Linear Systems and Control - Week 8

Controller Design - Full State Feedback Controller - MATLAB Code

Motivation for Controller Design

A system is unstable if:

- Any/all eigenvalue(s) of matrix A is/are non-negative
- Any/all pole(s) of transfer function is/are non-negative
- Step response is unbounded

Motivation for Controller Design

A system is unstable if:

- Any/all eigenvalue(s) of matrix A is/are non-negative
- Any/all pole(s) of transfer function is/are non-negative
- Step response is unbounded

If a system is unstable, then what we can do to stabilize it?

Motivation for Controller Design

A system is unstable if:

- Any/all eigenvalue(s) of matrix A is/are non-negative
- Any/all pole(s) of transfer function is/are non-negative
- Step response is unbounded

If a system is unstable, then what we can do to stabilize it?

Solution:

- Check the pre-requisites of controller (if pre-requisites full-filled then goto next step)
- Design a suitable controller and
- Integrate/connect the controller with the system.

Types of Controller

There are 3 types of techniques to design controllers which are:

- Full-state feedback controller or state feedback controller
- Observer-based state feedback controller
- Proportional, Integral and Derivative (PID) controller

Types of Controller

There are 3 types of techniques to design controllers which are:

- Full-state feedback controller or state feedback controller
- Observer-based state feedback controller
- Proportional, Integral and Derivative (PID) controller

In today lecture, we will design and simulate full-state feedback controller.

Example that we did last time

Consider a system having the following state space model:

$$\begin{bmatrix} \frac{dx_1}{dt} \\ \frac{dx_2}{dt} \end{bmatrix} = \begin{bmatrix} 2 & 3 \\ 0 & 5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 1 \\ 2 \end{bmatrix} u(t)$$

$$y = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

Check the following:

- Do we need a controller?
- If we need a controller, identify which controller to design
- Design that controller and place the eigenvalues at $(-3, -5)$.

MATLAB and Simulink Code division

There are four main segments in programming/coding using MATLAB and Simulink

1. Declare the variables and matrices using MATLAB
2. Check the stability using MATLAB (i.e. the answer to question: do we need a controller)
3. Design the controller using MATLAB
4. Simulate the system with controller connected to the system using Simulink

Segment 1 - Declaring the necessary variables and matrices

Let us declare the matrices in MATLAB

```
% Sample Code
clear;
clc;
close all;

A=[2 3; 0 5];
B=[1; 2];
C=[1 0; 0 1];
D=[0;0];

% Please ensure that matrix D has same columns as matrix B
% and same rows as matrix C
```

Figure: Variables and Matrices initialization in MATLAB

Segment 2 - Checking the stability of the system

```
% We check stability here
% There are 5/6 ways (3 we studied till now)
% Step response, eigen values, poles
% root-locus, nyquist, routh-hurwitz


% % Task 1 - Check stability of the system

% Method 1 - poles of tf
[n,d]=ss2tf(A,B,C,D);
poles_of_transfer_ftn=roots(d);
disp('The poles of the transfer function are ');
poles_of_transfer_ftn
```

Figure: Poles computation for stability information

Segment 2 - Checking the stability of the system

```
% Method 2
eigen_values=eig(A);
disp('The eigenvalues of matrix A are');
eigen_values
```

```
% Method 3
% Step response
step(A,B,C,D)
title('Step response of the system')
ylabel('Amplitude in response to unit step');
```

Figure: Eigenvalues and step response plot

Segment 2 - Checking the stability of the system

```matlab
% Method 4 - root locus
figure;
rlocus(A,B,[1 0],0)
title('Root locus of the system for first output')

figure;
rlocus(A,B,[0 1],0)
title('Root locus of the system for second output')
```

Figure: Root locus plot - you have covered this is lab

Pre-requisite check before controller design

```
% Task 2 Check controlability
P=ctrb(A,B);
rank_of_ctrb_matrix=rank(P);
disp('The rank of controllabilty matrix
rank_of_ctrb_matrix


order_of_system=size(A,1);
disp('The order of the system is')
order_of_system
```

Figure: Controllability matrix computation and rank check

Full-state static feedback controller design

```
% Design controller
desired_egnvalues=[-3 -5];
K=place(A,B,desired_egnvalues);
```

Figure: Full-state static feedback controller design

Simulation using MATLAB

```
% TASK 3
% we will do this simulation in SIMULINK
% Now let us see the step response
A_clp=A-B*K;
B_clp=B;
C_clp=C;
D_clp=D;

figure;
step(A_clp,B_clp,C_clp,D_clp);
title('Step response of close loop system');


% Let us compute the final value of the step response
final_value_of_close_loop_system=dcgain(A_clp,B_clp,C_clp,D_c
disp('The DC-gain of the system OR the final value should be'
final_value_of_close_loop_system
```

Construct the system using Simulink

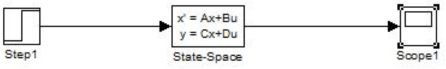


Figure: Construction of the system using Simulink

Construct the system using Simulink

Double-click the state-space block and make these changes

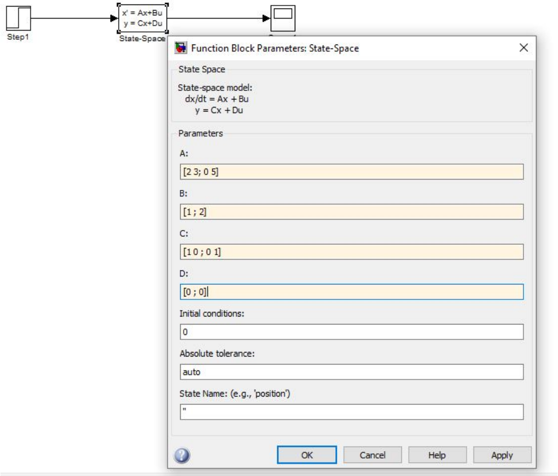
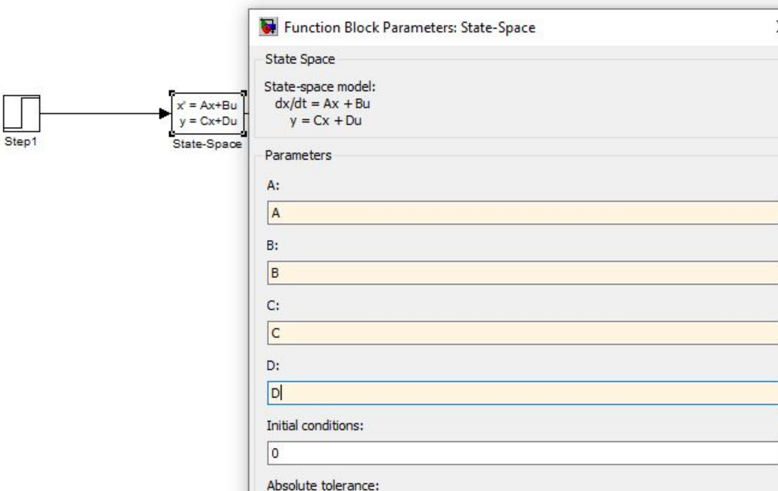


Figure: State-space block initialization to make Simulink understand our

Construct the system using Simulink

An easier way (provided these variables are declared in MATLAB work-space)



Trying to run the simulation using Simulink

Running the simulation from this button - You can increase or decrease the time of simulation

Figure: Press this button to run the simulation

Step response directly in Simulink

Step response of unstable system - its unbounded (going towards infinity)

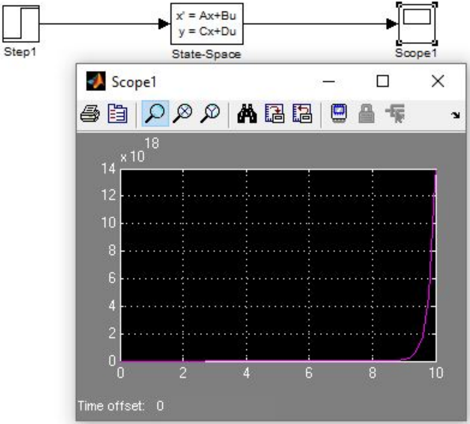


Figure: Step response of unstable system

Export of step response from Simulink to MATLAB

Now we need to export this data to MATLAB (for better plotting)

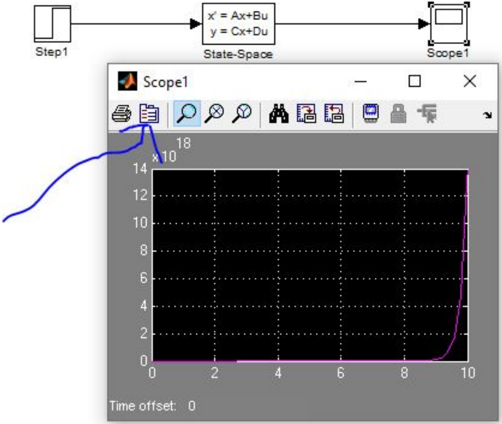


Figure: Step response of unstable system

Export of step response from Simulink to MATLAB

Now we need to export this data to MATLAB (for better plotting)

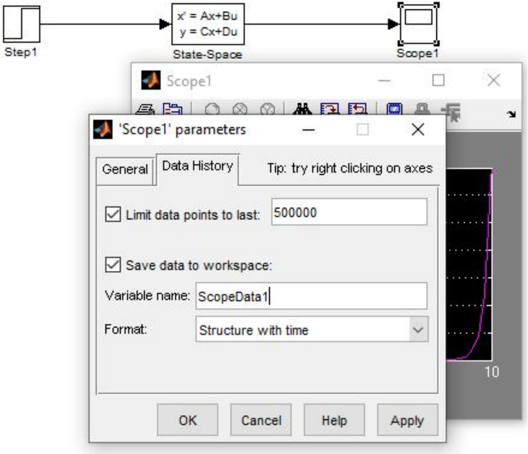


Figure: Step response of unstable system

Plot using MATLAB

Now we need to plot the exported data using MATLAB

```
>> plot(ScopeData1.time, ScopeData1.signals.values(:,1))
```

Figure: MATLAB code for plotting the first step responses

Plot using MATLAB

Now we need to plot the exported data using MATLAB

```
>> plot(ScopeData1.time, ScopeData1.signals.values(:,1))
>> figure
>> plot(ScopeData1.time, ScopeData1.signals.values(:,2))
>>
```

Figure: MATLAB code for plotting both the step responses

Simulation of augmented system using Simulink

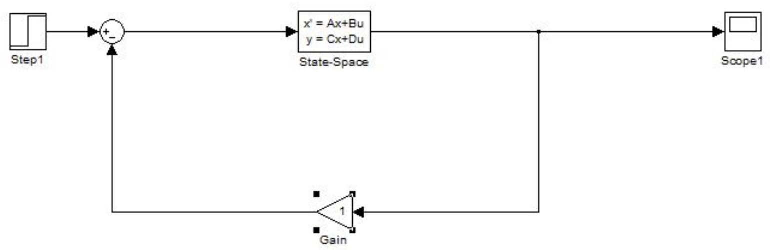


Figure: Sketch of augmented / close loop system

Simulation using Simulink

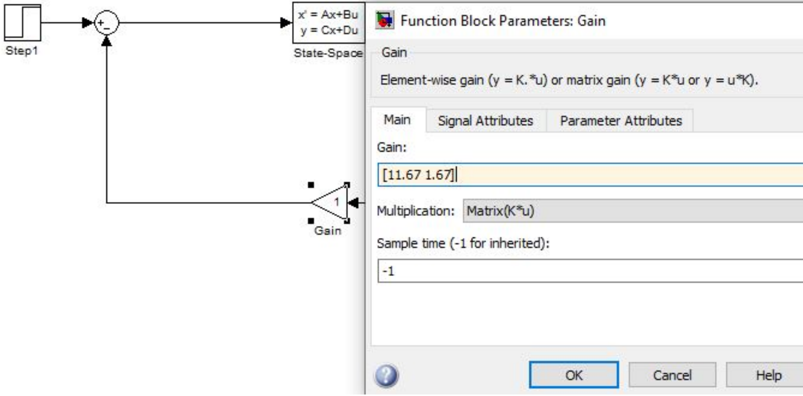


Figure: Sketch of augmented / close loop system

Simulation using Simulink

Now run the simulation and obtain plots of stable system. Check if the steady-state value is the same as the one obtained using MATLAB code