ECEN 5638 Exercies - Control of time-delayed systems

Riley Kenyon

Gabe Rodriguez

04/01/2020

```
clear all; close all; clc;
s = tf('s');
```

Temperature Control

A model of the water temperature of a shower can be shown in statespace as

$$\dot{x} = -ax(t) + bu(t)$$

$$y(t) = x(t - \tau)$$

where coefficients and *a* and *b* are positive real scalars and $\tau > 0$ is a delay that encompasses the water traveling up to the shower head from the mixer. As a transfer function, this can be represented as

$$T(s) = \frac{be^{-\tau s}}{s + a},$$

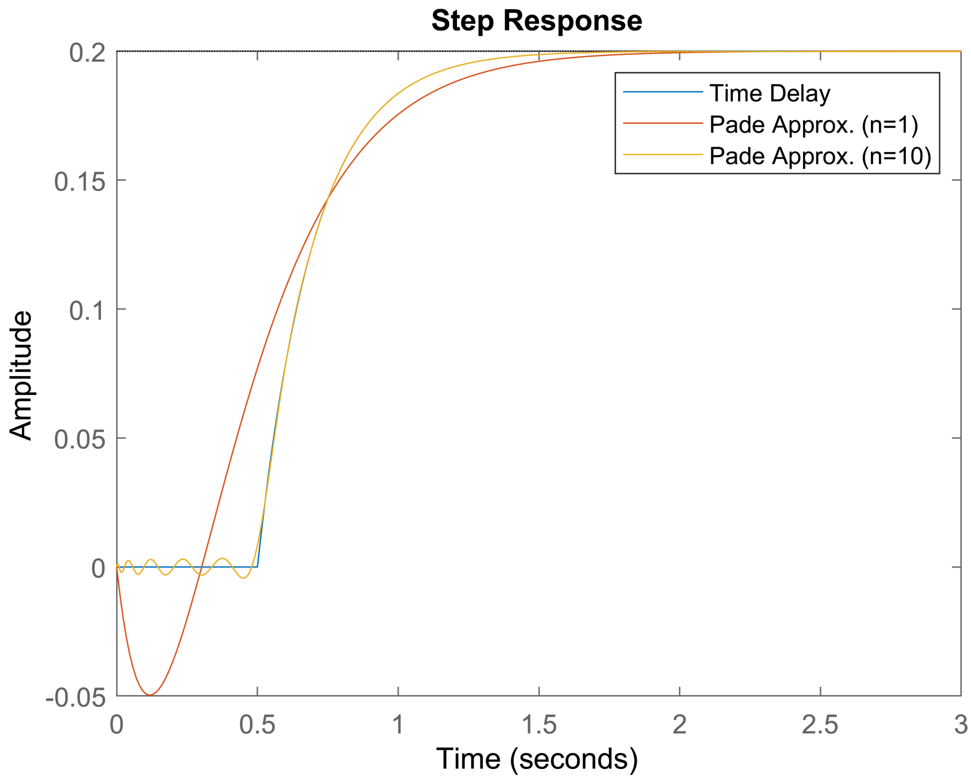
which is a non-linear transfer fucntion.

1. Given arbitrary positive values for *a*, *b*, and τ, compute the step response of the system.

```
b = 1;
a = 5;
tau = 0.5;
T = b*exp(-tau*s)/(s+a);
figure(1), step(T);
```

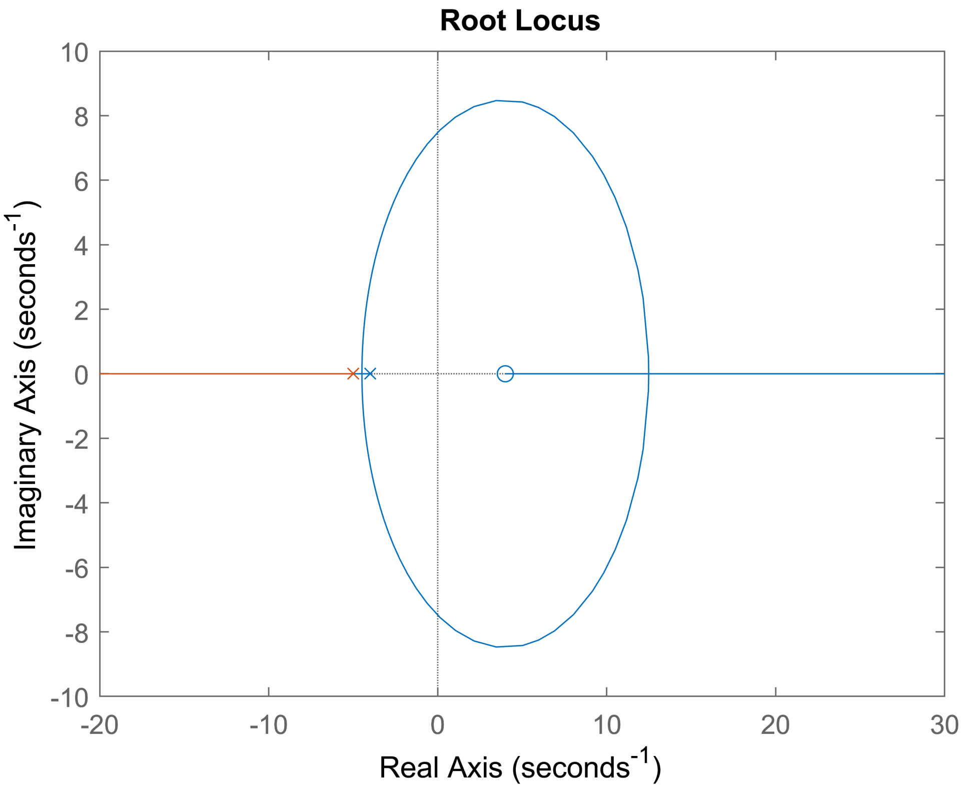
2. Use Tp = pade(T,n) to compute the Padé approximation of degree n of the transfer function $T(s)$. The approximation $T_p(s)$ will be the linear transfer function of order n which approximates the non-linear transfer function $T(s)$. Compare the step responses for n=1 and n=10

```
Tp_1 = pade(T,1);
Tp_10 = pade(T,10);
figure(1), hold on
h = stepplot(Tp_1,Tp_10);
p = getoptions(h);
p.XLim = {[0, 3]};
p.YLim = {[-0.05, 0.2]};
setoptions(h,p);
legend('Time Delay','Pade Approx. (n=1)','Pade Approx. (n=10)');
```



3. Using rltool, compute the feedback gain at which the closed loop system becomes unstable, how does the rlocus compare to the system without delay.

```
% Define system without delay
T_ndelay = b/(s+a);

% Root locus of system with/without delay
figure(2),
rlocus(Tp_1,T_ndelay);
```



Note that the system with delay will go unstable at a gain of **6.9**, while the sytem without delay is first order and remains stable for all gain.

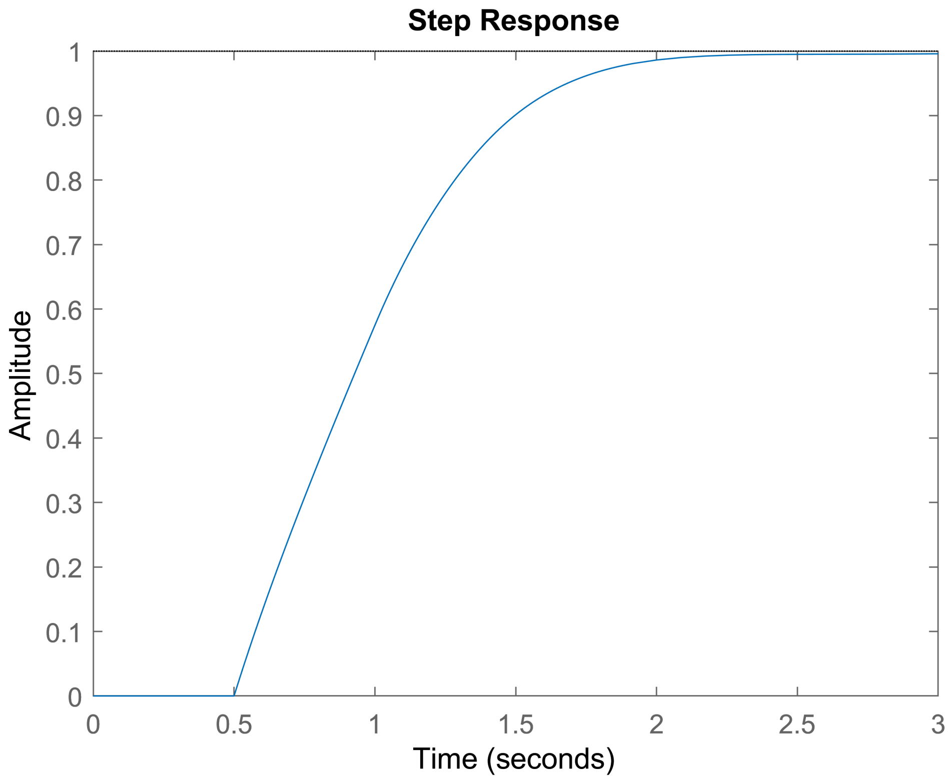
4. Design a feedback controller that achieves zero steady-state error for the approximated system $T_p(s)$ with n =1

```
% Design controller with rltool
load tempController.mat;
```

$$C(s) = \frac{1.4024(s + 3.575)}{s}$$

5. Compute the step response of the closed loop delayed system

```
T_CL = feedback(C*T,1);
figure(3),
step(T_CL);
```



Point of Social Distancing

Unstable system with delay such as the Coronavirus reproduction model

$$\dot{x}(t) = -ax(t) + bu(t)x(t)$$

$$y = cx(t - \tau)$$

where x is the total number of infectious people, *u* is the average interaction rate, and *y* is the number of people who tested positive. The scalars *a*, *b*, and *c* are positive scalars that represent the rate at which people overcome the disease, the probability of infection per interaction, and the percentage of people who get tested. Finally, τ is the incubation period. For the sake of simplicity, assume $c = 1$, $b = 10a$, and $\tau = 14$. Note that the system is nonlinear due to the product of $u * x$, in addition to the delay.

One option to control the system is to control the system around the current operating point by only making small variations. For this option

1. Linearize the System around the operating point $\bar{x}$ = 100 and $\bar{u}$ = 1. Compute the corresponding transfer function and use the Pade approximation (with n = 1) to obtain a linear transfer function.

$$\dot{x}(t) = -ax(t) + b\bar{u}\ \bar{x} + b\bar{u}(x - \bar{x}) + b\bar{x}(u - \bar{u})$$

$$y(t) = cx(t - \tau)$$

$$\mathscr{L}[\dot{x}(t)] = sX = (-a + b\bar{u})X + b\bar{x}\ U - b\bar{x}\ \bar{u}$$

$$(s + a - b\bar{u})X = b\bar{x}\ U - b\bar{x}\ \bar{u}$$

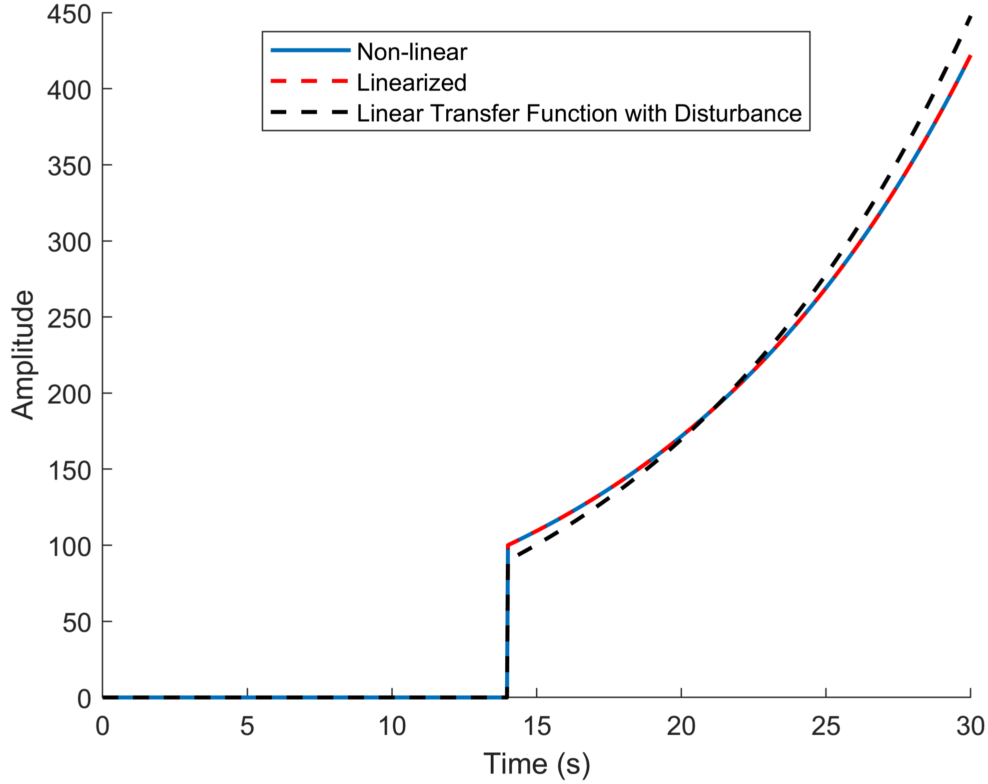
Assuming that the constant term can be treated as a disturbance and incorporating the delay, the equation can be simplified to the transfer function

$$\frac{Y}{U} = \frac{bc\bar{x}e^{-\tau s}}{s + (a - b\bar{u})}$$

```
% Delete workspace and define new problem
clear all; close all;
s = tf('s');
c = 1;
tau = 14;
u_bar = 1;
x_bar = 100;

% Choose a
a = 0.01;
b = 10*a;
x0 = 100;

% For simulink
funct = @(x,u,a,b) -a*x + b*u*x;
funct_lin = @(x,u,a) (-a+b*u_bar)*x + b*x_bar*u - b*x_bar*u_bar; % Linear approximation

% Linearized Transfer Function
T = b*c*x_bar*exp(-tau*s)/(s+(a-b*u_bar));
Tp_1 = pade(T,1);

% Disturbance to verify results
d = -b*c*x_bar*u_bar/(s+a-b*u_bar);

% Check step response
sim comparison.slx
```

```
Warning: The specified buffer for 'comparison/Transport Delay' was too small. During simulation, the buffer
 size was temporarily increased to 14336. In order to generate code, you need to update the buffer size
 parameter
```

```
figure(4),
hold on,
plot(time,y(:,1),'LineWidth',1.5)            % Non-linear
plot(time,y(:,2),'r--','LineWidth',1.5);     % Linearized
plot(time,y(:,3),'k--','LineWidth',1.5);     % Linearized TF w/ disturbance
legend('Non-linear','Linearized','Linear Transfer Function with Disturbance','location','best')
xlabel('Time (s)')
ylabel('Amplitude')
```

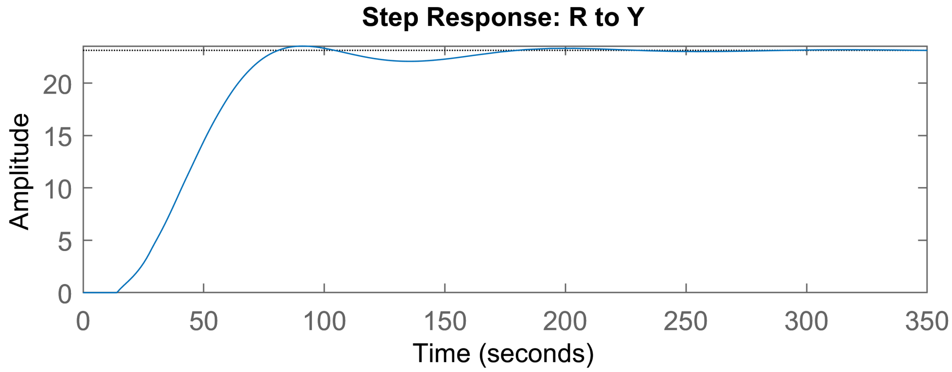


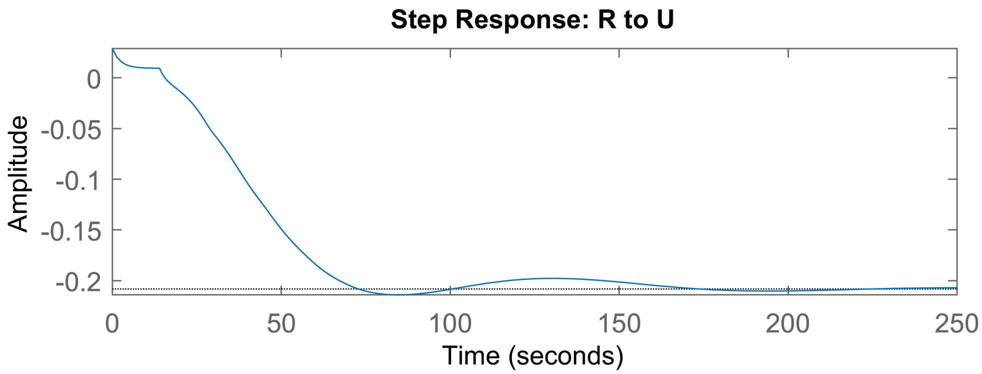
2. Design a stabilizing controller for the plant using rltool

```
load coronaController5.mat
```

$$C(s) = \frac{0.02898(s + 0.1356)}{s + 0.4178}$$

```
T_CL = feedback(C*T,1);
figure(5),
subplot(2,1,1)
step(T_CL);     % R to Y
title('Step Response: R to Y')
subplot(2,1,2)
step(C*(1-T_CL));   % R to U
title('Step Response: R to U')
```



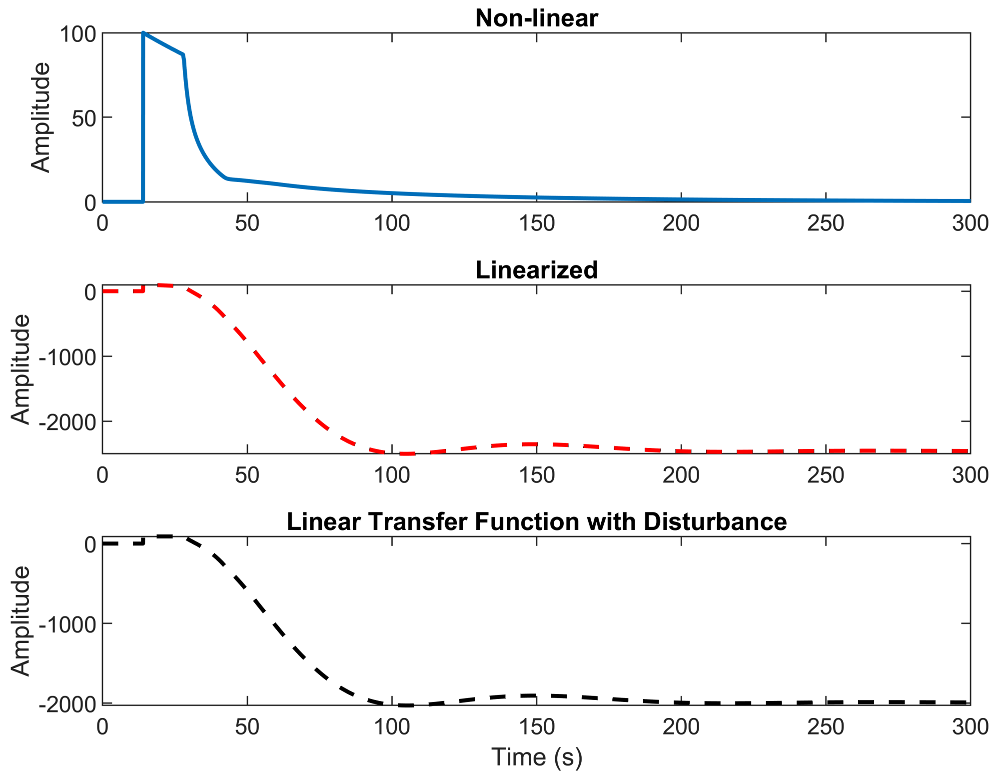


3. The step response of the system from R to Y has a DC gain of about 23 and has a risetime of 37 seconds. The behavior of R to U is is interesting because it does not have a large magnitude and reaches a steady state of -0.21 to maintain the constant Y output.

```
% Closed loop simulations
sim comparison_CL.slx
```

```
Warning: Output port 1 of 'comparison_CL/Step1' is not connected.
Warning: The specified buffer for 'comparison_CL/Transport Delay' was too small. During simulation, the
 buffer size was temporarily increased to 14336. In order to generate code, you need to update the buffer
 size parameter
```

```
figure(6),
subplot(3,1,1)
plot(time,y(:,1),'LineWidth',1.5)          % Non-linear
title('Non-linear')
ylabel('Amplitude')
subplot(3,1,2)
plot(time,y(:,2),'r--','LineWidth',1.5);    % Linearized
title('Linearized')
ylabel('Amplitude')
subplot(3,1,3)
plot(time,y(:,3),'k--','LineWidth',1.5);    % Linearized TF w/ disturbance
xlabel('Time (s)')
ylabel('Amplitude')
title('Linear Transfer Function with Disturbance')
```

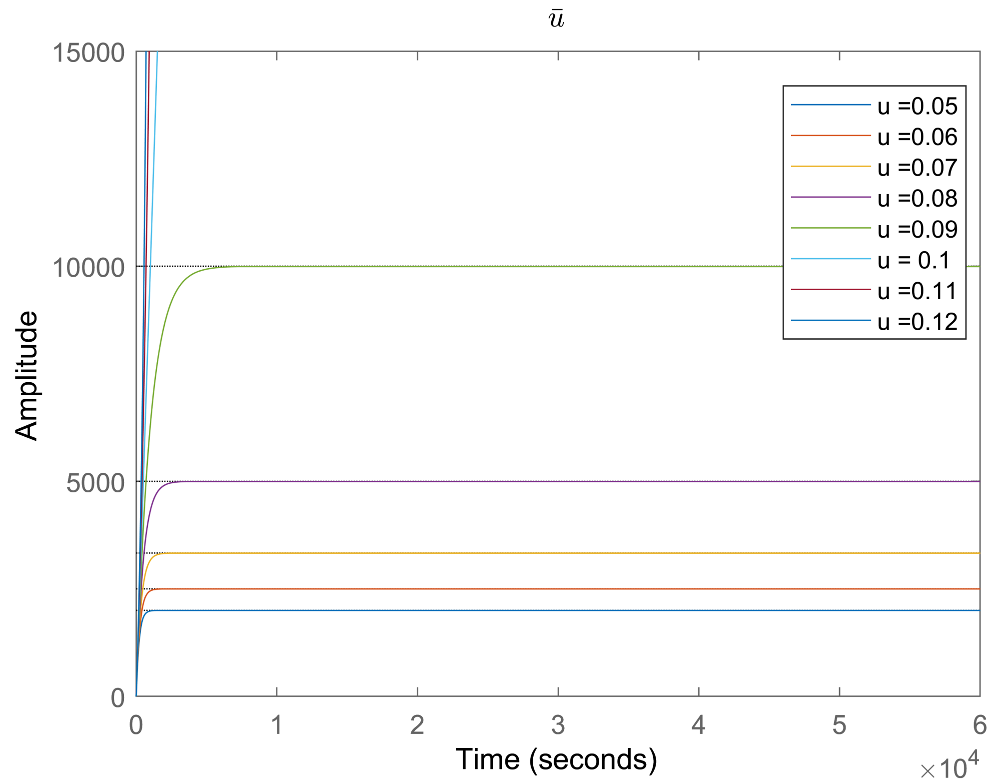


In a social setting it is impossible to implement the control law, the actuators would be people and their interactions - this can only really be controlled by state mandates to remain isolated. However, even then it is impossible to achieve a negative interaction rate as is required by the control law. The parameters for the model would be determined by the r_o estimate of the corona virus, data on the number of people getting tested with respect to the total population, and the estimate of the average recovery time as those infected recover.

Option #2 - modify the new operating point of the system

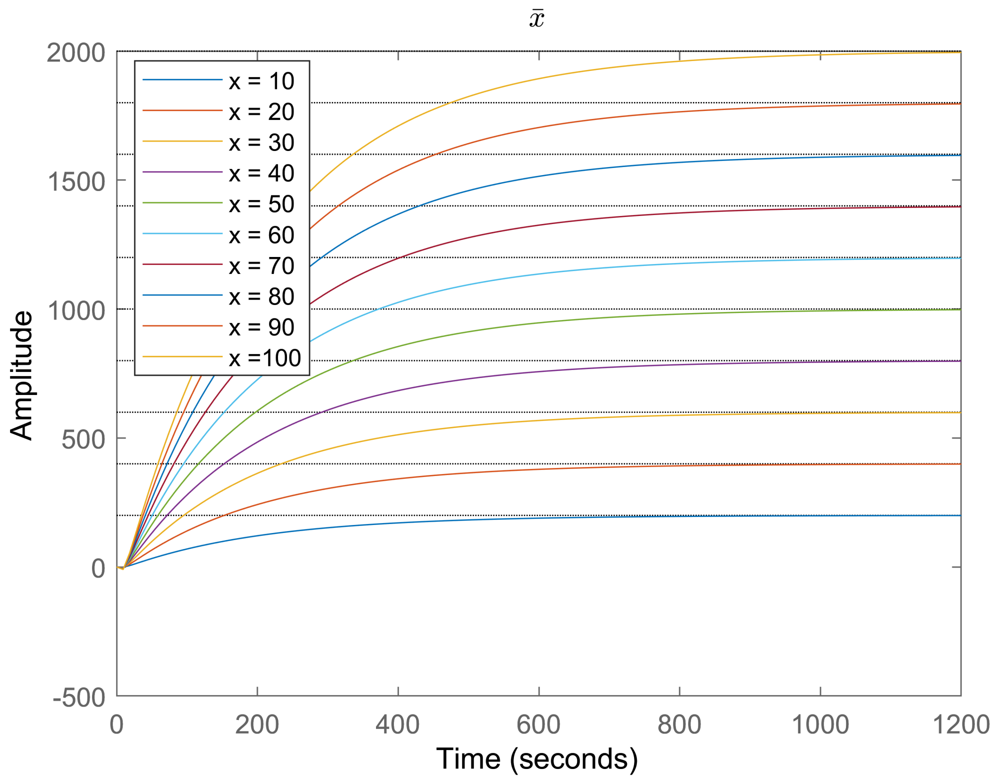
4. Based on the linearized model, determine a new point of operation $\bar{u}_n$ (prove that $\bar{x}$ does not matter) such that the linearized system is stable.

```
% Vary u_bar while keeping x_bar constant
x_bar = 100;
u_bar = [0.05:0.01:0.12];
figure(7)
hold on
for ii = 1:length(u_bar)
    T = b*c*x_bar*exp(-tau*s)/(s+(a-b*u_bar(ii)));
    Tp_1 = pade(T,1);
    step(Tp_1)
end
title("$\bar{u}$",'Interpreter',"latex")
legend(strcat('u =  ',num2str(u_bar')),'Location',"best")
ylim([0,15000])
```




```
Warning: Error updating Text.

 String scalar or character vector must have valid interpreter syntax:
$\bar{u}$
```

```
% Vary x_bar while keeping u_bar constant
u_bar = 0.05;
x_bar = [10:10:100];
figure(8)
hold on
for ii = 1:length(x_bar)
    T = b*c*x_bar(ii)*exp(-tau*s)/(s+(a-b*u_bar));
    Tp_1 = pade(T,1);
    step(Tp_1)
end
legend(strcat('x =  ',num2str(x_bar')),'Location',"best")
title("$\bar{x}$",'Interpreter',"latex")
```




```
Warning: Error updating Text.

 String scalar or character vector must have valid interpreter syntax:
$\bar{x}$
```

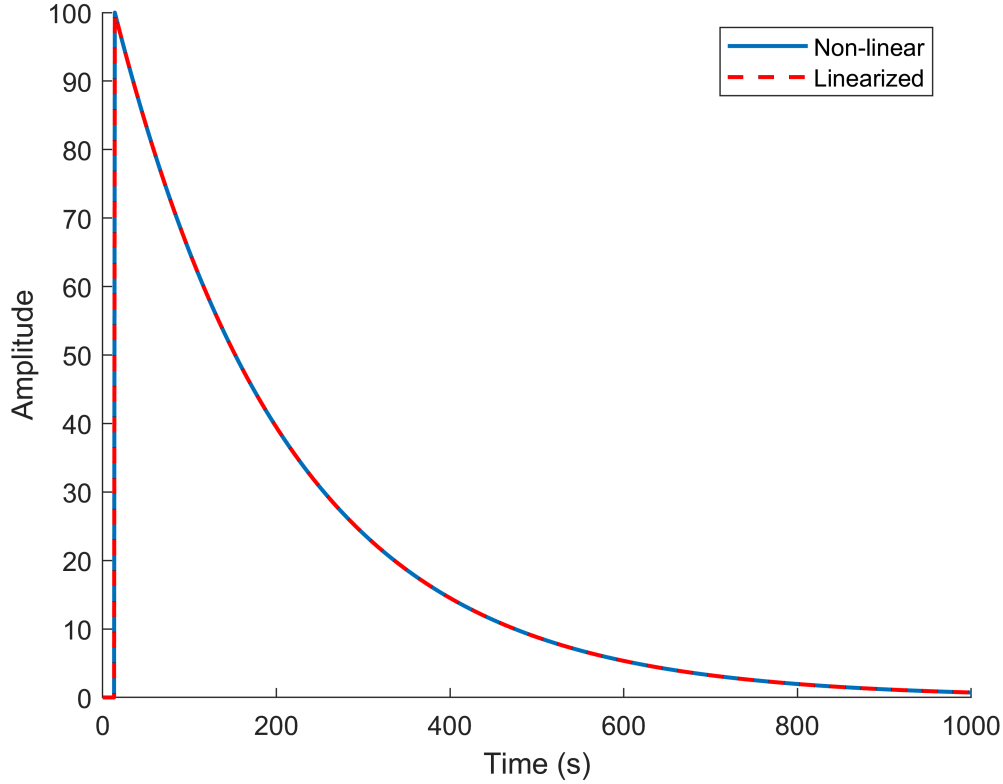
Note that at a stable $\bar{u}$, regardless of $\bar{x}$ the system will remain stable. Choose $\bar{u}$ to be 0.05, step response is stable and rise time was fastest of those tested with an $\bar{x}$ of 10.

5. Given any $u(t) < \bar{u}_n$, compute the behavior of the of the system given the starting condition $x(0) = 100$

```
x_bar = 10;                                     % To simulate for less time
u_bar = 0.05;
sim comparison_saturated.slx
```

```
Warning: The specified buffer for 'comparison_saturated/Transport Delay' was too small. During simulation,
 the buffer size was temporarily increased to 14336. In order to generate code, you need to update the buffer
 size parameter
```

```
figure(9)
hold on
plot(time,y(:,1),'LineWidth',1.5)               % Non-linear
plot(time,y(:,2),'r--','LineWidth',1.5);        % Linearized
legend('Non-linear','Linearized','Location',"best");
xlabel('Time (s)')
ylabel('Amplitude')
```



Conclusion here is that as long as $u(t)$ remains sufficiently small, the open loop system will stabilize naturally. In reality, this is implemented by limiting the average infection rate with social distancing.

Appendix

comparison.slx

comarison_CL.slx

comparison_saturated.slx