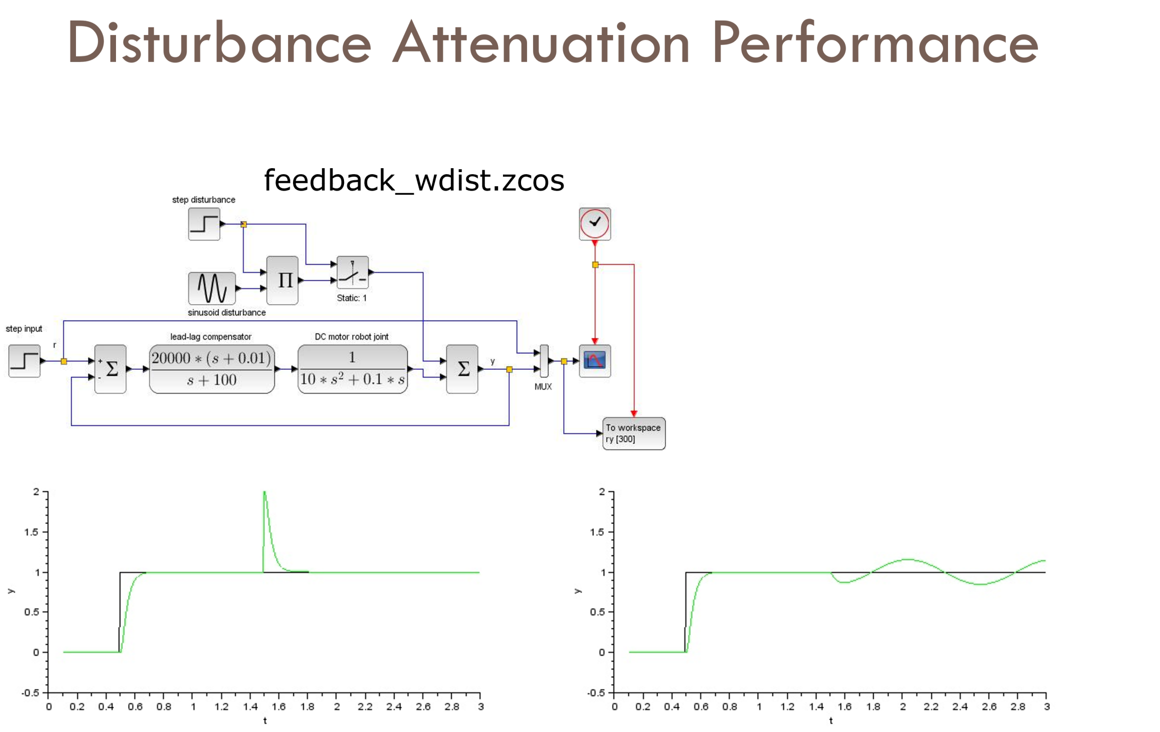
August 2021

Open in Colab

01211433 Homework 2

Figure 1 shows a Scilab/Xcos simulation diagram for output disturbance attenuation at the plant output



Response to step output disturbance   Response to sinusoidal output disturbance

Figure 1 Scilab/Xcos diagram for output disturbance attenuation

Create similar simulation setup in Python using the same plant and controller as in Figure 1; i.e.,

$$P(s) = \frac{1}{10s^2 + 0.1s}$$

$$C(s) = \frac{20000(s+0.01)}{(s+100)}$$

.

Since only disturbance attenuation performance is of interest, the step input can be discarded. The expected outputs are

1. For step input disturbance, the output should approach zero at steady state.
2. For low-frequency sinusoid disturbance, say, about 1 rad/s, output should be smaller in magnitude.

Remarks : For simplicity, the disturbance signal may start at t = 0.

Solution :

For colab user, install Python Control Library by uncommenting this cell

In [ ]:
```
#pip install control
```

Import the required packages

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import control as ctl
```

Create the plant and controller

In [2]:
```
# Your code
s = ctl.tf("s")
P = 1/(10*s**2 + 0.1*s)
C = 20000*(s+0.01)/(s+100)
```

Determine and contstruct the transfer function from output disturbance $d_o(s)$ to plant output $y(s)$

In [3]:
```
# Your code
L = C*P
S = 1/(1+L)
```

Use plot_response() function below for plotting, with arguments

- sys : transfer functtion mapping d_o to y
- u : a vector of disturbance signal d_o of 1. nonzero constant value 2. low frequency sinusoid wave
- t : time vector

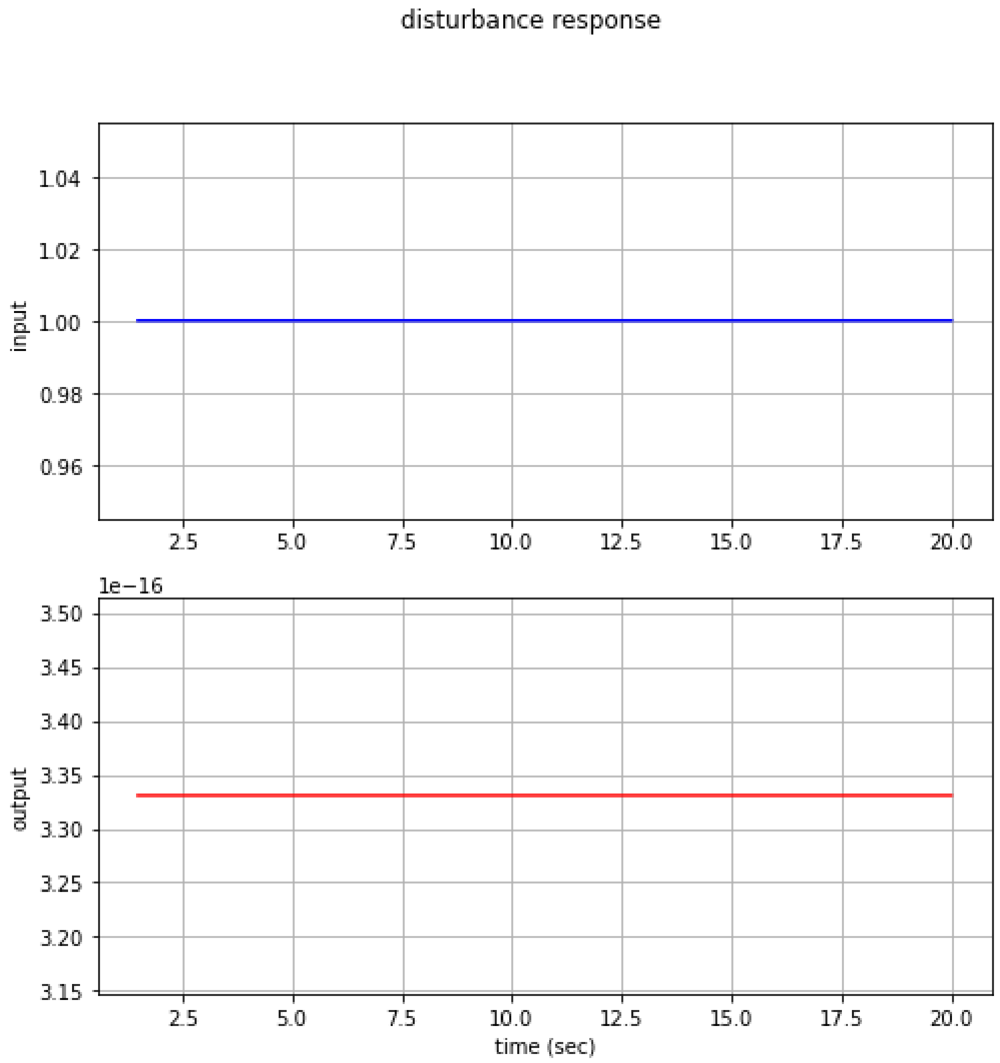
In [4]:
```
# Plotting function. No need to change anything.
def plot_response(sys,u,t,title="disturbance response"):
    tout, y = ctl.forced_response(sys,t, u)
    truncated_idx = 150  # get rid of transient
    tout = tout[truncated_idx:]
    u = u[truncated_idx:]
    y = y[truncated_idx:]
    fig, (ax1, ax2) = plt.subplots(2, figsize=(8,8))
    fig.suptitle(title)
    ax1.plot(tout,u,'b-')
    ax1.grid(True)
    ax1.set_ylabel('input')

    ax2.plot(tout,y,'r-')
    ax2.grid(True)
    ax2.set_ylabel('output')
    ax2.set_xlabel('time (sec)')

    plt.show()
```

1. Create time vector, and a constant disturbance input $d_o = c$

In [5]:
```
d_amplitude = 1  # can change amplitude of d_o
t = np.arange(0,20,0.01) # your code
u = d_amplitude*np.ones(t.shape) # your code
```

Plot step disturbance response

In [6]:
```
plot_response(S,u,t) # Replace ? with the right closed-loop transfer function
```

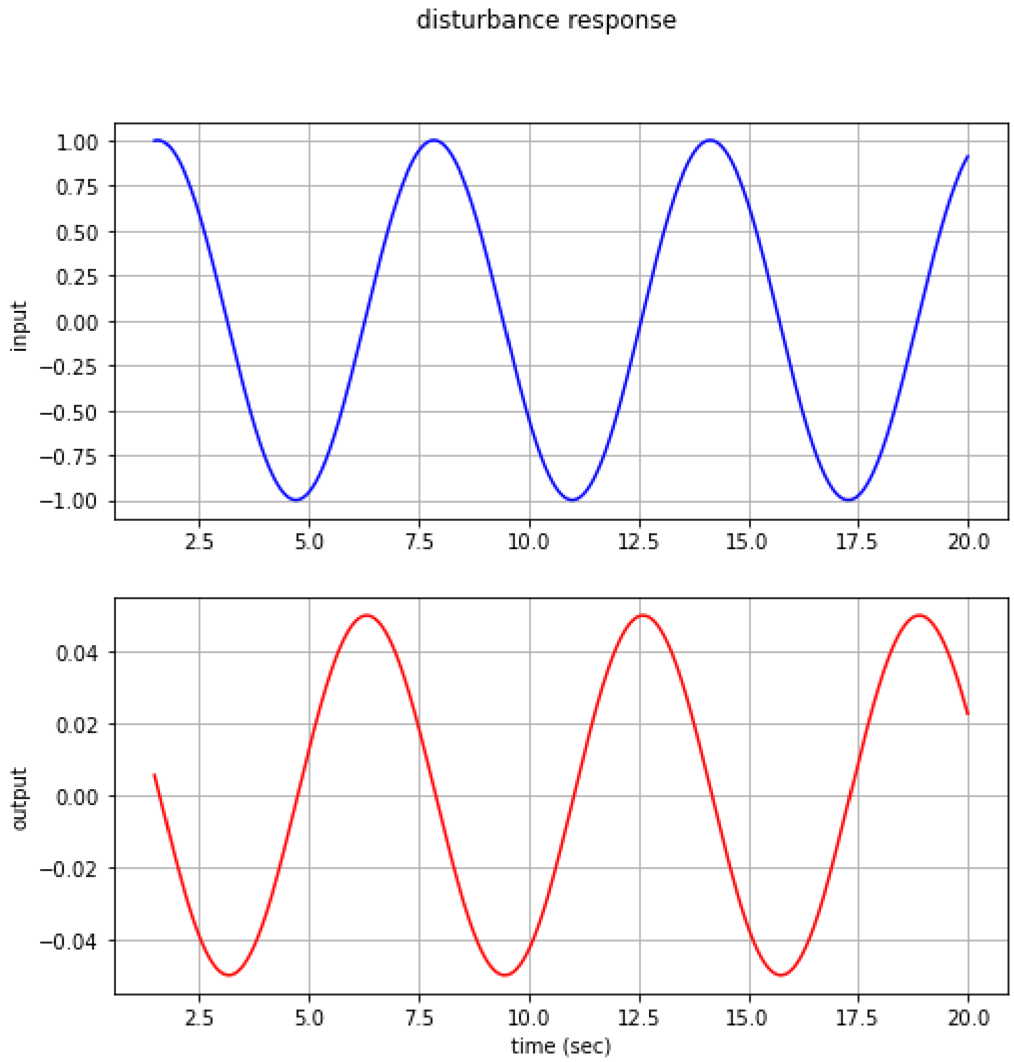


1. Use the time vector in 1. to construct a low-frequency sinusoid input vector (say, 1 rad/s)

In [7]:
```
w = 1 # rad/s
u = np.sin(w*t) # your code
```

Plot sinusoid disturbance response.

In [8]:
```
plot_response(S,u,t) # Replace ? with the right closed-loop transfer function
```


disturbance response
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
input
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
0.04
0.02
0.00
−0.02
−0.04
output
time (sec)

```
In [ ]:
```