Contents

- Parameters
- Part 2 - Linearize Model about Operating Point x = 0, u = 0
- Part 3 Stabilizing Feedback Controller for Linearized Model, u = -Kx
- Part 4 Feedback Controller on Nonlinear System
- Part 5 State Observer Design with Linear System
- Part 6 Closed Loop Transfer Function
- Part 7 State Observer with Non-Linear System
- Functions


```
clear
close all
clc
```

Parameters

```
m       = 1;            % [kg]
l       = 1;            % [m]
g       = 10;           % [m/s^2]
M       = 10;           % [kg]
x_star  = zeros(4,1);   % State Operating Point
u_star  = 0;            % Input Operating Point
dt      = .005;         % Time Step
time    = (0:dt:5)';    % Time
```

Part 2 - Linearize Model about Operating Point x = 0, u = 0

```
syms x1 x2 x3 x4 u s real

% Create Symbolic Set of Equations
x       = [x1;x2;x3;x4];
xdot    = NonlinearCartPendulum([],x,u,m,M,g,l);

% Symbolic Jacobian Matrices
A_sym    = [diff(xdot(1),x1), diff(xdot(1),x2), diff(xdot(1),x3), diff(xdot(1),x4);...
            diff(xdot(2),x1), diff(xdot(2),x2), diff(xdot(2),x3), diff(xdot(2),x4);...
            diff(xdot(3),x1), diff(xdot(3),x2), diff(xdot(3),x3), diff(xdot(3),x4);...
            diff(xdot(4),x1), diff(xdot(4),x2), diff(xdot(4),x3), diff(xdot(4),x4)];

B_sym   = [diff(xdot(1),u);diff(xdot(2),u);diff(xdot(3),u);diff(xdot(4),u)];

disp('--Part 2 ---------------------------------------------------------')
disp(' ')
disp('Linearized State Space Model')

% Linearized State Space Model
A       = double(subs(A_sym,[x1;x2;x3;x4;u],[x_star;u_star]))
B       = double(subs(B_sym,[x1;x2;x3;x4;u],[x_star;u_star]))
C       = [1 0 0 0;0 0 1 0]    % y = [x1;x3]
D       = zeros(2,1)

% Verify System is Controllable and Observable
CO      = ctrb(A,B);
Obs     = obsv(A,C);

if rank(CO) == length(A)
```

```
    disp('System is Controllable');
end

if rank(Obs) == length(A)
    disp('System is Observable');
end
```

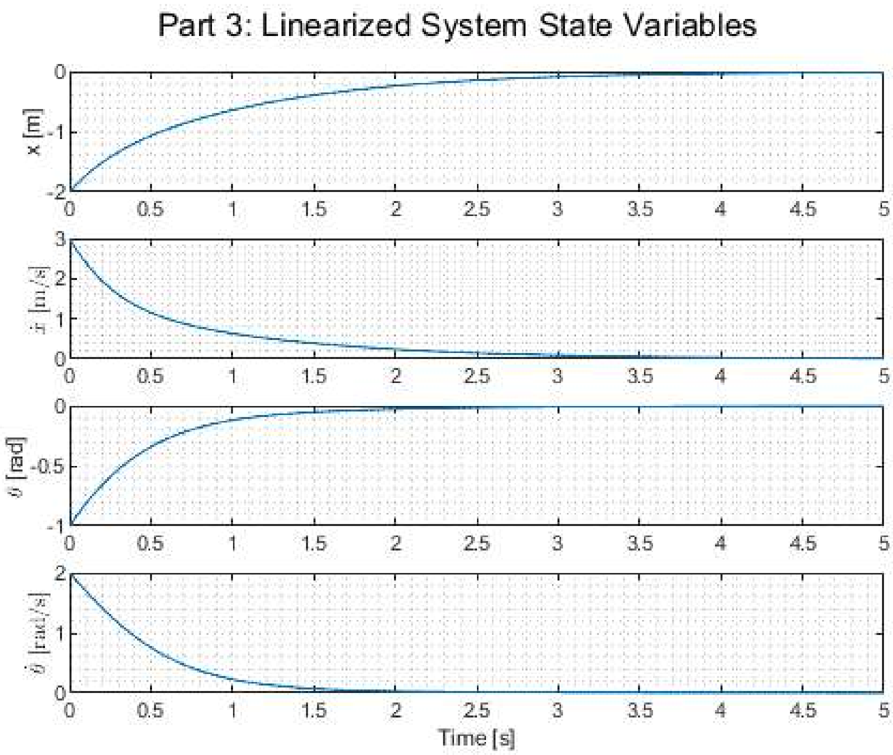
Part 3 Stabilizing Feedback Controller for Linearized Model, u = -Kx

del_xdot = A*del_x + B*del_u, del_u = -K*del_x, u = u_star + del_u, x = x_star + del_x, u = u_star - K*del_x => u = -K*x,

```
disp('--Part 3 --------------------------------------------------------')
disp(' ')

% desired poles
s_desired   = [-1;-4;-4 + 1i;-4 - 1i];

% Gain Matrix via ackerman formula
K           = acker(A,B,s_desired)

disp('Closed Loop Poles')
disp(eig(A - B*K))

% Simulate Controlled Linear System
x0          = [-2; 3;-1;2];     % Initial Conditions [m, ms/, rad, rad/s]

% ODE45 solver options
options     = odeset('AbsTol',1e-8,'RelTol',1e-8);

% ODE45 Function call
[T, X_lin]= ode45(@(t,x) ControlledLinearCartPendulum(t,x,A,B,K),time,x0,options);

figure
subplot(4,1,1)
sgtitle('Part 3: Linearized System State Variables')
plot(T,X_lin(:,1))
ylabel('x [m]')
grid minor
subplot(4,1,2)
plot(T,X_lin(:,2))
ylabel('$\dot{x}$ [m/s]','Interpreter','latex')
grid minor
subplot(4,1,3)
plot(T,X_lin(:,3))
grid minor
ylabel('\theta [rad]')
subplot(4,1,4)
plot(T,X_lin(:,4))
grid minor
ylabel('$\dot{\theta}$ [rad/s]','Interpreter','latex')
xlabel('Time [s]')
```

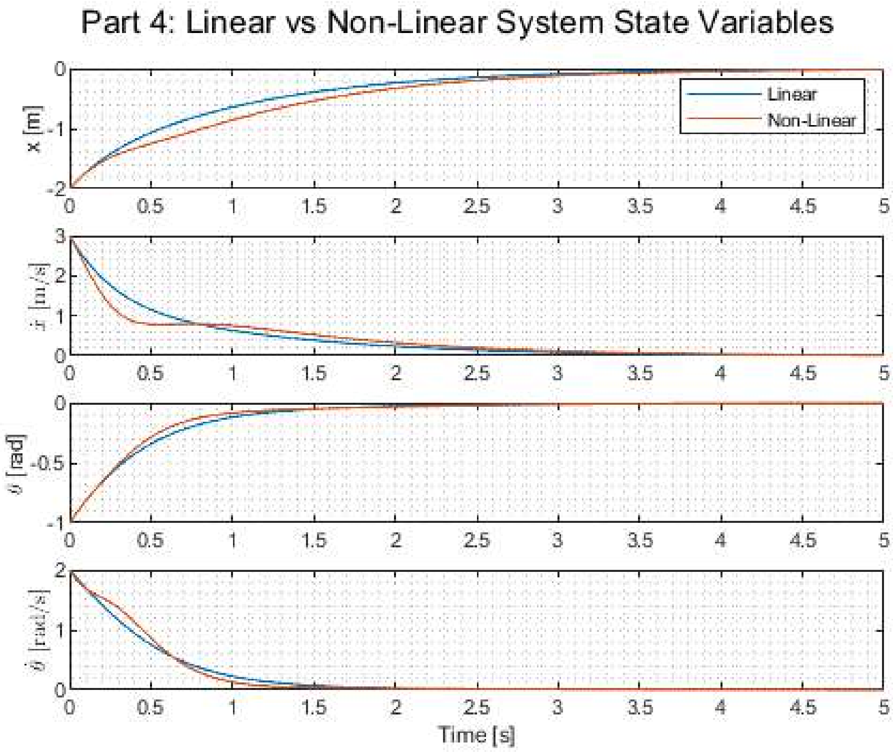
```
--Part 3 --------------------------------------------------------


K =

   -68   -117   -788   -247

Closed Loop Poles
  -4.0000 + 1.0000i
  -4.0000 - 1.0000i
```

```
    -4.0000 + 0.0000i
    -1.0000 + 0.0000i
```



Part 4 Feedback Controller on Nonlinear System

```
% ODE45 Function call
[T, X]      = ode45(@(t,x) ControlledCartPendulum(t,x,K,m,M,g,l),time,x0,options);

figure
subplot(4,1,1)
sgtitle('Part 4: Linear vs Non-Linear System State Variables')
plot(T,X_lin(:,1),T,X(:,1))
legend('Linear','Non-Linear')
ylabel('x [m]')
grid minor
subplot(4,1,2)
plot(T,X_lin(:,2),T,X(:,2))
ylabel('$\dot{x}$ [m/s]','Interpreter','latex')
grid minor
subplot(4,1,3)
plot(T,X_lin(:,3),T,X(:,3))
grid minor
ylabel('\theta [rad]')
subplot(4,1,4)
plot(T,X_lin(:,4),T,X(:,4))
grid minor
ylabel('$\dot{\theta}$ [rad/s]','Interpreter','latex')
xlabel('Time [s]')
```

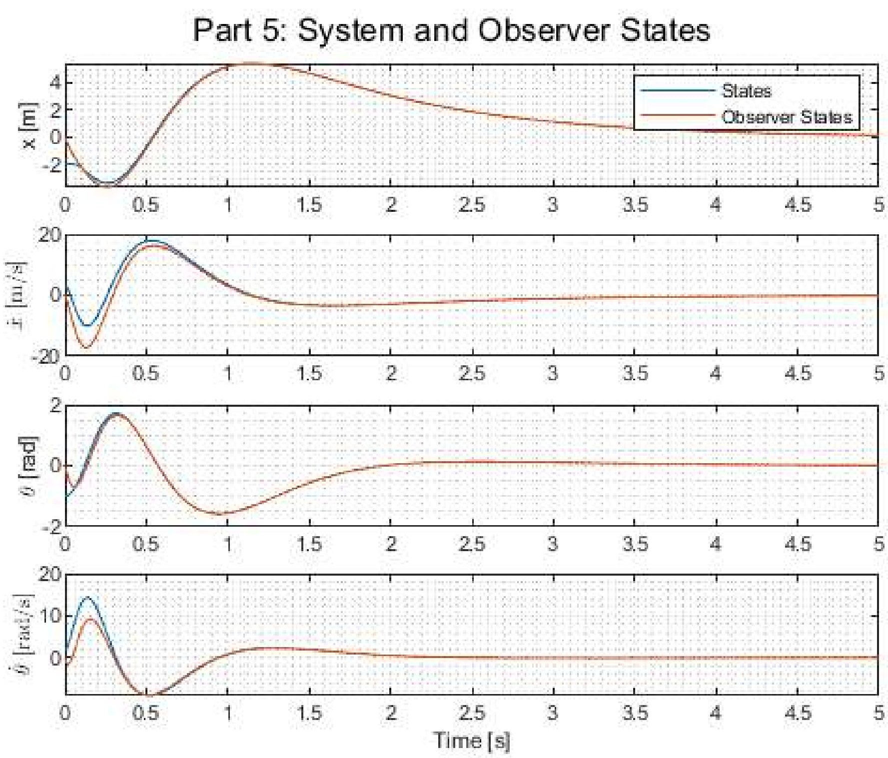


Part 5 State Observer Design with Linear System

```
disp('--Part 5 ---------------------------------------------------------')
disp(' ')

% desired observer poles
s_obs_desired   = [-3;-12;-12 + 1i;-12 - 1i];

% Observer Gain Matrix
L               = place(A',C',s_obs_desired)'

disp('Observer Poles')
disp(eig(A - L*C))

% Observer Initial Conditions
z0              = zeros(4,1);

% ODE45 Function call
[T, X_lin_obs]  = ode45(@(t,x) ControllerEstimatorLinearCartPendulum(t,x,A,B,C,K,L),time,[x0;z0],options);

figure
subplot(4,1,1)
sgtitle('Part 5: System and Observer States')
plot(T,X_lin_obs(:,1),T,X_lin_obs(:,5))
legend('States','Observer States')
ylabel('x [m]')
grid minor
subplot(4,1,2)
plot(T,X_lin_obs(:,2),T,X_lin_obs(:,6))
ylabel('$\dot{x}$ [m/s]','Interpreter','latex')
grid minor
subplot(4,1,3)
plot(T,X_lin_obs(:,3),T,X_lin_obs(:,7))
grid minor
ylabel('\theta [rad]')
subplot(4,1,4)
plot(T,X_lin_obs(:,4),T,X_lin_obs(:,8))
```

```
grid minor
ylabel('$\dot{\theta}$ [rad/s]','Interpreter','latex')
xlabel('Time [s]')
```

```
--Part 5 --------------------------------------------------------



L =

   15.4624    3.7298
   43.7948   43.0396
    0.6685   23.5376
   16.4663  146.7505

Observer Poles
  -3.0000 + 0.0000i
 -12.0000 + 1.0000i
 -12.0000 - 1.0000i
 -12.0000 + 0.0000i
```



Part 6 Closed Loop Transfer Function

```
disp('--Part 6 --------------------------------------------------------')
disp(' ')

% Closed Loop Transfer Function Equatiom: Y(s)/R(s) = C*(sI - A + BK)^-1*B
Y_R         = simplify(C*(inv(s*eye(size(A)) - A + B*K))*B);

% Closed loop matrices with states [x,e], where e is x - z
A_tilde     = [(A - B*K), B*K; zeros(size(A)), (A - L*C)];
B_tilde     = [B;zeros(size(B))];
C_tilde     = [C,zeros(size(C))];
```

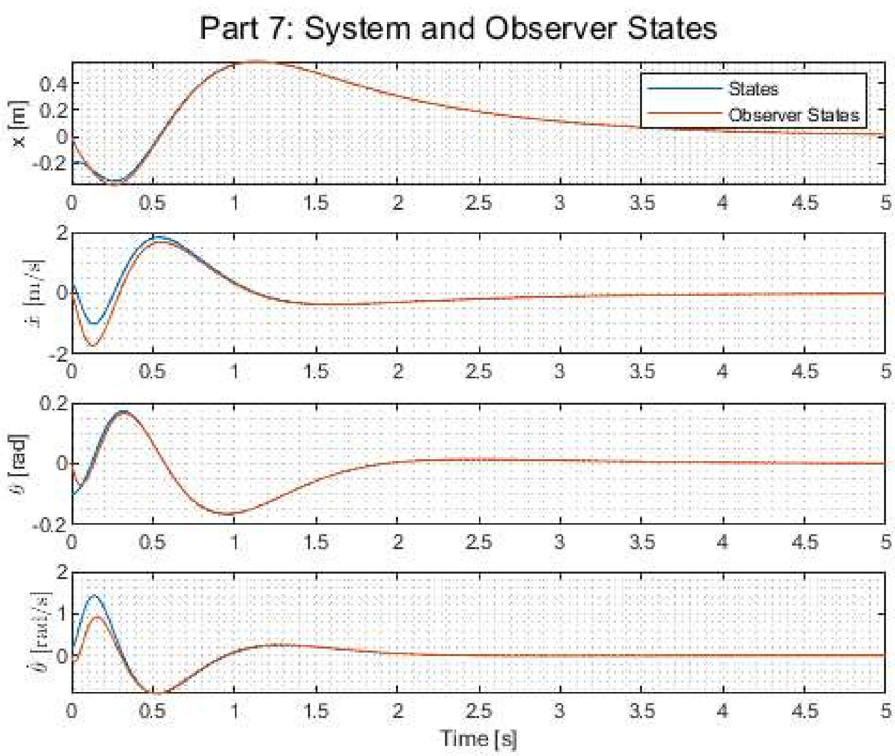
```
% Minimum realization for each output
[num,den]   = ss2tf(A_tilde,B_tilde,C_tilde,zeros(2,1),1);
Y1_R        = minreal(tf(num(1,:),den(1,:)));
Y2_R        = minreal(tf(num(2,:),den(1,:)));

% Verify both methods give same output
disp('Closed Loop Transfer Function: Y_1(s)/R(s): ss2tf output')
Y1_R
disp('Closed Loop Transfer Function: Y_1(s)/R(s): syms output')
disp(Y_R(1))

disp('Closed Loop Transfer Function: Y_2(s)/R(s): ss2tf output')
Y2_R
disp('Closed Loop Transfer Function: Y_2(s)/R(s): syms output')
disp(Y_R(2))
```

```
--Part 6 --------------------------------------------------------

Closed Loop Transfer Function: Y_1(s)/R(s): ss2tf output

Y1_R =

      0.1 s^2 - 1.732e-15 s - 1
  ----------------------------------
  s^4 + 13 s^3 + 61 s^2 + 117 s + 68

Continuous-time transfer function.

Closed Loop Transfer Function: Y_1(s)/R(s): syms output
(s^2 - 10)/(10*(s^4 + 13*s^3 + 61*s^2 + 117*s + 68))

Closed Loop Transfer Function: Y_2(s)/R(s): ss2tf output

Y2_R =

  -0.1 s^2 + 2.512e-16 s + 4.179e-30
  ----------------------------------
  s^4 + 13 s^3 + 61 s^2 + 117 s + 68

Continuous-time transfer function.

Closed Loop Transfer Function: Y_2(s)/R(s): syms output
-s^2/(10*(s^4 + 13*s^3 + 61*s^2 + 117*s + 68))
```

Part 7 State Observer with Non-Linear System

```
% ODE45 Function call
[T, X_obs]  = ode45(@(t,x) ControllerEstimatorCartPendulum(t,x,A,B,C,K,L,m,M,g,l),time,[-.2;.3;-.1;.2;z0],options);

figure
subplot(4,1,1)
sgtitle('Part 7: System and Observer States')
plot(T,X_obs(:,1),T,X_obs(:,5))
legend('States','Observer States')
ylabel('x [m]')
grid minor
subplot(4,1,2)
plot(T,X_obs(:,2),T,X_obs(:,6))
ylabel('$\dot{x}$ [m/s]','Interpreter','latex')
grid minor
subplot(4,1,3)
```

```
plot(T,X_obs(:,3),T,X_obs(:,7))
grid minor
ylabel('\theta [rad]')
subplot(4,1,4)
plot(T,X_obs(:,4),T,X_obs(:,8))
grid minor
ylabel('$\dot{\theta}$ [rad/s]','Interpreter','latex')
xlabel('Time [s]')
```



Functions

```
function xdot = NonlinearCartPendulum(t,state,u,m,M,g,l)
% Part 2 Function - nonlinear model

x1          = state(1,1);
x2          = state(2,1);
x3          = state(3,1);
x4          = state(4,1);

xdot(1,1)   = x2;
xdot(2,1)   = (u - m*g*cos(x3)*sin(x3) + m*l*x4^2*sin(x3))/(m + M - m*cos(x3)^2);
xdot(3,1)   = x4;
xdot(4,1)   = (-u*cos(x3) - m*l*x4^2*sin(x3)*cos(x3) + m*g*sin(x3) + M*g*sin(x3))/(l*(m + M - m*cos(x3)^2));

end

function [xdot]  = ControlledLinearCartPendulum(t,state,A,B,K)
% Part 3 Function - Linear Model with state feedback controller

% State Vector
x               = state(1:4,1);

% Feedback Control Law
u               = -K*x;

% State Space Equation
```

```
xdot            = A*x + B*u;
end

function [xdot] = ControlledCartPendulum(t,state,K,m,M,g,l)
% Part 4 Function - Non-linear model with state feedback controller

% State Vector
x1          = state(1,1);
x2          = state(2,1);
x3          = state(3,1);
x4          = state(4,1);
x           = state(1:4,1);

% Control Law
u           = -K*x;

% Non linear model
xdot(1,1)   = x2;
xdot(2,1)   = (u - m*g*cos(x3)*sin(x3) + m*l*x4^2*sin(x3))/(m + M - m*cos(x3)^2);
xdot(3,1)   = x4;
xdot(4,1)   = (-u*cos(x3) - m*l*x4^2*sin(x3)*cos(x3) + m*g*sin(x3) + M*g*sin(x3))/(l*(m + M - m*cos(x3)^2));

end

function [out] = ControllerEstimatorLinearCartPendulum(t,state,A,B,C,K,L)
% Part 5 Function - Linear Model with Luenberger Observer
% and Estimated State Feedback compensator

% State Vector
x       = state(1:4,1);

% Observer State Vector
z       = state(5:8,1);

% Control Law
u       = -K*z;

% Output, Y = Cx = [x1, x3]
Y       = [state(1,1);state(3,1)];

% Observer Dynamics -> dz/dt = Az + Bu + L(Y - C*z), u = -Kz
zdot    = (A - L*C - B*K)*z + L*Y;

% State Dynamics
xdot    = A*x + B*u;

out     = [xdot;zdot];
end

function [out] = ControllerEstimatorCartPendulum(t,state,A,B,C,K,L,m,M,g,l)
% Part 7 Function Non-Linear Model with Luenberger Observer
% and Estimated State Feedback compensator

% State Vector
x1          = state(1,1);
x2          = state(2,1);
x3          = state(3,1);
x4          = state(4,1);

% Observer State Vector
z           = state(5:8,1);

% Control Law
u           = -K*z;
```

```
% Output Y = [x1 x3]'
Y           = [x1;x3];

% Observer Dynamics
zdot        = (A - L*C - B*K)*z + L*Y;

% Non linear model
xdot(1,1)   = x2;
xdot(2,1)   = (u - m*g*cos(x3)*sin(x3) + m*l*x4^2*sin(x3))/(m + M - m*cos(x3)^2);
xdot(3,1)   = x4;
xdot(4,1)   = (-u*cos(x3) - m*l*x4^2*sin(x3)*cos(x3) + m*g*sin(x3) + M*g*sin(x3))/(l*(m + M - m*cos(x3)^2));

out         = [xdot;zdot];
end
```

```
--Part 2 ---------------------------------------------------------

Linearized State Space Model

A =

     0     1     0     0
     0     0    -1     0
     0     0     0     1
     0     0    11     0


B =

         0
    0.1000
         0
   -0.1000


C =

     1     0     0     0
     0     0     1     0


D =

     0
     0

System is Controllable
System is Observable
```

Published with MATLAB® R2021b