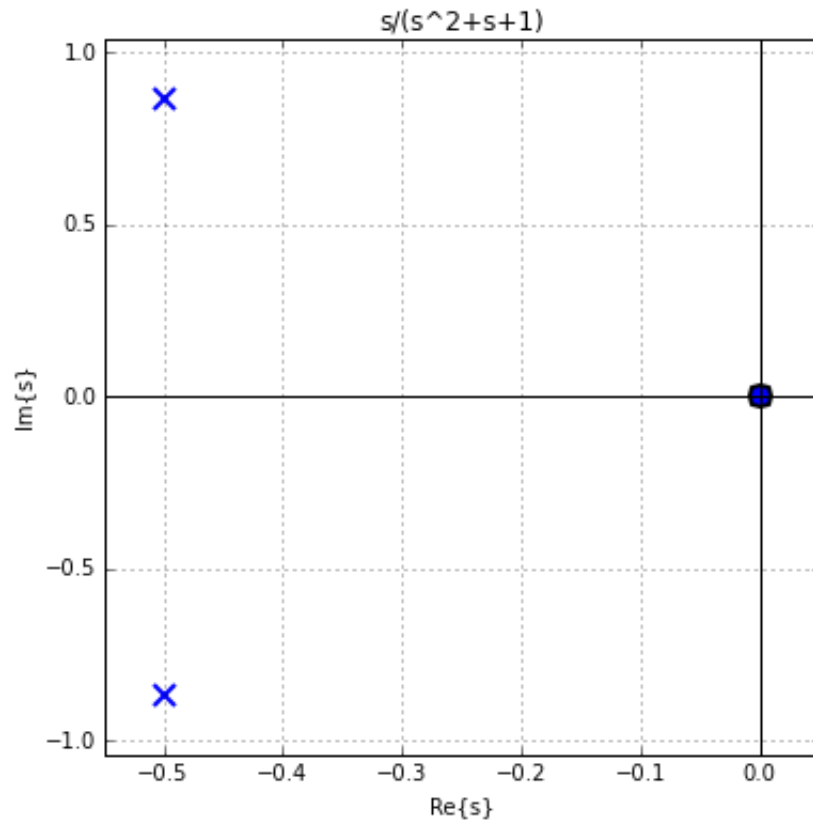
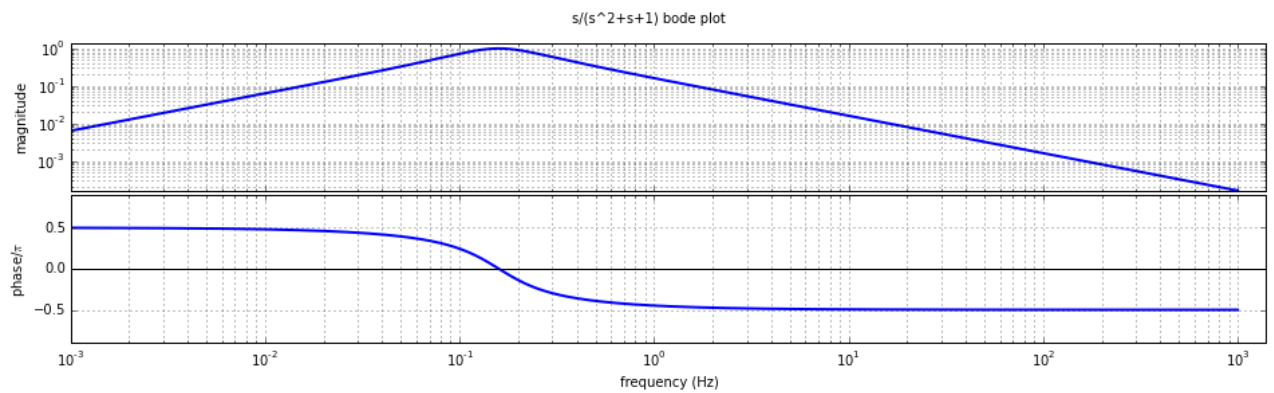
```
plot_pole_zeros(title,system)
```

```
plot_step_response(title,system,t=np.linspace(0,200,500))
```



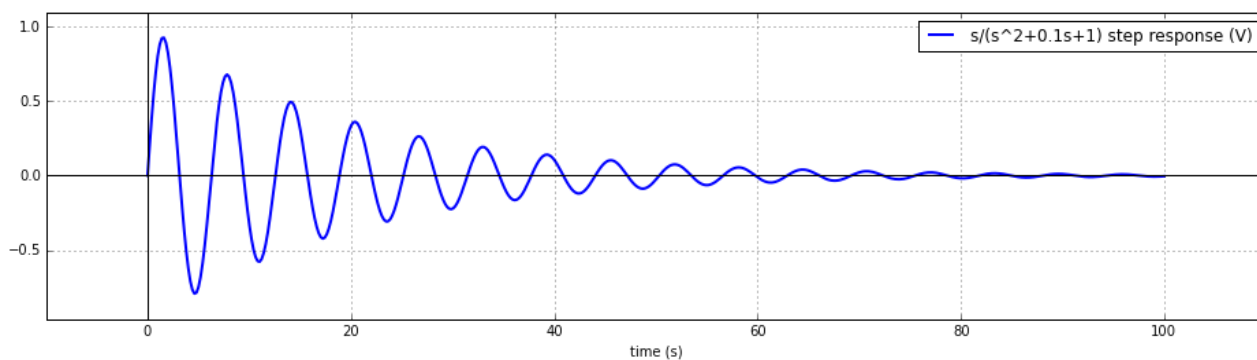
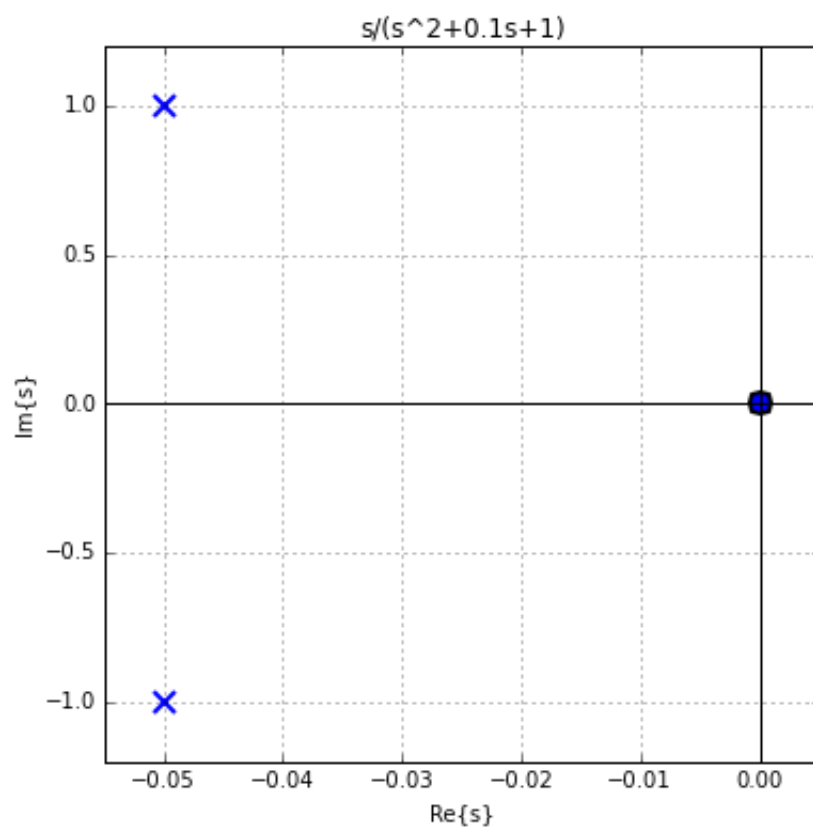
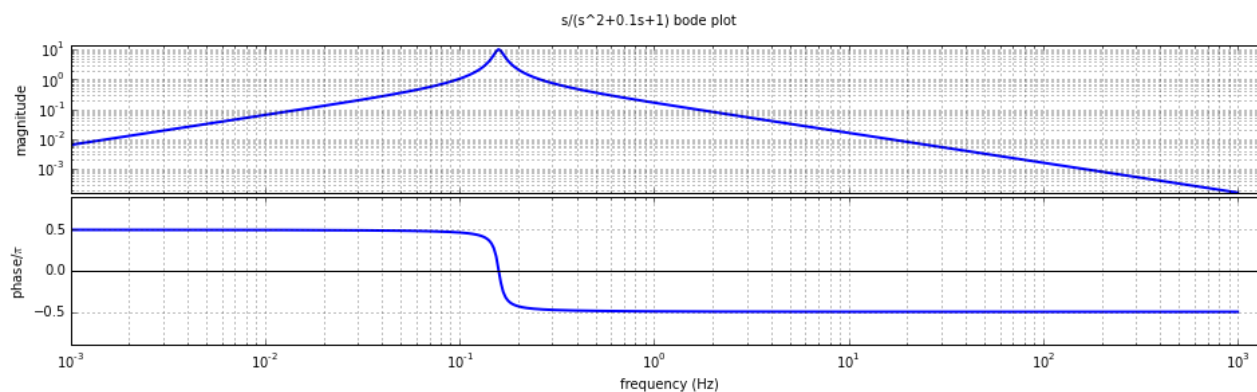
3B: This second order system acts as a bandpass filter. There is one pole and one zero, both on the real axis. There is no oscillation and the step response slowly dies off. It looks like there is no oscillation when the poles and zeros are on the real axis.

```
In [20]: #3C
title = 's/(s^2+s+1)'
lti_args = ([1,0],[1,1,1])
system = get_system(lti_args)
plot_bode(title,system,np.logspace(-3,3,500))
plot_pole_zeros(title,system)
plot_step_response(title,system)
```



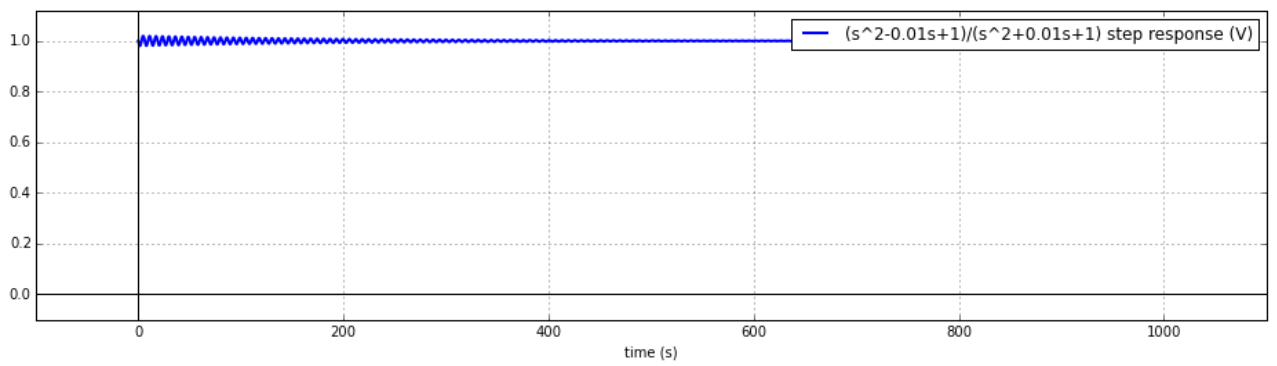
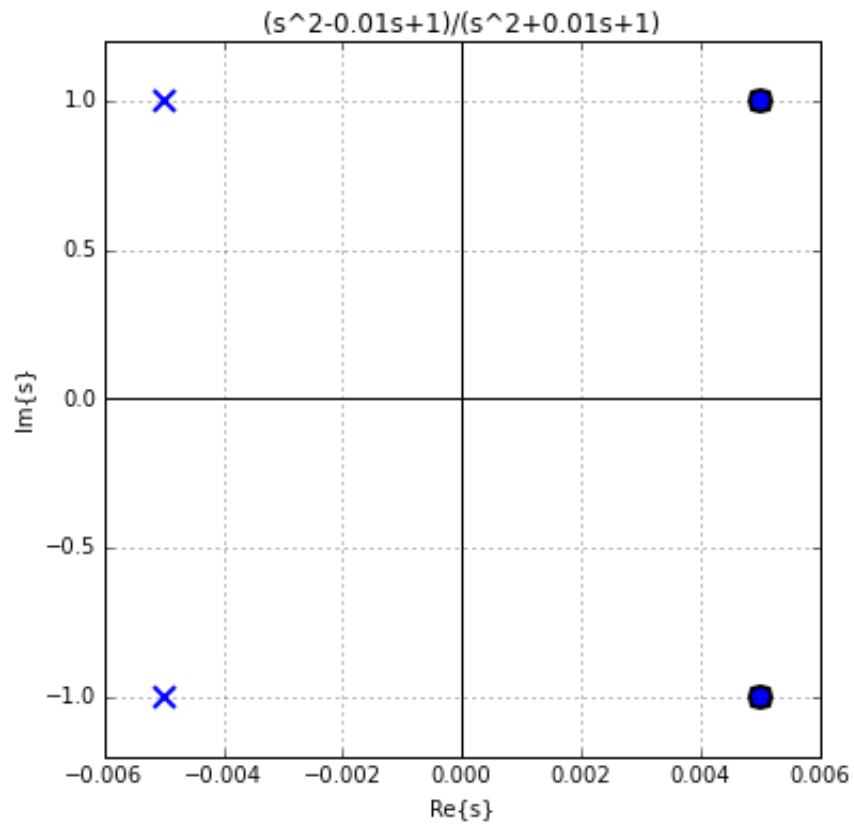
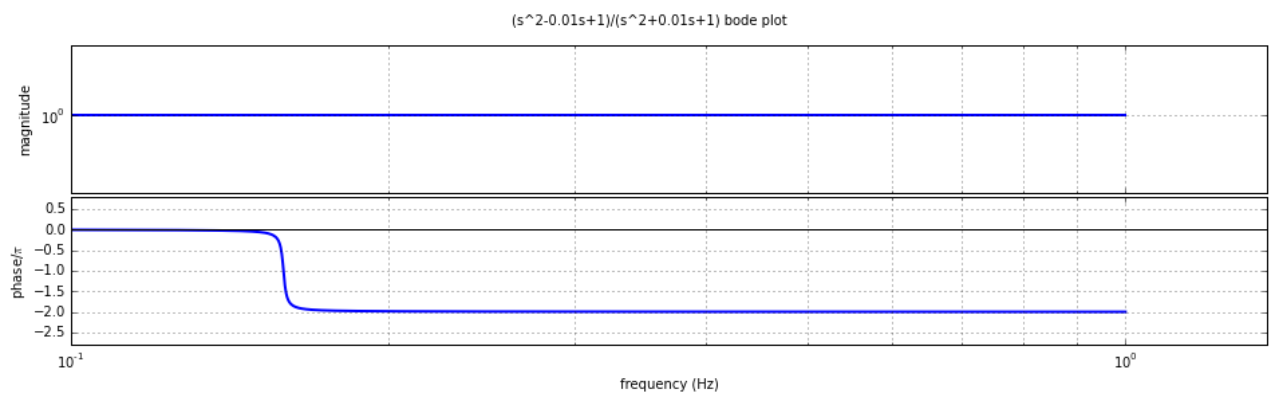
3C: This second order system has a peak response at a specific frequency and dies off elsewhere. There is damped oscillation in the system, and there are 2 imaginary poles and one real zero. It looks like having imaginary poles causes oscillation.

```
In [22]: #3D
title = 's/(s^2+0.1s+1)'
lti_args = ([1,0],[1,0.1,1])
system = get_system(lti_args)
plot_bode(title,system,np.logspace(-3,3,500))
plot_pole_zeros(title,system)
plot_step_response(title,system)
```



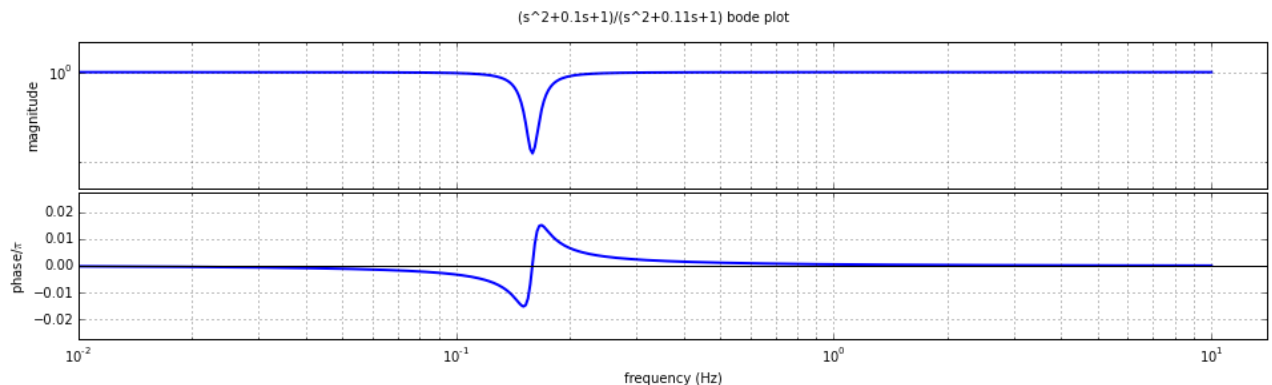
3D: This 2nd order system looks similar to the system in #3C except the oscillations don't die out as quickly. It has a bode plot with a peak at a specific frequency, and it has one real zero and two imaginary poles. The poles are closer to the zero on the imaginary axis in 3D than in 3C, so a guess I have is that having closer poles to the zero causes more oscillation in the step response.

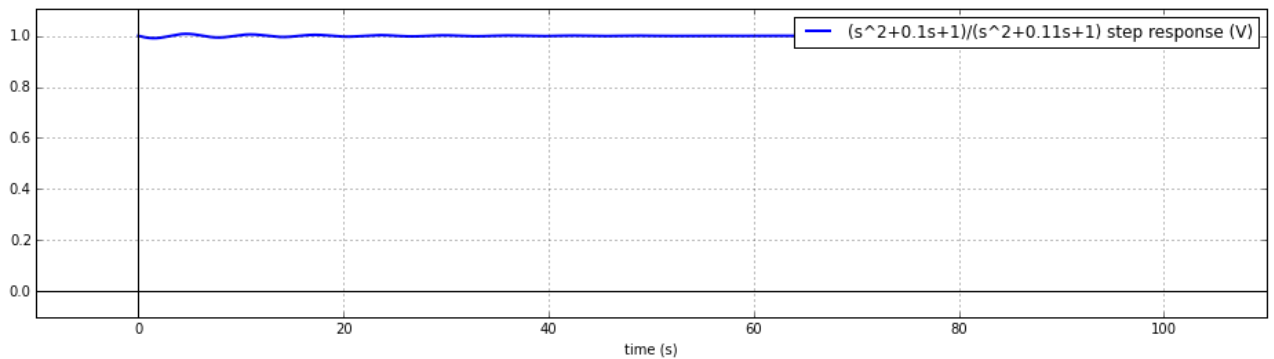
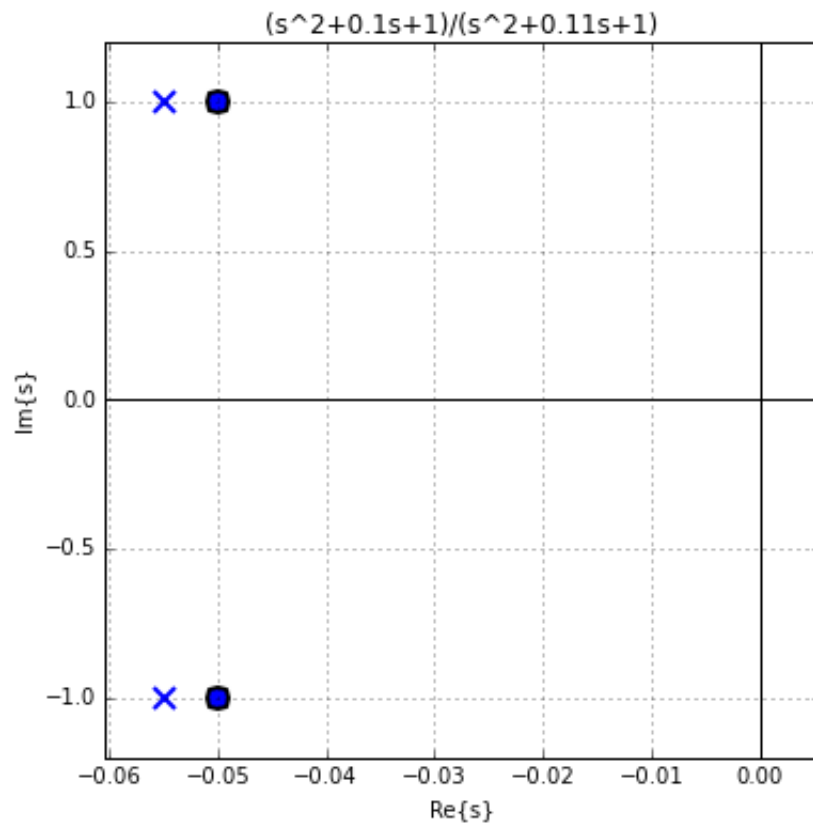
```
In [9]: #3E
title = '(s^2-0.01s+1)/(s^2+0.01s+1)'
lti_args = ([1,-0.01,1],[1,0.01,1])
system = get_system(lti_args)
plot_bode(title,system,np.logspace(-1,0,1000))
plot_pole_zeros(title,system)
plot_step_response(title,system,t=np.linspace(0,1000,5000))
```



3E: The bode plot indicates that this 2nd order system does not amplify/kill off any frequencies, but it does shift the phase of certain frequencies. There are a pair of imaginary poles and a pair of imaginary zeros as well. The step response oscillates at 1 and then slowly dies off. It looks like having imaginary zeros shifted the step response to center around 1 instead of 0 as seen in previous examples. The oscillation is probably because of the imaginary poles.

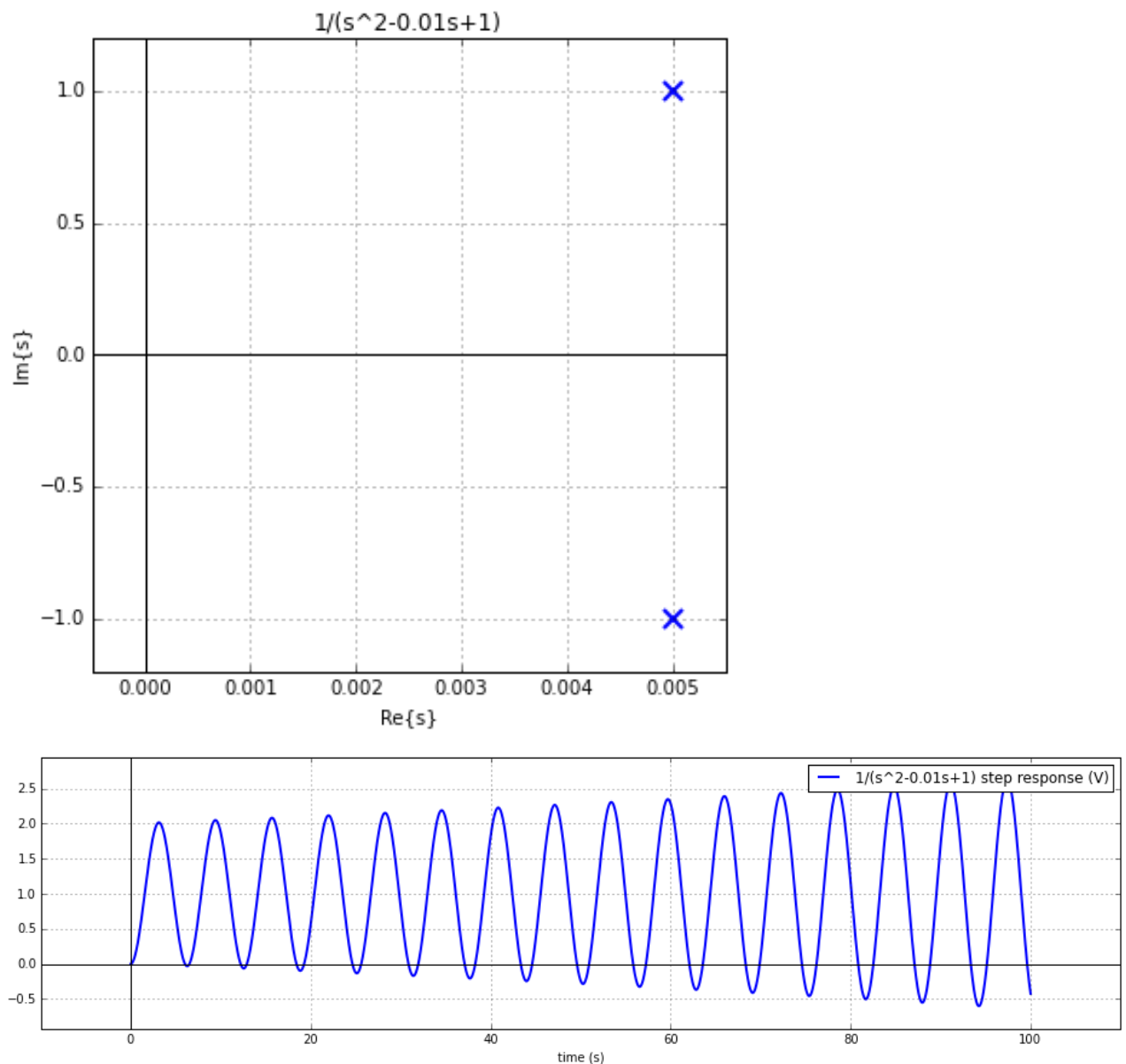
```
In [8]: #3F
title = '(s^2+0.1s+1)/(s^2+0.11s+1)'
lti_args = ([1,0.1,1],[1,0.11,1])
system = get_system(lti_args)
plot_bode(title,system,np.logspace(-2,1,500))
plot_pole_zeros(title,system)
plot_step_response(title,system,t=np.linspace(0,100,5000))
```





3F: The bode plot for this system looks like a notch filter that kills off very specific frequencies. There are a pair of imaginary poles and a pair of imaginary zeros, that are both to the left of the imaginary axis and close together. This seems to cause the step response to die off far more quickly than in the previous example.

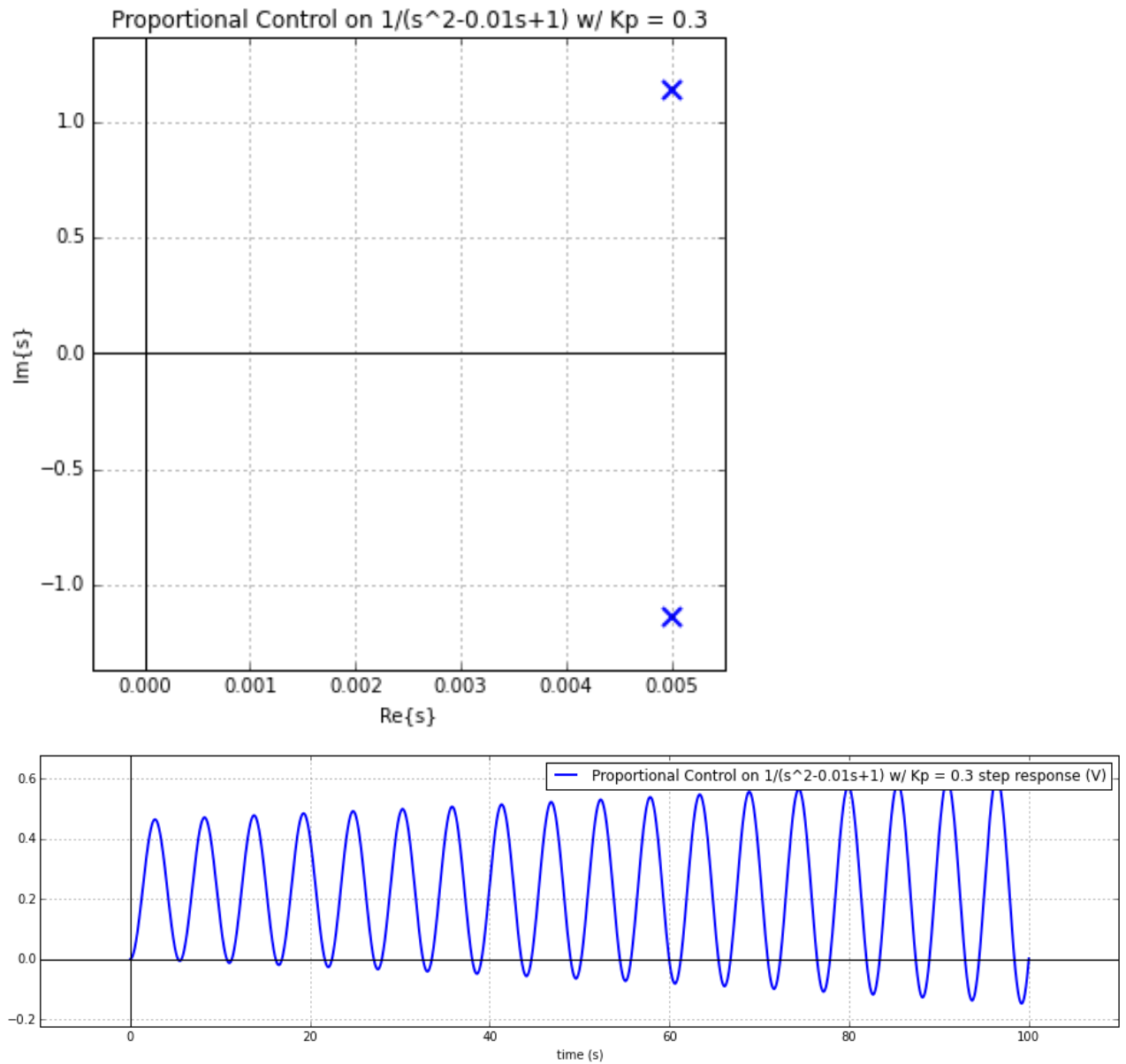
```
In [11]: #4A
title = '1/(s^2-0.01s+1)'
lti_args = ([1],[1,-0.01,1])
system = get_system(lti_args)
plot_pole_zeros(title,system)
plot_step_response(title,system,t=np.linspace(0,100,5000))
```



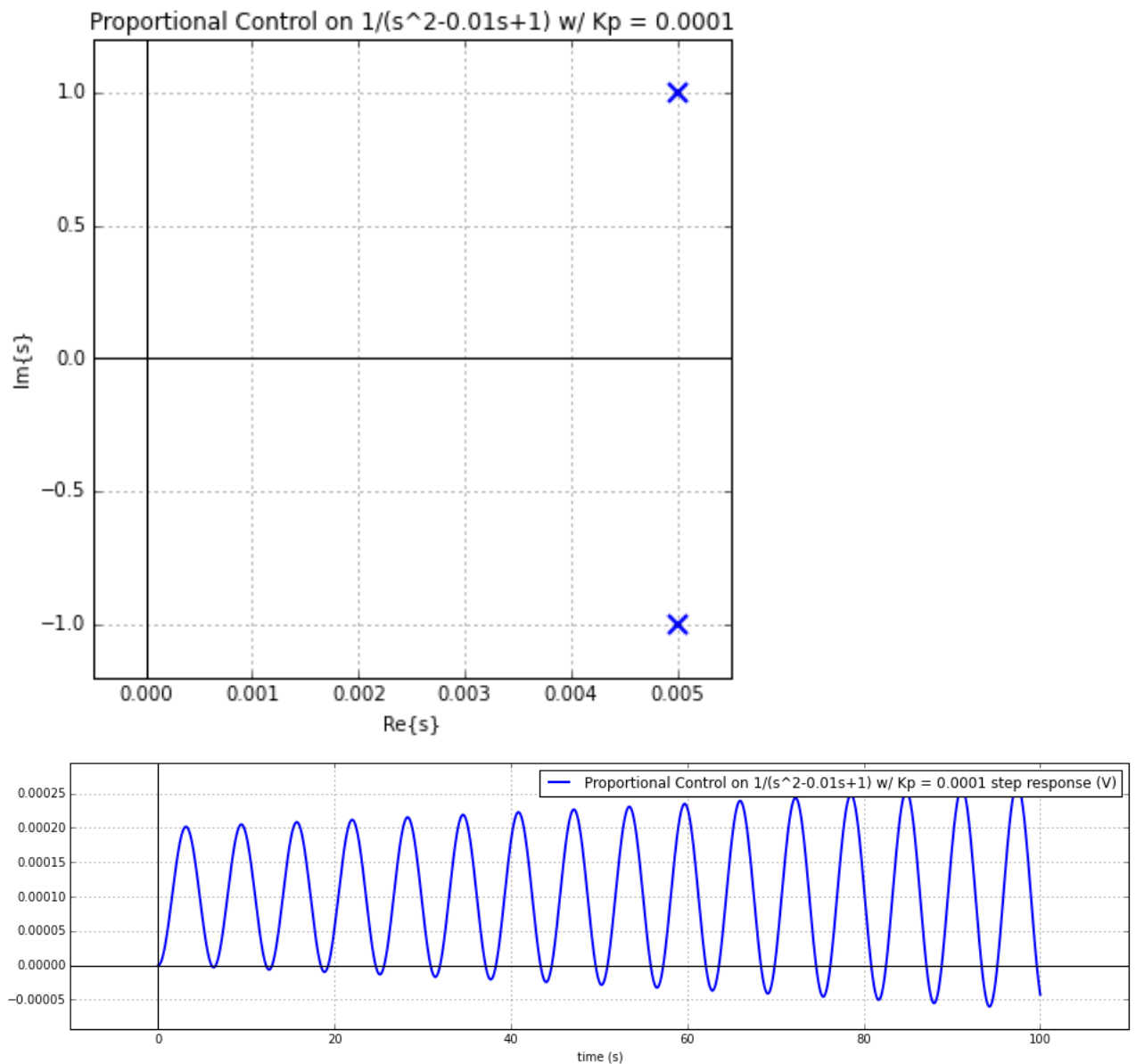
4A: The system has two imaginary poles to the right of the imaginary axis. The step response shows that the system is not stable, and the step response is a sin wave that grows larger and larger.

```
In [18]: #4B, using proportional control
title = 'Proportional Control on 1/(s^2-0.01s+1) w/ Kp = 0.3'
Kp = 0.3
lti_args = ([Kp],[1,-0.01,1+Kp])
system = get_system(lti_args)

plot_pole_zeros(title,system)
plot_step_response(title,system,t=np.linspace(0,100,5000))
```



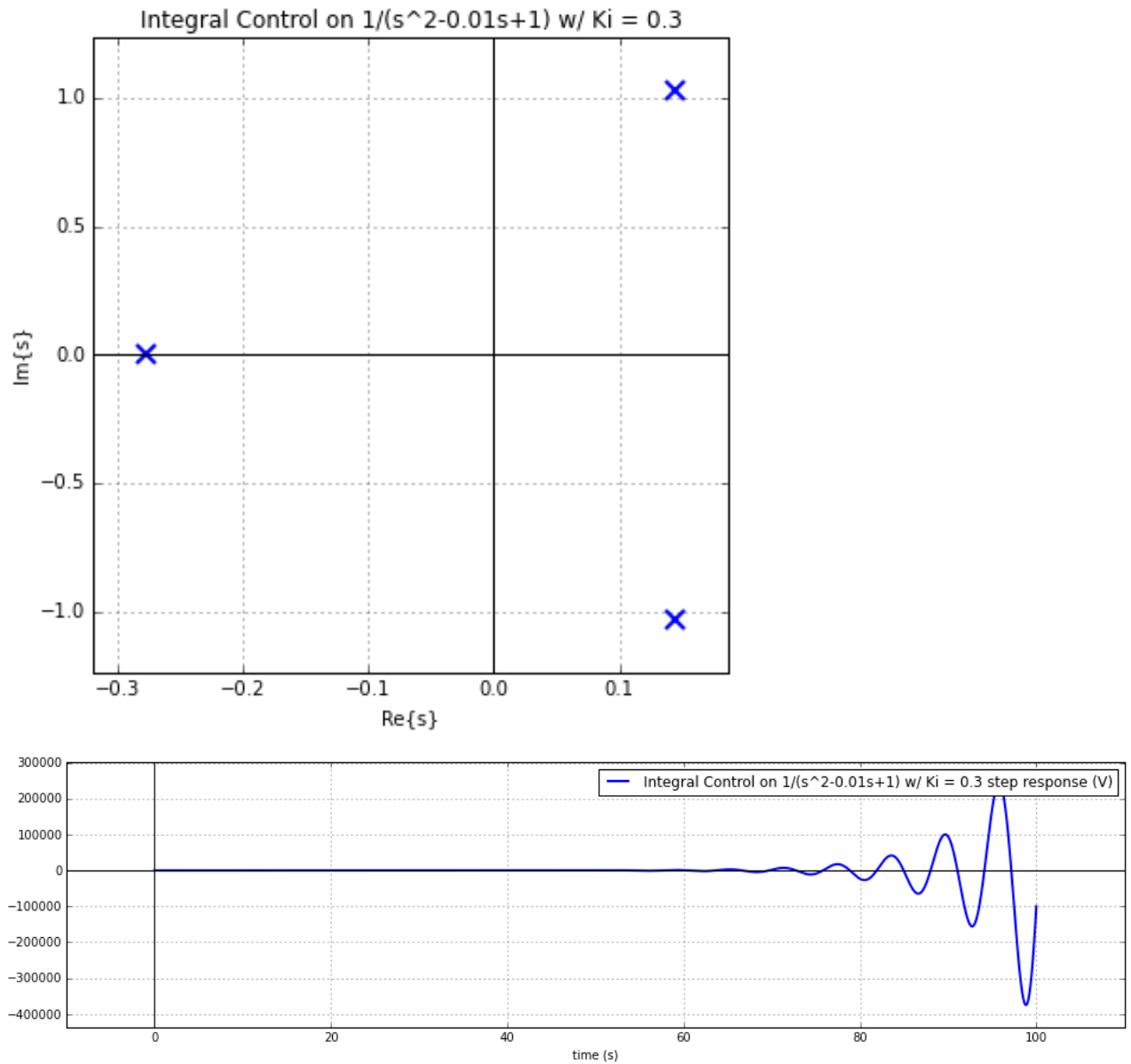
```
In [16]: #4B, using proportional control
title = 'Proportional Control on 1/(s^2-0.01s+1) w/ Kp = 0.0001'
Kp = 0.0001
lti_args = ([Kp],[1,-0.01,1+Kp])
system = get_system(lti_args)
plot_pole_zeros(title,system)
plot_step_response(title,system,t=np.linspace(0,100,5000))
```



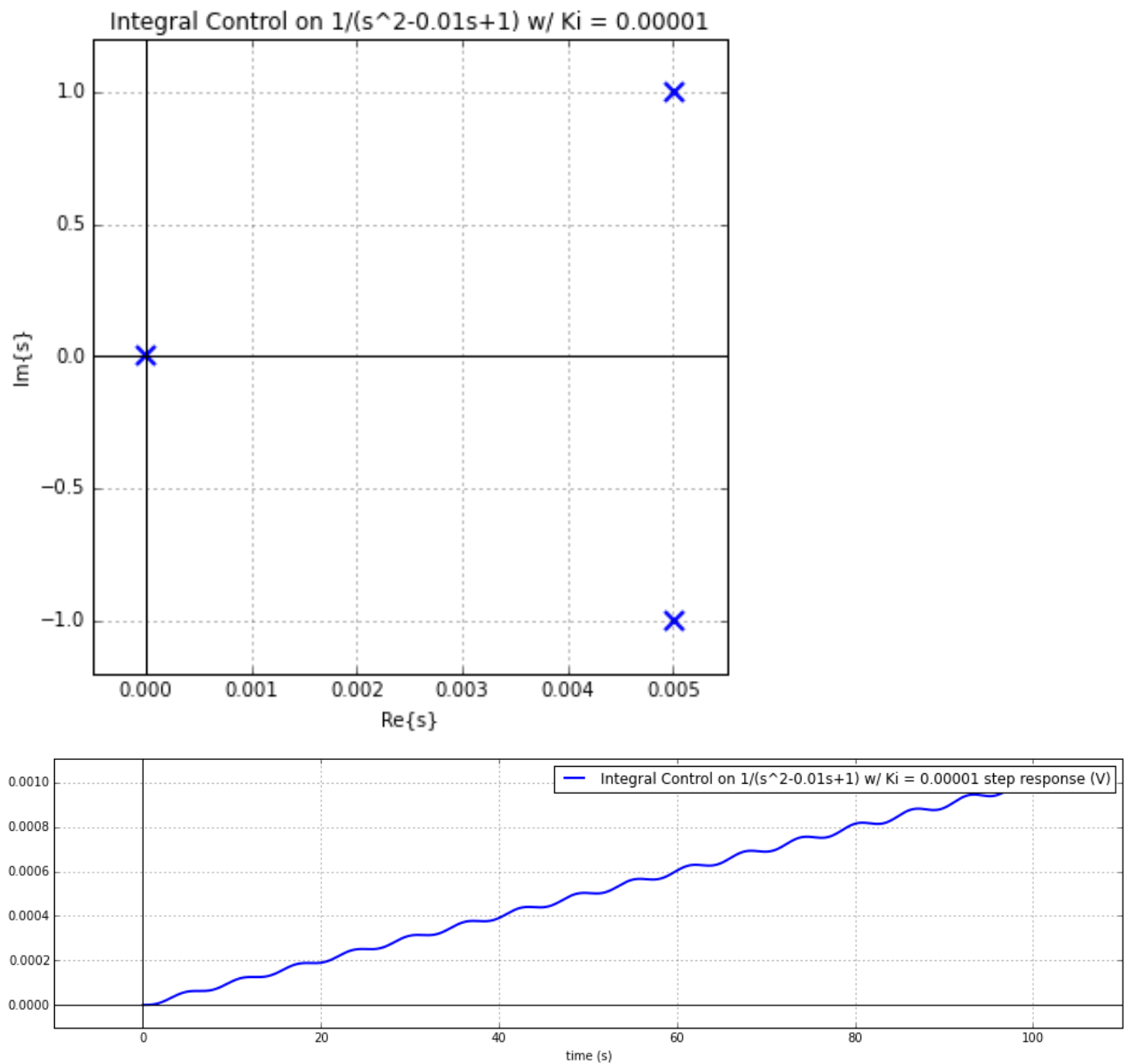
4B: The system can't be stabilized with proportional control, the oscillations grow larger and larger, tending towards infinity. Introducing proportional control just moved the poles along the imaginary axis.

```
In [19]: #4C, using integral control
title = 'Integral Control on 1/(s^2-0.01s+1) w/ Ki = 0.3'
Ki = 0.3
lti_args = ([Ki],[1,-0.01,1,Ki])

system = get_system(lti_args)
plot_pole_zeros(title,system)
plot_step_response(title,system,t=np.linspace(0,100,5000))
```

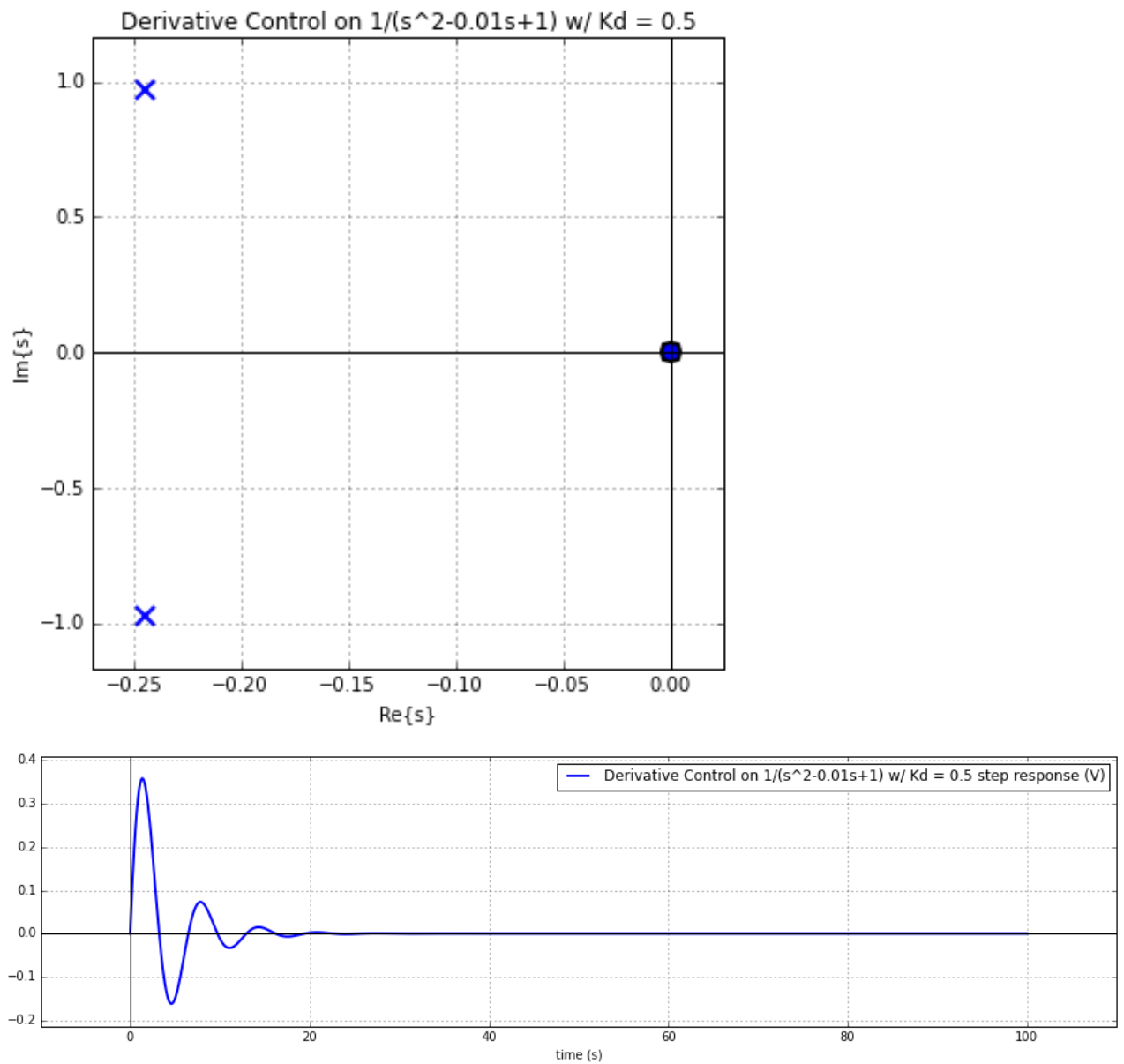



```
In [22]: #4C, using integral control
title = 'Integral Control on 1/(s^2-0.01s+1) w/ Ki = 0.00001'
Ki = 0.00001
lti_args = ([Ki],[1,-0.01,1,Ki])
system = get_system(lti_args)
plot_pole_zeros(title,system)
plot_step_response(title,system,t=np.linspace(0,100,5000))
```

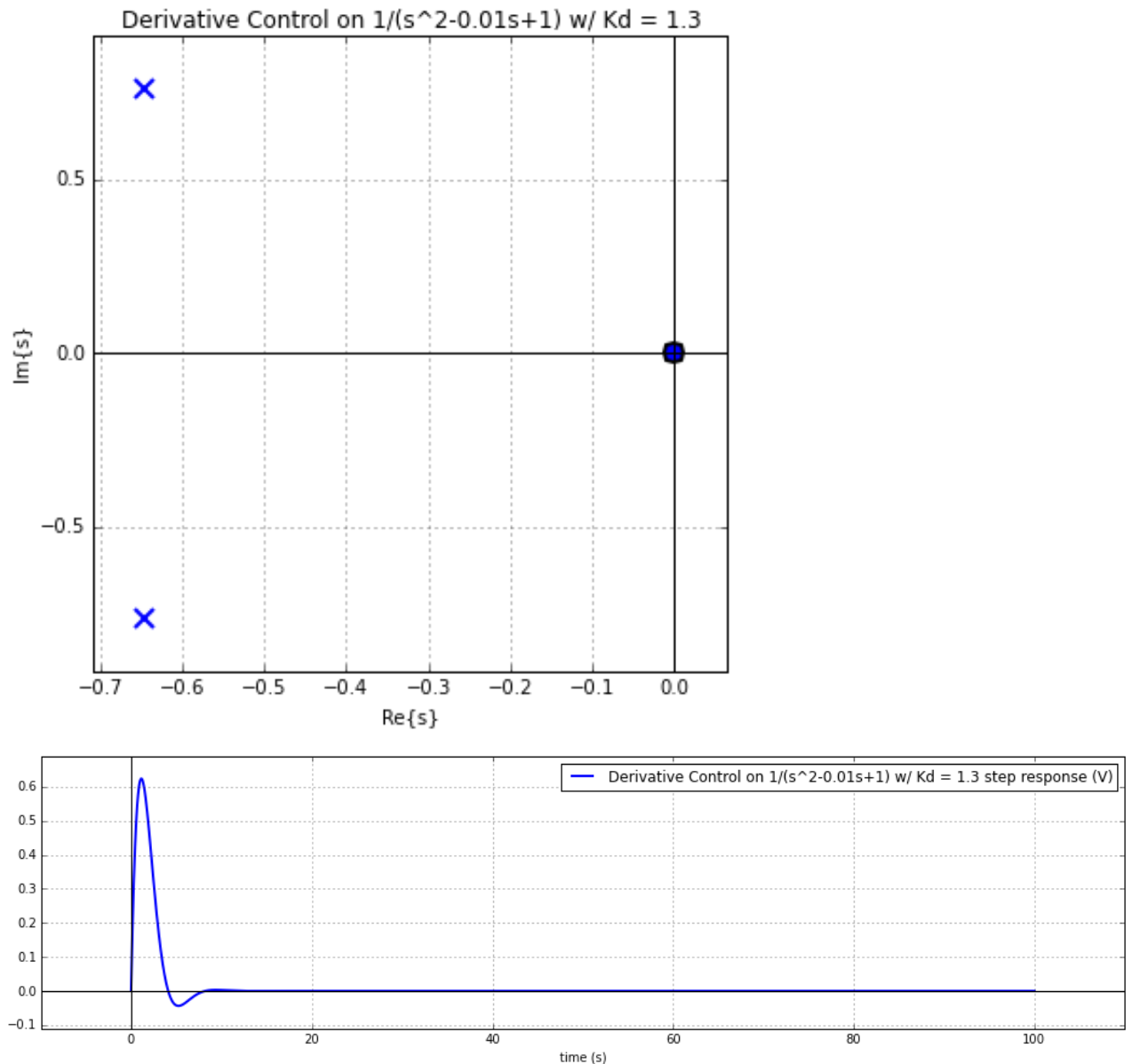


4C: Integral control can't do the job either! It introduces a third pole at zero on the imaginary and real axis, but that doesn't seem to do anything. Oscillations grow until they're huge, or the oscillation stays small but doesn't stay centered and slowly moves off of the set point.

```
In [23]: #4D, using derivative control
title = 'Derivative Control on 1/(s^2-0.01s+1) w/ Kd = 0.5'
Kd = 0.5
lti_args = ([Kd,0],[1,Kd-0.01,1])
system = get_system(lti_args)
plot_pole_zeros(title,system)
plot_step_response(title,system,t=np.linspace(0,100,5000))
```



```
In [24]: #4D, using derivative control
title = 'Derivative Control on 1/(s^2-0.01s+1) w/ Kd = 1.3'
Kd = 1.3
lti_args = ([Kd,0],[1,Kd-0.01,1])
system = get_system(lti_args)
plot_pole_zeros(title,system)
plot_step_response(title,system,t=np.linspace(0,100,5000))
```



4D: Derivative Control works! It introduces a zero, and moves the poles from the right of the imaginary axis to the left of it. This seems to stabilize the system.