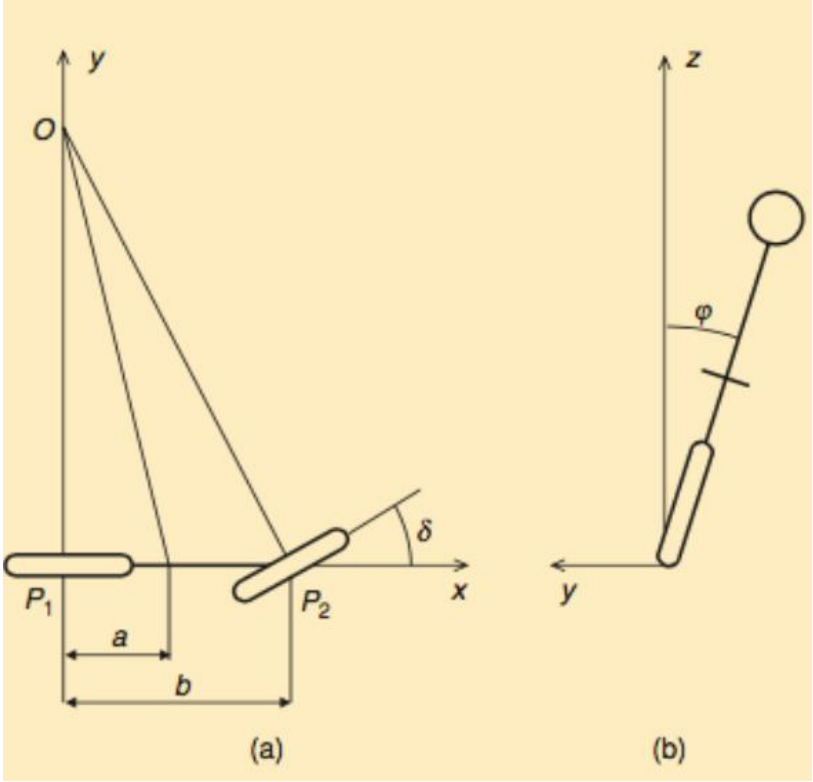
Problem Based Assignment 4

Callum Rohweder

METR7203
October 2017



The above figure represents the parameters associated with the Whipple bicycle model; where δ represents the change in steering angle and ϕ denotes the change in tilt angle (this is referred to as gamma in the proceeding MATLAB files) . The Whipple model describes the bicycle by the following system of equations:

$$M \begin{bmatrix} \ddot{\phi} \\ \ddot{\delta} \end{bmatrix} + C\, v_0 \begin{bmatrix} \dot{\phi} \\ \dot{\delta} \end{bmatrix} + (K_0 + K_2 v_0^2) \begin{bmatrix} \phi \\ \delta \end{bmatrix} = \begin{bmatrix} 0 \\ T \end{bmatrix}$$

To derive the individual equations of motion of the system, M, the mass matrix is constructed by 2x2 elements denoted $m_1, \dots, m\_4$, similarly for Cv_0 and the spring-like matrix $(K_0 + K_2 v_0^2)$, such at:

$$\begin{bmatrix} \mathrm{m}_1 & \mathrm{m}_2 \\ \mathrm{m}_3 & \mathrm{m}_4 \end{bmatrix} \begin{bmatrix} \ddot{\phi} \\ \ddot{\delta} \end{bmatrix} + \begin{bmatrix} \mathrm{c}_1 \mathrm{v}_0 & \mathrm{c}_2 \mathrm{v}_0 \\ \mathrm{c}_3 \mathrm{v}_0 & \mathrm{c}_4 \mathrm{v}_0 \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\delta} \end{bmatrix} + \begin{bmatrix} \mathrm{k}_1 & \mathrm{k}_2 \\ \mathrm{k}_3 & \mathrm{k}_4 \end{bmatrix} \begin{bmatrix} \phi \\ \delta \end{bmatrix} = \begin{bmatrix} 0 \\ T \end{bmatrix}$$

The equations of motion can be evaluated to be:

$$m_1 \ddot{\phi} + m_2 \ddot{\delta} + c_1 v_0 \dot{\phi} + c_2 v_0 \dot{\delta} + k_1 \phi + k_2 \delta = 0$$
$$m_3 \ddot{\phi} + m_4 \ddot{\delta} + c_3 v_0 \dot{\phi} + c_4 v_0 \dot{\delta} + k_3 \phi + k_4 \delta = T$$

To develop the state-space model, the states will need to be chosen, let:

$$x_1 = \phi, \quad x_2 = \dot{\phi}, \quad \dot{x_1} = x_2, \quad \dot{x_2} = \ddot{\phi}$$
$$x_3 = \delta, \quad x_4 = \dot{\delta}, \quad \dot{x_3} = x_4, \quad \dot{x_4} = \ddot{\delta}$$

Thus, the equations of motion described above can be rearranged to be:

$$\dot{x_2} = -\frac{m_2}{m_1}\dot{x_4} - \frac{c_1 v_0}{m_1}x_2 - \frac{c_2 v_0}{m_1}x_4 - \frac{k_1}{m_1}x_1 - \frac{k_2}{m_1}x_3$$

$$\dot{x_4} = -\frac{m_3}{m_4}\dot{x_2} - \frac{c_3 v_0}{m_4}x_2 - \frac{c_4 v_0}{m_4}x_4 - \frac{k_3}{m_4}x_1 - \frac{k_4}{m_4}x_3 + \frac{T}{m_4}$$

Part A

Using the equations of motion written with state-space variables, the state-space model can be computed using MATLAB. This is done by solving for $\dot{x_4}$ in the second state-space equation above, and then substituting this into the expression for $\dot{x_2}$, to gain an expression without derivative terms. This result is then put back into the last equation for an expression of $\dot{x_4}$ in terms of state variables. This is presented in the code below, where the elements of the M, C and $K_0 + K_2 v_0^2$ matrices are assigned to the appropriate element variables, from the equations of motion.

M th_dd + C v0 th_d + (K0 + K2*v0) th = [0; T]; where th = [gamma; delta] -> angles of bicycle dynamics let x1 = gamma, x2 = gamma_d, x3 = delta, x4 = delta_d thus, x1_d = x2, x2_d = gamma_dd x3_d = x4, x4_d = delta_dd expand matrix equations for 2 eom in terms of state-variables

```
syms x1 x2 x3 x4 x1_d x2_d x3_d x4_d T real
% get coefficents from matrices
m1 = M(1,1); m2 = M(1,2); m3=M(2,1); m4=M(2,2);
c1 = C0(1,1); c2=C0(1,2); c3=C0(2,1); c4=C0(2,2);
K = [K0 + K2*v0^2];
k1 = K(1,1); k2=K(1,2); k3=K(2,1); k4=K(2,2);

% form equations of motion using state-variables

%x2_d = (-m2/m1)*x4_d - (c1*v0/m1)*x2 - (c2*v0/m1)*x4-(k1/m1)*x1-(k2/m1)*x3;
%x4_d = (-m3/m4)*x2_d - (c3*v0/m4)*x2 - (c4*v0/m4)*x4-(k3/m4)*x1 - (k4/m4)*x3 + T/m4;

% rearrange and substitute expression for x2_d into eom of x4_d; eom2
x2_d_eqn = (-m2/m1)*x4_d - (c1*v0/m1)*x2 - (c2*v0/m1)*x4-(k1/m1)*x1-(k2/m1)*x3;
eom2 = (-m3/m4)*(x2_d_eqn) - (c3*v0/m4)*x2 - (c4*v0/m4)*x4-(k3/m4)*x1 - (k4/m4)*x3 + (T/m4) - x4_d;

% solve for x4_d and x2_d, solution denoted with _s
x4_d_s = collect(solve(eom2 == 0, x4_d), [x1; x2; x3; x4; T]);
x2_d_s = collect(subs(x2_d_eqn, x4_d, x4_d_s), [x1; x2; x3; x4; T]);
x1_d_s = x2;
x3_d_s = x4;
```

From here, the state-space equation can be constructed by arranging the coefficients of the state-derivate expressions into the state and input matrices. This is done in MATLAB, using the coeffs() function, and assigning values to A and B, as seen below. The eigenvalues of the state-space system for a velocity of 5ms^{-1} is computed, and the state-space representation highlighted.

```
% construct state-space presentation
A = [0 1 0 0;...
     0 0 0 0;...
     0 0 0 1;...
     0 0 0 0];

 B = [0; 0; 0; 0];

 x = [x1; x2; x3; x4];

 % put the coeffients from the solution to x2_d, into A,
 % and input term into B
[C_, T_] = coeffs(vpa(x2_d_s));

for i = 1:size(T_,2)
    if T_(i) == x1
        A(2, 1) = C_(i);
    elseif T_(i) == x2
        A(2, 2) = C_(i);
    elseif T_(i) == x3
        A(2, 3) = C_(i);
    elseif T_(i) == x4
        A(2, 4) = C_(i);
    elseif T_(i) == T
        B(2) = C_(i);
    end
end

% put the coeffients from x4_d into A, and input term into B
[C_, T_] = coeffs(vpa(x4_d_s));

for i = 1:size(T_,2)
    if T_(i) == x1
        A(4, 1) = C_(i);
    elseif T_(i) == x2
        A(4, 2) = C_(i);
    elseif T_(i) == x3
        A(4, 3) = C_(i);
    elseif T_(i) == x4
        A(4, 4) = C_(i);
    elseif T_(i) == T
        B(4) = C_(i);
    end
end

% view poles of the state-space model, in Hz
P_A = eig(A)/2/pi;

% findings: v0=5 yields two unstable poles, whilst v0=10 is stable
%
%
% x1_d  =         0    1.0000         0         0   * x1   +       0    * T
% x2_d       8.7611   -0.6370   23.2100    2.1486     x2       0.2922
% x3_d            0         0         0    1.0000     x3            0
% x4_d     -14.9477  -17.2465   29.1529  -12.9089     x4       7.9109
```

This results in the following state-space equation:

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_4} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 8.7611 & -0.637 & 23.210 & 2.1486 \\ 0 & 0 & 0 & 1 \\ -14.948 & -17.247 & 29.153 & -12.909 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 0 \\ 0.2922 \\ 0 \\ 79.109 \end{bmatrix} u$$

Of which has two unstable poles:

$$P_A = \begin{bmatrix} 0.2063 + 0.7326i \\ 0.2063 - 0.7326i \\ -0.2933 \\ -2.2753 \end{bmatrix}$$

And has a full rank controllability matrix, thus the system poles are moveable using state-feedback. This was worked out using the rank(ctrb(A,B)) command, which finds the rank of the controllability matrix of a system defined by A and B; the result was 4.

It was found that the system was stable for velocities between 6 and 10 meters per second, however in the above form is unstable.

Part B

The acker function can be used to place the poles of a state-space system, defined by A and B, at a desired location using state-feedback. The matrix on the feedback loop, K, gains the states back to the input, creating the desired closed loop poles.

The input thus becomes $u = -Kx$, resulting in a remodel defined by:

$$\dot{x} = Ax + B(-K * x)$$
$$\dot{x} = (A - BK)x$$
$$\dot{x} = A_f x$$

Where the the eigenvalues of A_f will be positioned at the desired locations; -2, -10, -1$\pm$1i. This can be seen in the code below.

u = -kx, x_d = Ax + Bu, y = Cx -> x_d = (A-Bk)x poles of closed loop system = eig(A-Bk) desired poles:

```
Pol = [-2 -10 -1+1i -1-1i];
% required feeback gain, k:
k = acker(A, B, Pol);

%k =
%    -1.2565    5.2073  -32.5252   -0.4145
clp = eig(A-B*k);
```

This gives:

$$clp = \begin{bmatrix} -10 \\ -1 + i \\ -1 - i \\ -2 \end{bmatrix}$$

Thus it correctly places the poles at the desired location.

Part 3

Disturbance input of delta

Consider the situation where, before state-feedback, a step change in delta was caused by an input, u. The state-space model would be:

$$\dot{x} = (A - BK)\begin{bmatrix} x_1 \\ x_2 \\ x_3 + u \\ x_4 \end{bmatrix}$$

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_4} \end{bmatrix} = A_f \begin{bmatrix} x_1 \\ x_2 \\ x_3 + u \\ x_4 \end{bmatrix}$$

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_4} \end{bmatrix} = A_f \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + A_f \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} u$$

$$\dot{x} = A_f x + B_f u$$

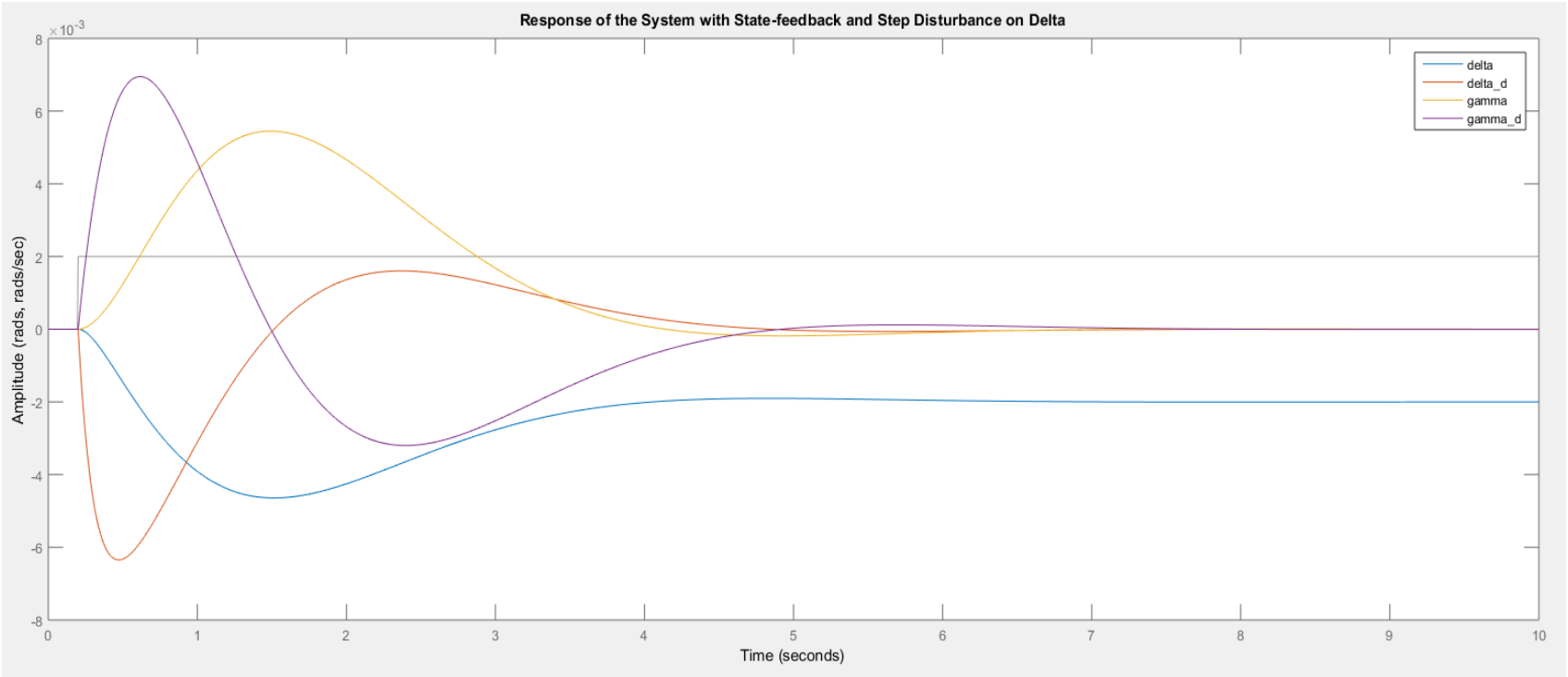
$$\dot{x} = (A - BK)x + (A - BK) * [0; 0; 1; 0]\, u$$

To view delta on the output, the C matrix is:

$$C = [0 \quad 0 \quad 1 \quad 0]$$

The result of a step input of 0.002 on this state can be modelled in MATLAB, in the following way:

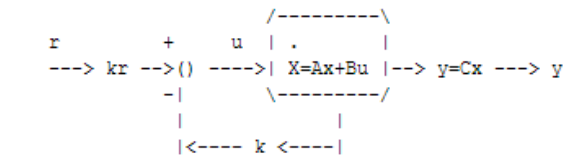
```
%                      .
%      u        +     x  /----------\  x
%      --> B_f ->() ---->| integral |-----> y=Cx ---> y
%               +|        \---------/   |
%                |                      |
%                |<---(A_f = A-Bk)<-----|
%
% with state-feedback and input change in delta
% x_d = (A - Bk) [x1 ; x2 ; (x3 + u) ; x4]
% x_d = (A - Bk) x + (A-Bk)[0 ; 0 ; 1 ; 0] u
% x_d = A_f * x + B_f * u

sr = [0;0;1;0];                          % selected state for step input
A_f = A-B*k;                             % state matrix with state-feedback
B_f = A_f*sr;                            % construct B matrix
C = [0;0;1;0]';                          % view only delta on the output
D = 0;
sys_delta = ss(A_f,B_f,C,D);

% apply a step input with size of 0.002, plot response of each state
% using the step() function
opt = stepDataOptions;
opt.StepAmplitude = 0.002;               % step amplitude set
t = (0:0.005:10);                        % time set
figure(1); hold on;
step(sys_delta, t, opt);                 % response on delta
sys_gamma = ss(A_f,B_f,[1,0,0,0],D);     % response on gamma
sys_gamma_d = ss(A_f,B_f,[0,1,0,0],D);   % response on gamma_d
step(sys_gamma, t, opt);
sys_delta_d = ss(A_f,B_f,[0,0,0,1],D);   % response on delta_d
step(sys_delta_d, t, opt);
hold off;
```



It can be seen that all of the state's change as a result of this disturbance, where the steering angle remains at -0.002 and the other's return to their initial conditions. The response of delta is negative due to the input matrix B_f containing a component related to -B*K, of magnitude greater than A, thus resulting in a sign change when evaluated. This can be noticed when computing B_f, giving [0; 20.7544; 0; -37.333], where when state 4 is integrated it results in delta being multiplied by (-37.33 + some feedback component).

In a physical sense, a disturbance in position of the steering angle will result in a coupling with the tilt angle, which is not evident in the above graph. Thus in order to simulate a change in steering angle, this will need to be done through developing a model with an input torque that creates a steady-state step change in angle. This procedure follows:

State reference input



insert a reference for state (delta)

```
% x_d = (A-Bk)*x + (kr*B)*r
% x_d = A_star*x + B_star*r
% where r is a reference step input for delta

A_star = A-B*k;
C = [0,0,1,0];                              %design kr for only delta on output
D = 0;
kr = -(C*(A-B*k)^-1 *B)^-1;%-(rscale(ss(A-B*k,B,C,D),k) - 1.68);     %equal to -0.541 (rscale calculates Kr)
B_star = kr*B;

% apply a step input with size of 0.002, plot response of each state
% using the step() function
t = (0:0.005:10);                       % time set
figure(6); hold on;
sys_delta = ss(A_star,B_star,C,D);
sys_gamma = ss(A_star,B_star,[1,0,0,0],D);    % response on gamma
sys_gamma_d = ss(A_star,B_star,[0,1,0,0],D);  % response on gamma_d
sys_delta_d = ss(A_star,B_star,[0,0,0,1],D);  % response on delta_d

lsim(sys_delta, u, t);
lsim(sys_delta_d, u, t);
lsim(sys_gamma, u, t);
lsim(sys_gamma_d, u, t);
% lsim(syslqrcl, u, t);

hold off;
```

For the system presented above, it can be seen that there is a reference input for the state, r, and gain kr, which are inputted into the system along with the negative state-feedback. Similarly the state-space equations of this model can be written as follows:

$$\dot{x} = Ax + r * k_r * B - BKx$$
$$\dot{x} = (A - BK)x + k_r B\, r$$

During steady-state, there is no change in angles (gamma and delta), thus the angle's velocities and accelerations are zero, resulting in $\dot{x} = 0$. Thus, the value of kr to develop a steady state reference r on delta (the output is only delta, the other states are 0) can be computed as follows:

$$0 = (A - BK)x + k_r B\, r$$
$$x = -(A - BK)^{-1} B\, k_r r$$

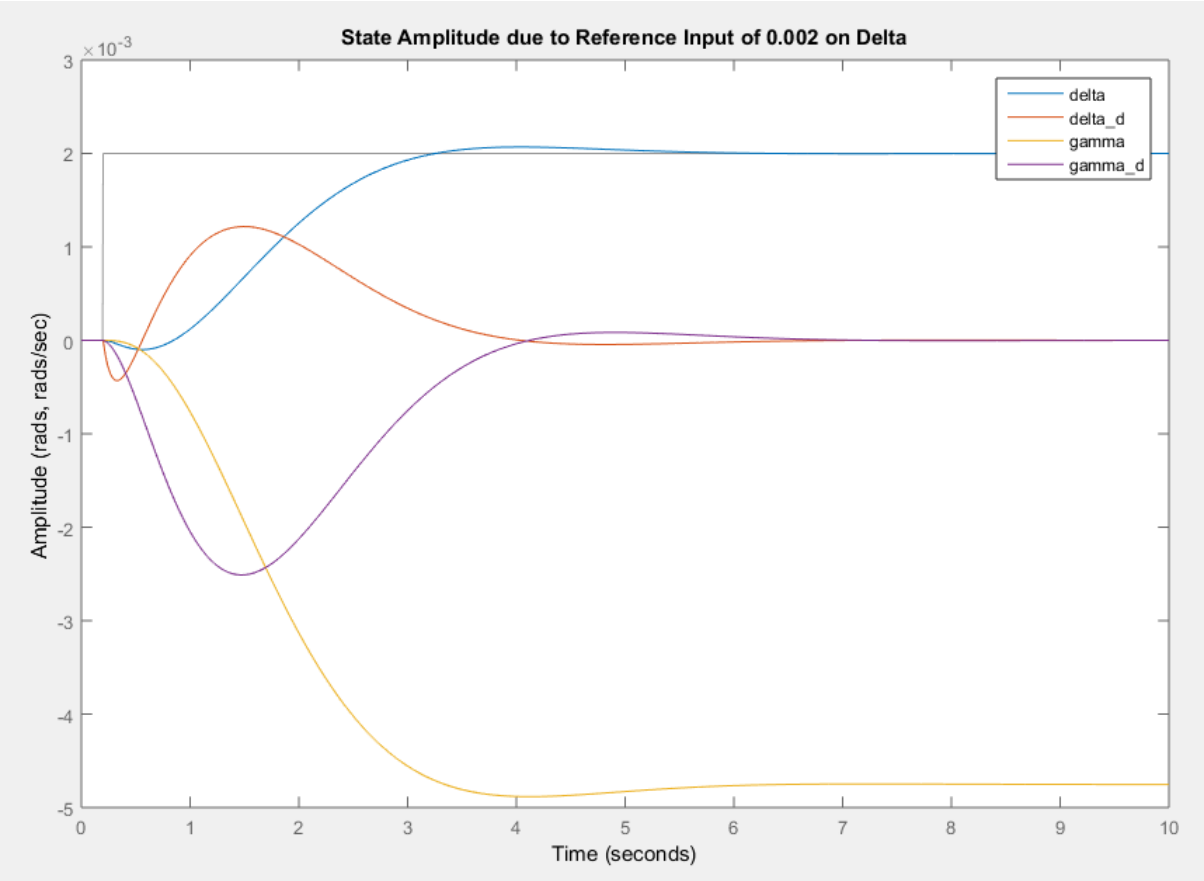
similarly, at the output:

$$y = Cx$$
$$y = -C(A - BK)^{-1} B\, k_r\, r$$

As the output is only made up of the state of delta, we desire y = r:

$$k_r = \frac{1}{C\,(A - BK)^{-1} B} = -0.5429$$

It can be seen that the code above computes this, and the response of the states to a step input of 0.002, with kr designed to produce this response on delta:



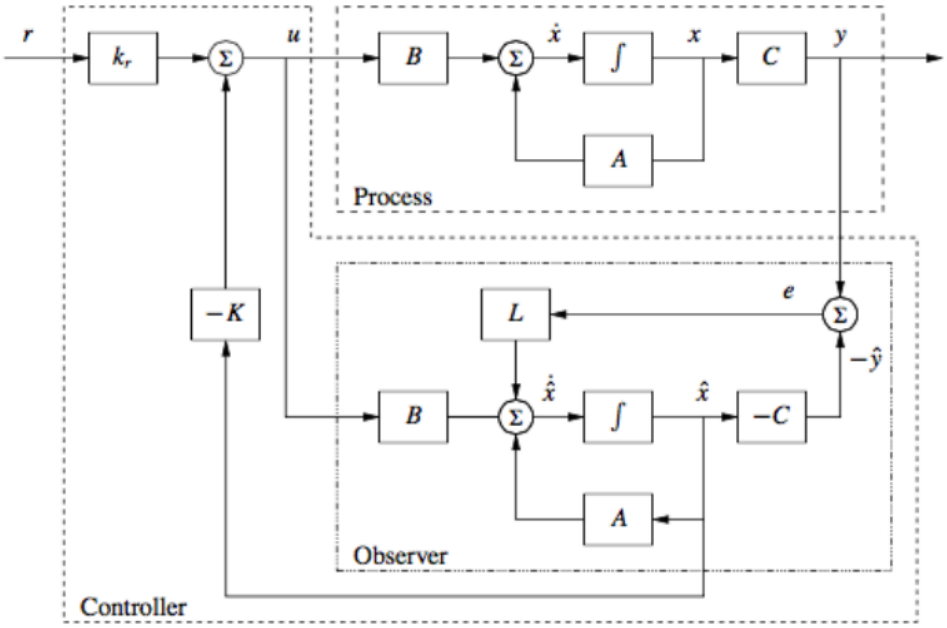
The coupling between the steering and tilt angles are evident in this circumstance.

Part D

Below is a block-diagram representation of an observer, realistic process or plant, and the associated controller. Output feedback with a gain matrix, L, is used to place the poles of the system and correct for error between the plant and observer model. Without being able to directly measure the states of the plant, state-feedback comes from the observer state's, $\hat{x}$; designed to mimic that of the plant. Similarly to the previous section, the response of this system can be due to an input r and gain kr. The poles of the observer are typically designed to be 4 or 5 times the size of the plant, in order to correct for unexpected changes in the plant's output. The state-space model can be written as:

$$\dot{x} = Ax - BK\,\hat{x} + B\,k_r r$$
$$\dot{\hat{x}} = A\,\hat{x} - BK\,\hat{x} + B\,k_r\,r + L(y - \hat{y})$$
$$\dot{\hat{x}} = A(-BK)\hat{x} + B\,k_r\,r + LC(x - \hat{x})$$

This can be rearranged into state-space form where the state variables are the four of those from the plant, and the 4 'equivalent' from the observer, denoted x and $\hat{x}$, respectively.



The state-space representation of this system can described by:

$$\begin{bmatrix}\dot{x}\\ \dot{\hat{x}}\end{bmatrix} = \begin{bmatrix}A & -BK\\ LC & (A-BK-LC)\end{bmatrix}\begin{bmatrix}x\\ \hat{x}\end{bmatrix} + \begin{bmatrix}B\ k_r\\ B\ k_r\end{bmatrix} r$$
$$y = [C \quad 0]\begin{bmatrix}x\\ \hat{x}\end{bmatrix}$$

The eigenvalues of this system can be calculated in a similar manner to those in part B, however it was not shown, they are calculated by finding the determinant of (sI – A*); where I is the identity matrix.

$$\det(sI - A^*) = \begin{vmatrix}sI - A & BK\\ -LC & (sI - A + BK + LC)\end{vmatrix}$$

Let row2 = row1 – row2:

$$= \begin{vmatrix}sI - A + LC & -sI + A - LC\\ -LC & (sI - A + BK + LC)\end{vmatrix}$$

let column2 = column 1 + column 2:

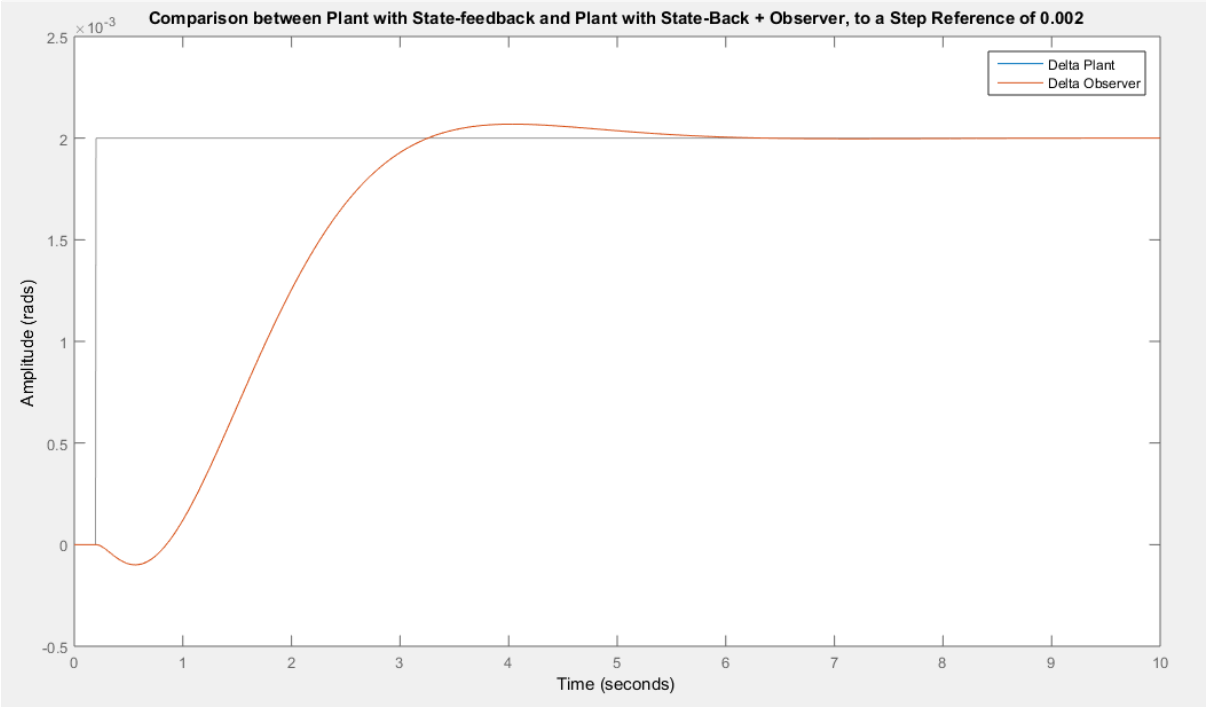
$$= \begin{vmatrix}sI - A + LC & 0\\ -LC & (sI - A + BK)\end{vmatrix}$$
$$= \det\big(sI - (A - LC)\big)\det\big(sI - (A - BK)\big)$$
$$= 0$$

It can be seen above that the eigenvalues of the system are a combination of poles of the observer and poles of the state-feedback; where the state-feedback poles have already been chosen. Therefore, the eigenvalues of the observer can be chosen in a similar manner, using MATLAB's acker() or place() function, such that:

$$L = place(A^T, C^T, P), \qquad where\ P\ contains\ the\ pole\ locations$$

The desired pole locations are P = [-8, -16, -4+1i, -4-1i]. The code below calculates the state-space model, the required L matrix for the aforementioned poles, and then plots the response of the model from part C and this new model to a step input reference; defined in the previous task.

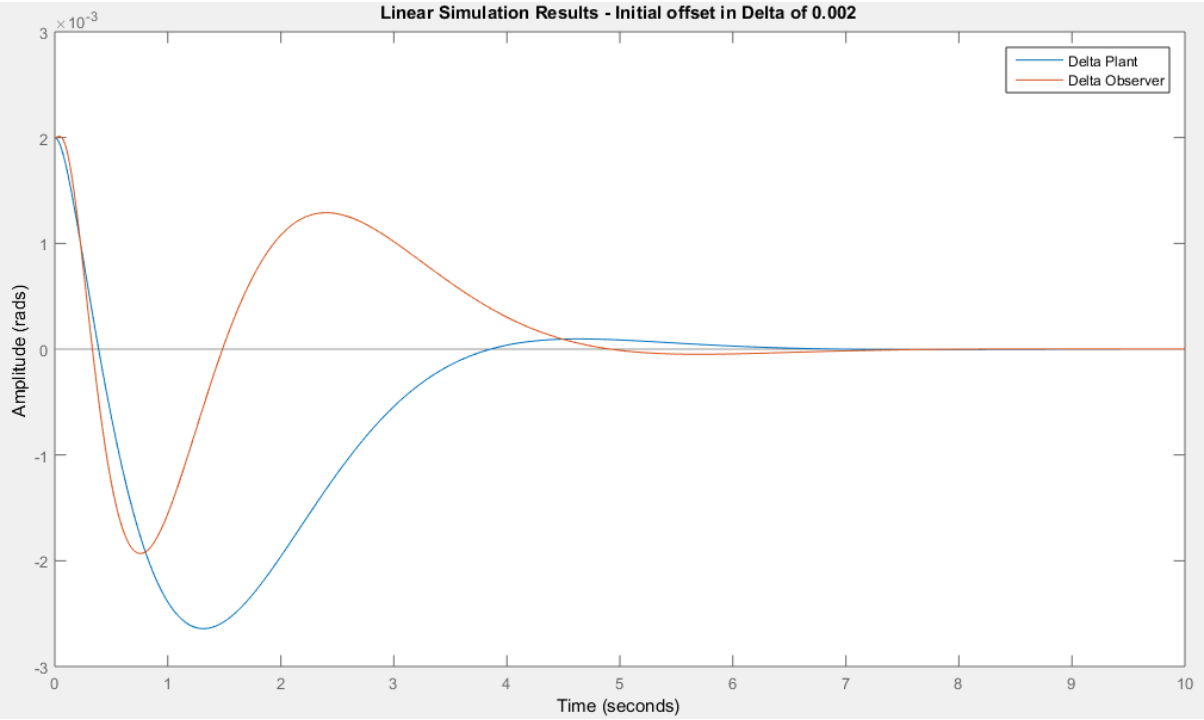
```
P = [-8 -16 -4+1i -4-1i];           % pole locations for observer
L = place(A', C', P)';              % poles at eig(A-LC);

% state space model for complete state-feeback and observer

A_p = [A -B*k; L*C A-B*k-L*C];
B_p = [B*kr; B*kr];
C_p = [C zeros(size(C))];
C_p_hat = [zeros(size(C)) C];

sys_p = ss(A_star, B_star, C, D);           % system which outputs plant delta
sys_p_hat = ss(A_p, B_p, C_p, 0);       % system which outputs observer delta

figure(8); hold on;
no_input = zeros(size(t));              % no step input

%show response for initial disturbance on plant delta
lsim(sys_p, u, t, [0, 0, 0, 0]);
lsim(sys_p_hat, u, t, [0, 0, 0, 0, zeros(1,4)]);
hold off;
```



It can be seen that the observer's delta matches the plant's exactly, when a step input is applied.

Interestingly, if the plant had a state originally offset from those of the observer, the responses don't exactly match. With an initial state of 0.002, it can be seen that the observers unforced response is faster than the plant's:

```
%show response for initial disturbance on plant delta
lsim(sys_p, no_input, t, [0, 0, 0.002, 0]);
lsim(sys_p_hat, no_input, t, [0, 0, 0.002, 0, zeros(1,4)]);
```

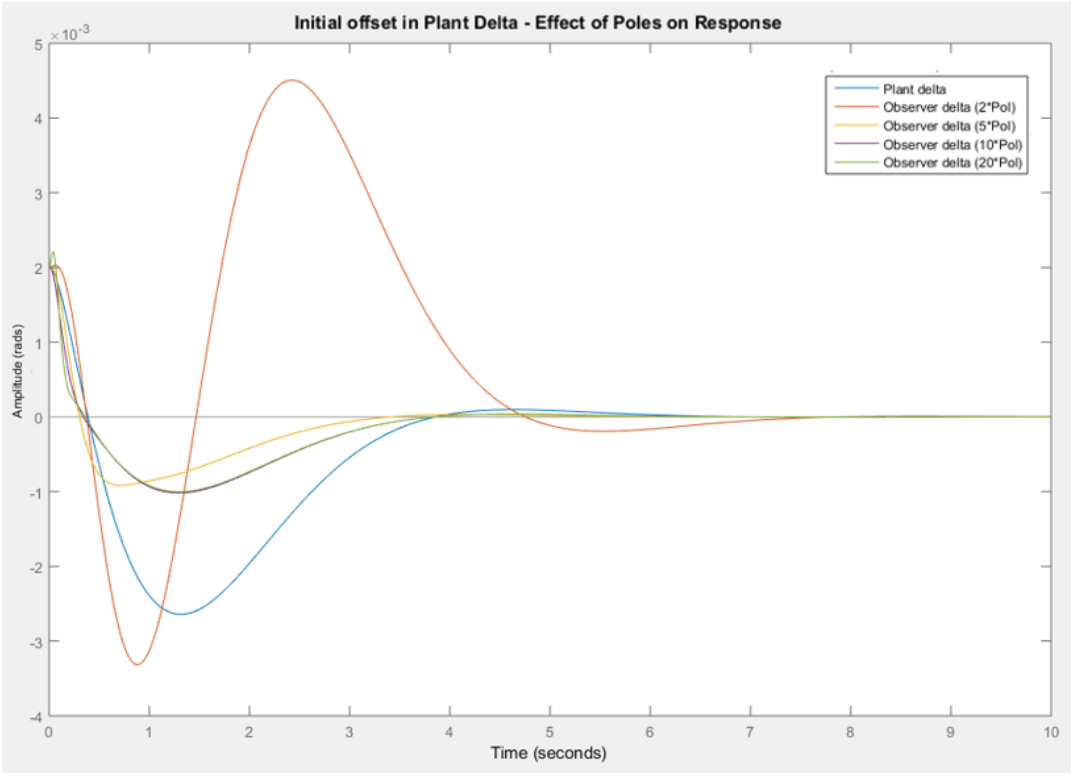


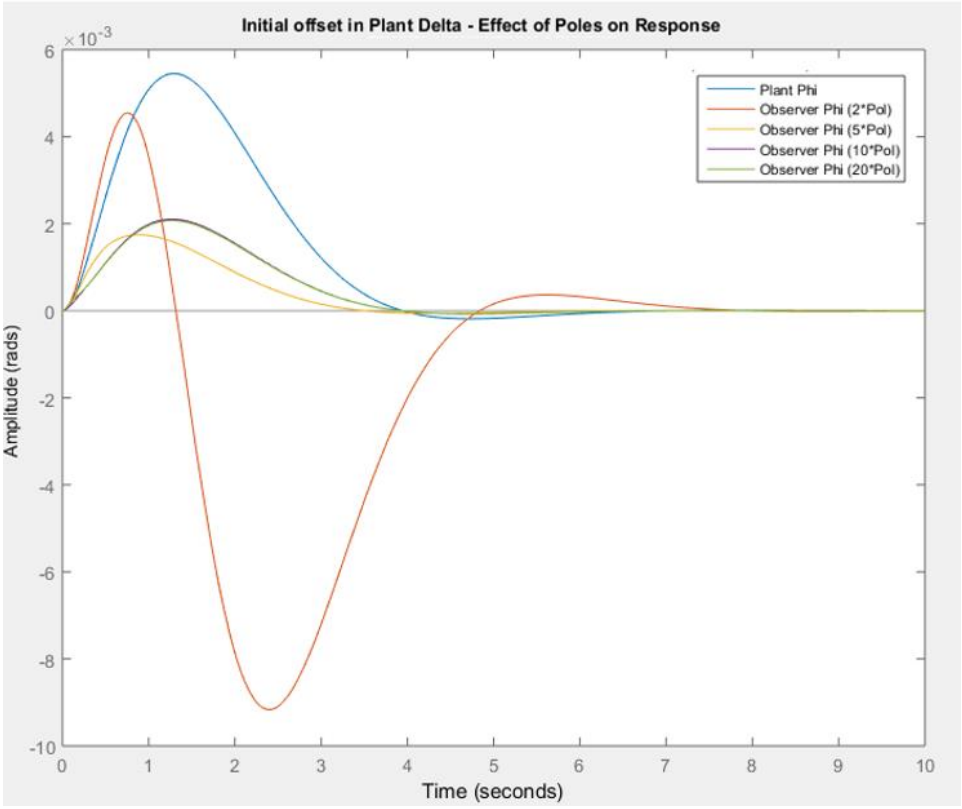
Part E

To understand the effect of the observer poles, in observing a plant with state-feedback poles denoted in the matrix Pol, the observer poles were changed to different multiples to that of the state-feedback poles. The responses of the different observers were plotted against the plant's response, for both delta and phi, illustrated below. It can be seen that the observer with poles at 10*Pol and 20*Pol had identical responses, following the plant in shape but lower in amplitude. When the multiple of Pol was too low, below 2, the response became increasingly unstable. This is due to the observer compensating for the error between the plant and observer state's too slowly, leading to an increase in error, and increase in amplitude, but a too large of a rise time to compensate.

Having high frequency poles allows the observer to respond to changes in the plant's state without noticeable deviation. High frequency poles can be sensitive to noise, and cause saturation of actuators, so there may be design considerations associated with how fast the poles can be. From the responses presented below, it is noticeable that poles 10 times those of the plant is sufficient for providing a similar shaped response. From research, most sources agree that 5 times faster is sufficient, however some too sight 10; again, this could be a design consideration.

The plots were generated by replicating the previous part, part D, various times for different scalings of Pol. Other variations of poles were simulated and the outcome the same, the response is constantly the same for frequencies higher than 10 times the plant poles, with the poles having a damping ratio between .707 and 1.


Initial offset in Plant Delta - Effect of Poles on Response
×10^-3
Plant delta
Observer delta (2*Pol)
Observer delta (5*Pol)
Observer delta (10*Pol)
Observer delta (20*Pol)
Amplitude (rads)
Time (seconds)


Initial offset in Plant Delta - Effect of Poles on Response
×10^-3
Plant Phi
Observer Phi (2*Pol)
Observer Phi (5*Pol)
Observer Phi (10*Pol)
Observer Phi (20*Pol)
Amplitude (rads)
Time (seconds)

to show the effect of observer poles on replicating the plant Pol is a matrix containing the poles of the state-feedback component of the plant

```
% make the observer poles a multiple of the plant observes, my multiplying
% Pol by an element in co before evaluating L.
co = [2, 5, 10, 20];

% For each multiple of Pol, plot the response against the plant for an
% initial state offset -> delta by 0.002

for i = 1:4
    % outputing delta
    C = [0,0,1,0];
    % place poles of observer at a multiple of Pol
    L = place(A', C', co(i)*Pol)';              % poles at eig(A-LC);
    A_p = [A -B*k; L*C A-B*k-L*C];              % observer A
    B_p = [B*kr; B*kr];                         % observer B
    C_p = [C zeros(size(C))];                   % observer C

    % retrieve the state-space model for an output of delta only
    if i == 1
        sys_delta = ss(A_star, B_star, C, D);        % system which outputs plant delta
    end
    sys_delta_hat = ss(A_p, B_p, C_p, 0);        % system which outputs observer delta
    % retrieve the state-space model for an output of phi
    C = [1,0,0,0];
    C_p_hat = [ C, zeros(size(C))];
    if i == 1
        sys_phi = ss(A_star, B_star, C, D);        % system which outputs plant delta
    end
    sys_phi_hat = ss(A_p, B_p, C_p_hat, 0);        % system which outputs observer delta
    figure(9); hold on;
    % plot response on phi
    if i == 1
        % plot the plant response, only once
        lsim(sys_phi, no_input, t, [0, 0, 0.002, 0]);
    end
    lsim(sys_phi_hat, no_input, t, [0, 0, 0.002, 0, zeros(1,4)]);
    hold off;
    figure(10); hold on;
    % plot reponse on delta
    if i == 1
        % plot the plant response, only once
        lsim(sys_delta, no_input, t, [0, 0, 0.002, 0]);
    end
    lsim(sys_delta_hat, no_input, t, [0, 0, 0.002, 0, zeros(1,4)]);
    hold off;

end
```

Contents


```
clear all;
close all;
figure(1);
figure(2);
figure(3);
% Parameters that are used for the Whipple bicycle model. The model is
% based on the linearized 4th order model and analysis of eigenvalues
% from IEEE CSM (25:4) August 2005 pp 26-47

%   Basic data is given by 26 parameters
g = 9.81;                       % Acceleration of gravity [m/s^2]
b = 1.00;                       % Wheel base [m]
c = 0.08;                       % Trail [m]
Rrw = 0.35; Rfw = 0.35;         % Wheel radii
lambda = pi*70/180;             % Head angle [radians]

% Rear frame mass [kg], center of mass [m], and inertia tensor [kgm^2]
mrf=12;xrf=0.439;zrf=0.579;
Jxxrf=0.475656;Jxzrf=0.273996;Jyyrf=1.033092;Jzzrf=0.527436;
mrf=87;xrf=0.491586;zrf=1.028138;
Jxxrf=3.283666;Jxzrf=0.602765;Jyyrf=3.8795952;Jzzrf=0.565929;

% Front frame mass [kg], center of mass [m], and inertia tensor [kgm^2]
mff=2;xff=0.866;zff=0.676;
Jxxff=0.08;Jxzff=-0.02;Jyyff=0.07;Jzzff=0.02;

% Rear wheel mass [kg], center of mass [m], and inertia tensor [kgm^2]
mrw=1.5;Jxxrw=0.07;Jyyrw=0.14;

% Front wheel mass [kg], center of mass [m], and inertia tensor [kgm^2]
mfw=1.5;Jxxfw=0.07;Jyyfw=0.14;

% Auxiliary variables
xrw=0;zrw=Rrw;xfw=b;zfw=Rfw;
Jzzrw=Jxxrw;Jzzfw=Jxxfw;
mt=mrf+mrw+mff+mfw;
xt=(mrf*xrf+mrw*xrw+mff*xff+mfw*xfw)/mt;
zt=(mrf*zrf+mrw*zrw+mff*zff+mfw*zfw)/mt;
Jxxt=Jxxrf+mrf*zrf^2+Jxxrw+mrw*zrw^2+Jxxff+mff*zff^2+Jxxfw+mfw*zfw^2;
Jxzt=Jxzrf+mrf*xrf*zrf+mrw*xrw*zrw+Jxzff+mff*xff*zff+mfw*xfw*zfw;
Jzzt=Jzzrf+mrf*xrf^2+Jzzrw+mrw*xrw^2+Jzzff+mff*xff^2+Jzzfw+mfw*xfw^2;
mf=mff+mfw;
xf=(mff*xff+mfw*xfw)/mf;zf=(mff*zff+mfw*zfw)/mf;
Jxxf=Jxxff+mff*(zff-zf)^2+Jxxfw+mfw*(zfw-zf)^2;
Jxzf=Jxzff+mff*(xff-xf)*(zff-zf)+mfw*(xfw-xf)*(zfw-zf);
Jzzf=Jzzff+mff*(xff-xf)^2+Jzzfw+mfw*(xfw-xf)^2;
d=(xf-b-c)*sin(lambda)+zf*cos(lambda);
Fll=mf*d^2+Jxxf*cos(lambda)^2+2*Jxzf*sin(lambda)*cos(lambda)+Jzzf*sin(lambda)^2;
Flx=mf*d*zf+Jxxf*cos(lambda)+Jxzf*sin(lambda);
Flz=mf*d*xf+Jxzf*cos(lambda)+Jzzf*sin(lambda);
gamma=c*sin(lambda)/b;
Sr=Jyyrw/Rrw;Sf=Jyyfw/Rfw;St=Sr+Sf;Su=mf*d+gamma*mt*xt;

% Matrices for linearized fourth order model
M=[Jxxt -Flx-gamma*Jxzt;-Flx-gamma*Jxzt Fll+2*gamma*Flz+gamma^2*Jzzt];
```

```
K0=[-mt*g*zt g*Su;g*Su  -g*Su*cos(lambda)];
K2=[0 -(St+mt*zt)*sin(lambda)/b;0 (Su+Sf*cos(lambda))*sin(lambda)/b];
c12=gamma*St+Sf*sin(lambda)+Jxzt*sin(lambda)/b+gamma*mt*zt;
c22=Flz*sin(lambda)/b+gamma*(Su+Jzzt*sin(lambda)/b);
C0=[0 -c12;(gamma*St+Sf*sin(lambda)) c22];
one=diag([1 1]);null=zeros(2,2);

% Nominal velocity
v0=5;
```

Part A

M th_dd + C v0 th_d + (K0 + K2*v0) th = [0; T]; where th = [gamma; delta] -> angles of bicycle dynamics let x1 = gamma, x2 = gamma_d, x3 = delta, x4 = delta_d thus, x1_d = x2, x2_d = gamma_dd x3_d = x4, x4_d = delta_dd expand matrix equations for 2 eom in terms of state-variables

```
syms x1 x2 x3 x4 x1_d x2_d x3_d x4_d T real
% get coefficents from matrices
m1 = M(1,1); m2 = M(1,2); m3=M(2,1); m4=M(2,2);
c1 = C0(1,1); c2=C0(1,2); c3=C0(2,1); c4=C0(2,2);
K = [K0 + K2*v0^2];
k1 = K(1,1); k2=K(1,2); k3=K(2,1); k4=K(2,2);

% form equations of motion using state-variables

%x2_d = (-m2/m1)*x4_d - (c1*v0/m1)*x2 - (c2*v0/m1)*x4-(k1/m1)*x1-(k2/m1)*x3;
%x4_d = (-m3/m4)*x2_d - (c3*v0/m4)*x2 - (c4*v0/m4)*x4-(k3/m4)*x1 - (k4/m4)*x3 + T/m4;

% rearrange and substitute expression for x2_d into eom of x4_d; eom2
x2_d_eqn = (-m2/m1)*x4_d - (c1*v0/m1)*x2 - (c2*v0/m1)*x4-(k1/m1)*x1-(k2/m1)*x3;
eom2 = (-m3/m4)*(x2_d_eqn) - (c3*v0/m4)*x2 - (c4*v0/m4)*x4-(k3/m4)*x1 - (k4/m4)*x3 + (T/m4) - x4_d;

% solve for x4_d and x2_d, solution denoted with _s
x4_d_s = collect(solve(eom2 == 0, x4_d), [x1; x2; x3; x4; T]);
x2_d_s = collect(subs(x2_d_eqn, x4_d, x4_d_s), [x1; x2; x3; x4; T]);
x1_d_s = x2;
x3_d_s = x4;

% construct state-space presentation
A = [0 1 0 0;...
     0 0 0 0;...
     0 0 0 1;...
     0 0 0 0];

 B = [0; 0; 0; 0];

 x = [x1; x2; x3; x4];

 % put the coeffients from the solution to x2_d, into A,
 % and input term into B
```

```
[C_, T_] = coeffs(vpa(x2_d_s));

for i = 1:size(T_,2)
    if T_(i) == x1
        A(2, 1) = C_(i);
    elseif T_(i) == x2
        A(2, 2) = C_(i);
    elseif T_(i) == x3
        A(2, 3) = C_(i);
    elseif T_(i) == x4
        A(2, 4) = C_(i);
    elseif T_(i) == T
        B(2) = C_(i);
    end
end

% put the coeffients from x4_d into A, and input term into B
[C_, T_] = coeffs(vpa(x4_d_s));

for i = 1:size(T_,2)
    if T_(i) == x1
        A(4, 1) = C_(i);
    elseif T_(i) == x2
        A(4, 2) = C_(i);
    elseif T_(i) == x3
        A(4, 3) = C_(i);
    elseif T_(i) == x4
        A(4, 4) = C_(i);
    elseif T_(i) == T
        B(4) = C_(i);
    end
end

% view poles of the state-space model, in Hz
P_A = eig(A)/2/pi;

% findings: v0=5 yields two unstable poles, whilst v0=10 is stable
%
%
% x1_d  =         0    1.0000         0         0   * x1   +       0    * T
% x2_d       8.7611   -0.6370   23.2100    2.1486     x2       0.2922
% x3_d            0         0         0    1.0000     x3            0
% x4_d     -14.9477  -17.2465   29.1529  -12.9089     x4       7.9109
```

Part B

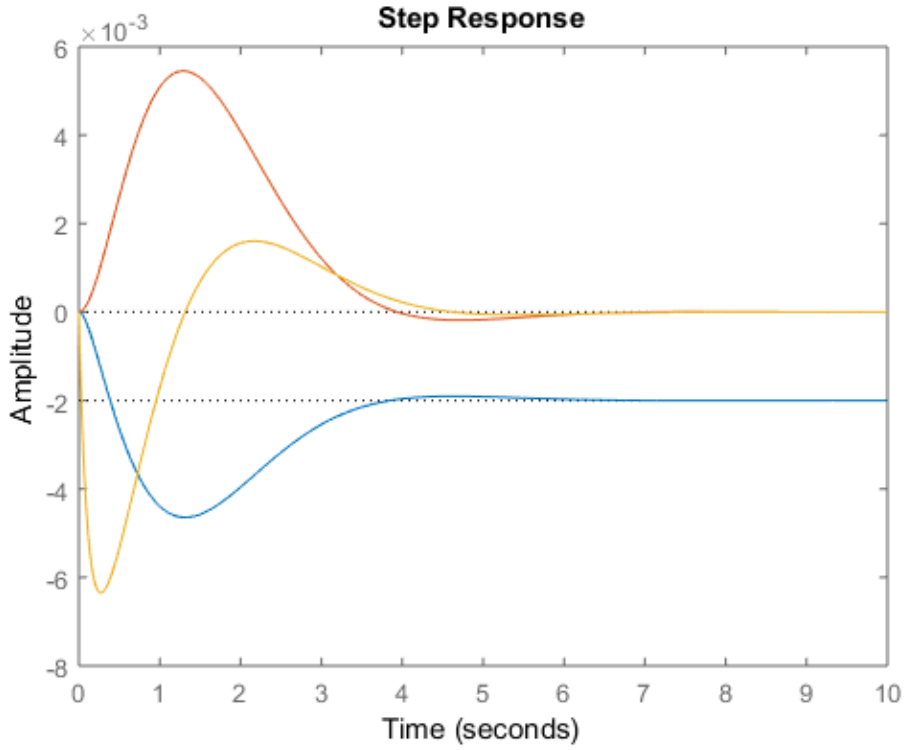
u = -kx, x_d = Ax + Bu, y = Cx -> x_d = (A-Bk)x poles of closed loop system = eig(A-Bk) desired poles:

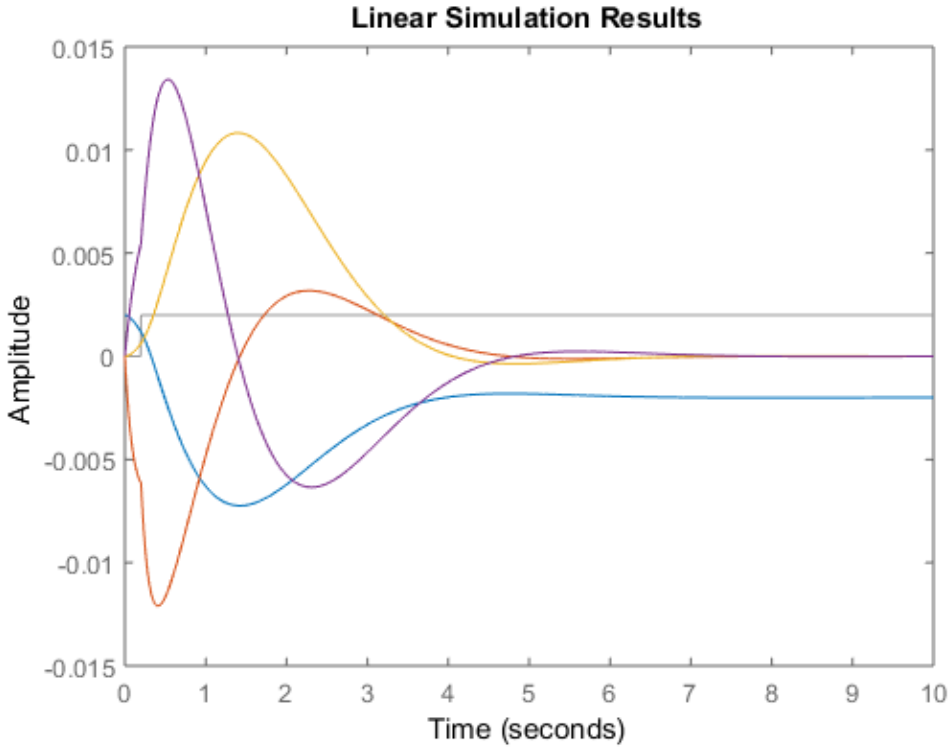
```
Pol = [-2 -10 -1+1i -1-1i];
% required feeback gain, k:
k = acker(A, B, Pol);

%k =
%    -1.2565    5.2073  -32.5252   -0.4145
clp = eig(A-B*k);
```

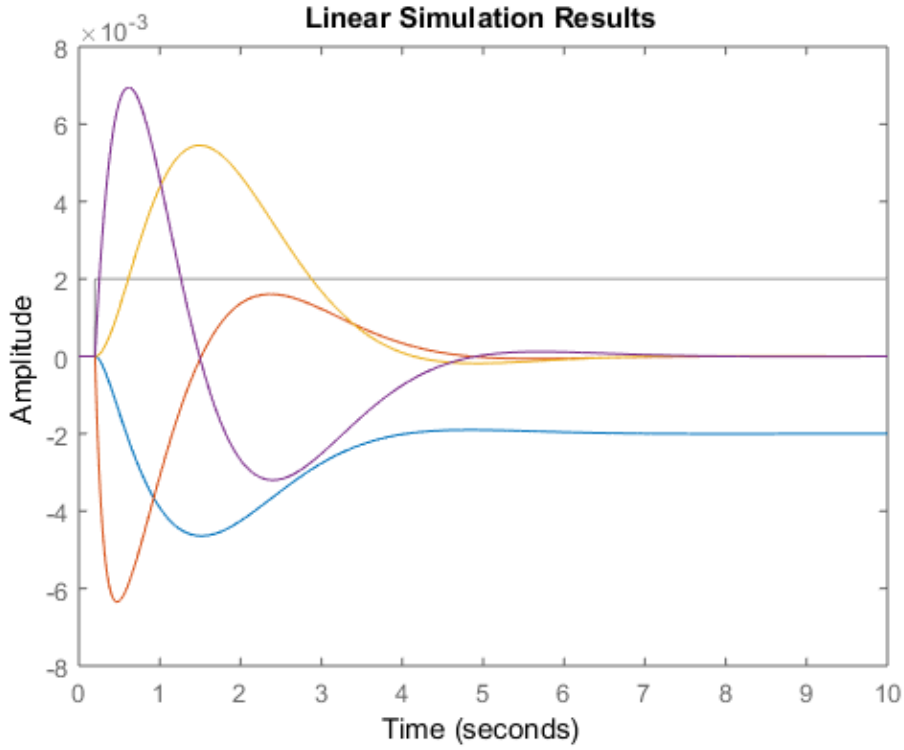
Part C

```
%                    .
%      u        +     x  /----------\  x
%      --> B_f ->() ---->| integral |-----> y=Cx ---> y
%               +|        \---------/  |
%                |                     |
%                |<---(A_f = A-Bk)<----|
%
```

```
% with state-feedback and input change in delta
% x_d = (A - Bk) [x1 ; x2 ; (x3 + u) ; x4]
% x_d = (A - Bk) x + (A-Bk)[0 ; 0 ; 1 ; 0] u
% x_d = A_f * x + B_f * u

sr = [0;0;1;0];                         % selected state for step input
A_f = A-B*k;                            % state matrix with state-feedback
B_f = A_f*sr;                           % construct B matrix
C = [0;0;1;0]';                         % view only delta on the output
D = 0;
sys_delta = ss(A_f,B_f,C,D);

% apply a step input with size of 0.002, plot response of each state
% using the step() function
opt = stepDataOptions;
opt.StepAmplitude = 0.002;              % step amplitude set
t = (0:0.005:10);                       % time set
figure(1); hold on;
step(sys_delta, t, opt);                % response on delta
sys_gamma = ss(A_f,B_f,[1,0,0,0],D);    % response on gamma
sys_gamma_d = ss(A_f,B_f,[0,1,0,0],D);  % response on gamma_d
step(sys_gamma, t, opt);
sys_delta_d = ss(A_f,B_f,[0,0,0,1],D);  % response on delta_d
step(sys_delta_d, t, opt);
hold off;

% model using lsim, providing an initial condition for delta = 0.002

% construct a step input, u
 u = [];
 for n = 1:size(t,2)
     if t(n) < 0.2
         u(n) = 0;
     else
         u(n) = 0.002;
     end
 end

 figure(2); hold on;
 lsim(sys_delta, u, t, [0;0;0.002;0]);
 lsim(sys_delta_d, u, t, [0;0;0.002;0]);
 lsim(sys_gamma, u, t, [0;0;0.002;0]);
 lsim(sys_gamma_d, u, t, [0;0;0.002;0]);
 hold off;

 % without the initial condition, can be seen that the same as
 % using the step function
 figure(3); hold on;
 lsim(sys_delta, u, t, [0;0;0;0]);
 lsim(sys_delta_d, u, t, [0;0;0;0]);
 lsim(sys_gamma, u, t, [0;0;0;0]);
 lsim(sys_gamma_d, u, t, [0;0;0;0]);
% lsim(syslqrcl, u, t);
 hold off;
```


Step Response
×10^-3
6
4
2
0
-2
-4
-6
-8
Amplitude
0
1
2
3
4
5
6
7
8
9
10
Time (seconds)


Linear Simulation Results
0.015
0.01
0.005
0
-0.005
-0.01
-0.015
Amplitude
0
1
2
3
4
5
6
7
8
9
10
Time (seconds)



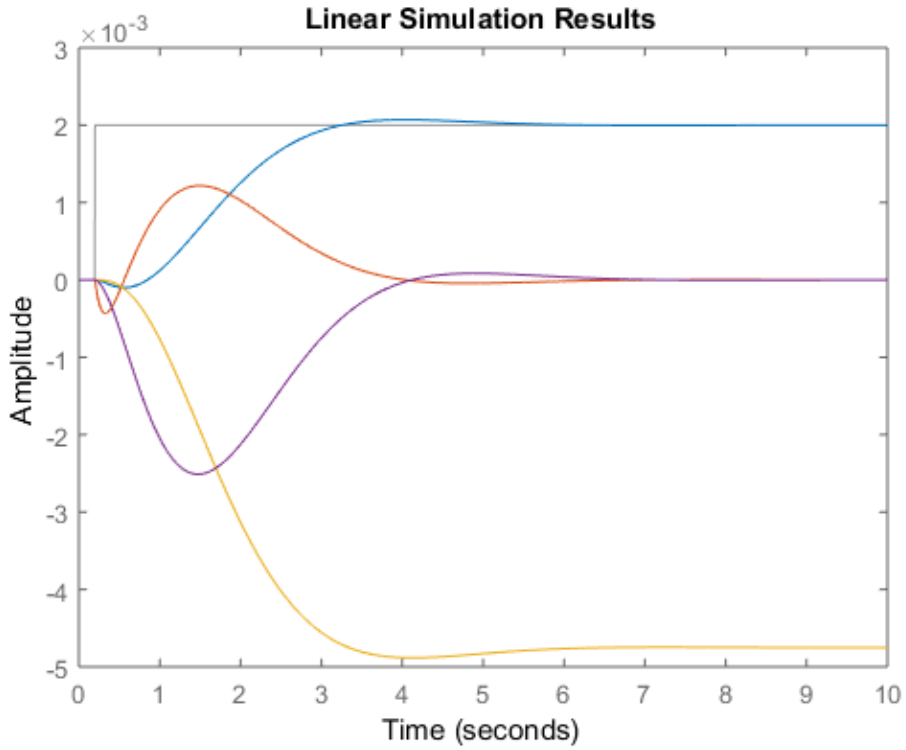
Part C - 2.0

```
                    /---------\
r         +     u   | .       |
---> kr -->() ----->| X=Ax+Bu |--> y=Cx ---> y
          -|        \---------/
           |             |
           |<---- k <----|
```

insert a reference for state (delta)

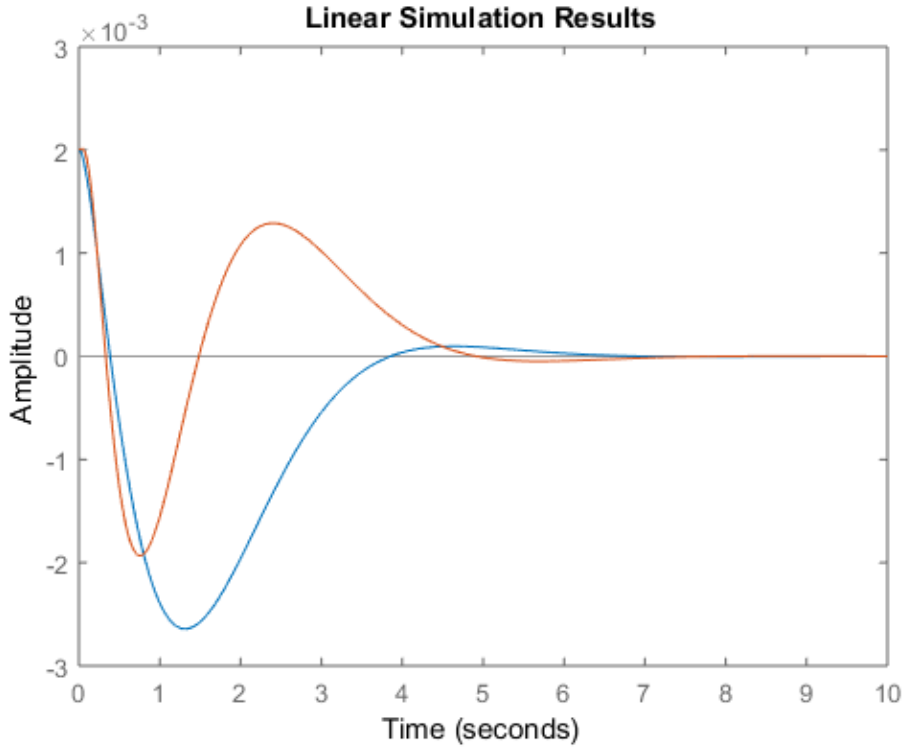
```
% x_d = (A-Bk)*x + (kr*B)*r
% x_d = A_star*x + B_star*r
% where r is a reference step input for delta

A_star = A-B*k;
C = [0,0,1,0];                              %design kr for only delta on output
D = 0;
kr = -(C*(A-B*k)^-1 *B)^-1;%-(rscale(ss(A-B*k,B,C,D),k) - 1.68);     %equal to -0.541 (rscale calculates Kr)
B_star = kr*B;

% apply a step input with size of 0.002, plot response of each state
% using the step() function
t = (0:0.005:10);                       % time set
figure(6); hold on;
sys_delta = ss(A_star,B_star,C,D);
sys_gamma = ss(A_star,B_star,[1,0,0,0],D);    % response on gamma
sys_gamma_d = ss(A_star,B_star,[0,1,0,0],D);  % response on gamma_d
sys_delta_d = ss(A_star,B_star,[0,0,0,1],D);  % response on delta_d

lsim(sys_delta, u, t);
lsim(sys_delta_d, u, t);
lsim(sys_gamma, u, t);
lsim(sys_gamma_d, u, t);
% lsim(syslqrcl, u, t);

hold off;
%Nbar = rscale(ss(A-B*k,B,C,D),k)
```

```
%aa = ss(A-B*k, -(Nbar-1.68)*B, C, D)
%figure(7); lsim(aa, u, t)
```



Part D

```
P = [-8 -16 -4+1i -4-1i];           % pole locations for observer
L = place(A', C', P)';              % poles at eig(A-LC);

% state space model for complete state-feeback and observer

A_p = [A -B*k; L*C A-B*k-L*C];
B_p = [B*kr; B*kr];
C_p = [C zeros(size(C))];
C_p_hat = [zeros(size(C)) C];

sys_p = ss(A_star, B_star, C, D);           % system which outputs plant delta
sys_p_hat = ss(A_p, B_p, C_p, 0);       % system which outputs observer delta

figure(8); hold on;
no_input = zeros(size(t));              % no step input

%show response for initial disturbance on plant delta
lsim(sys_p, no_input, t, [0, 0, 0.002, 0]);
lsim(sys_p_hat, no_input, t, [0, 0, 0.002, 0, zeros(1,4)]);
hold off;
```

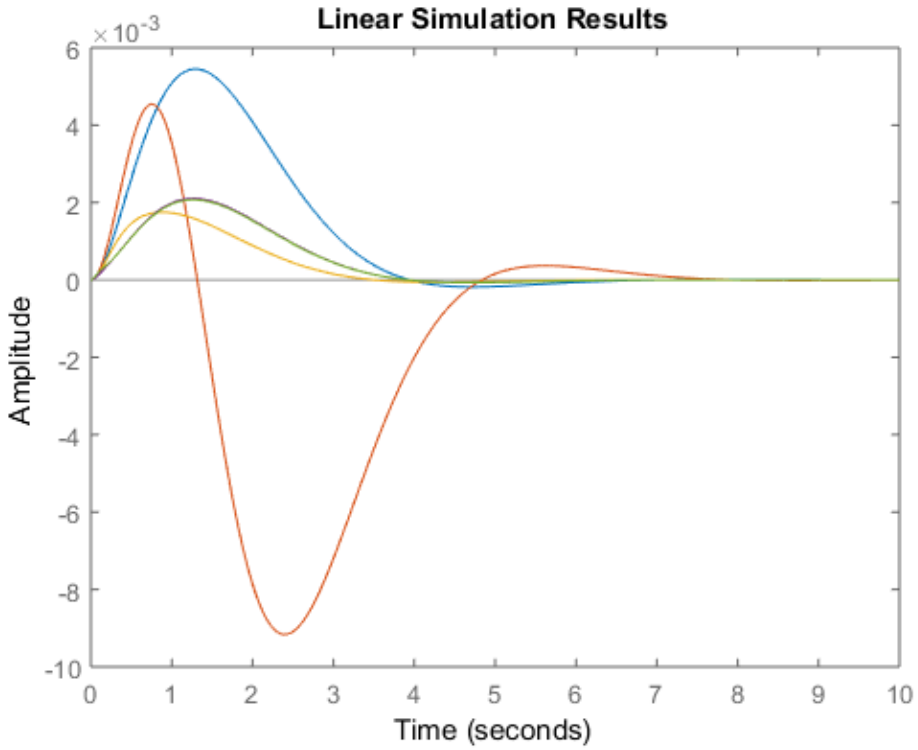


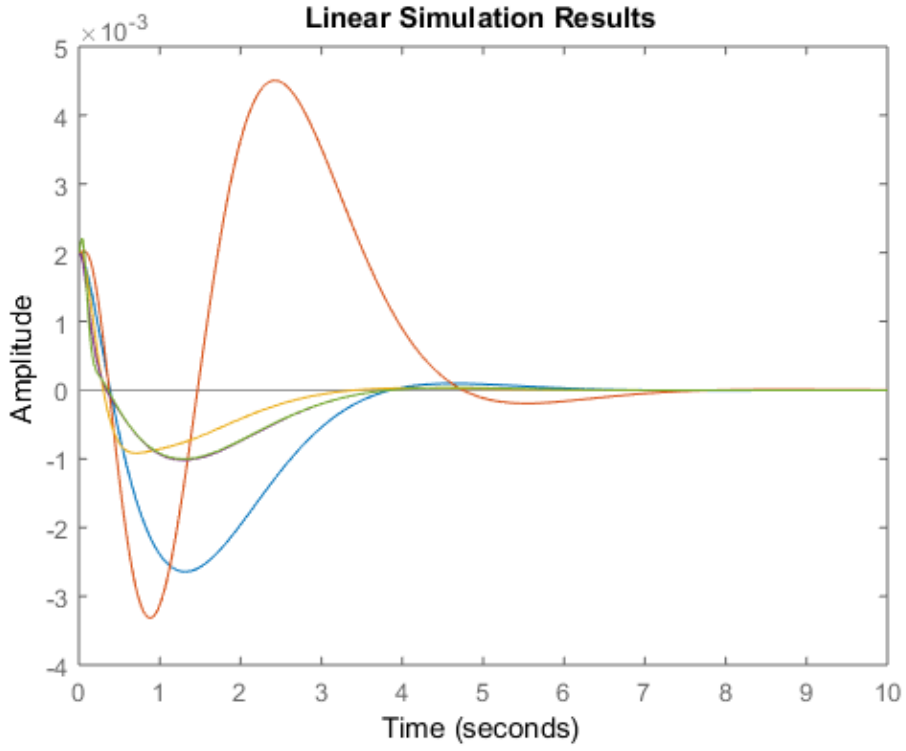
Part E

to show the effect of observer poles on replicating the plant Pol is a matrix containing the poles of the state-feedback component of the plant

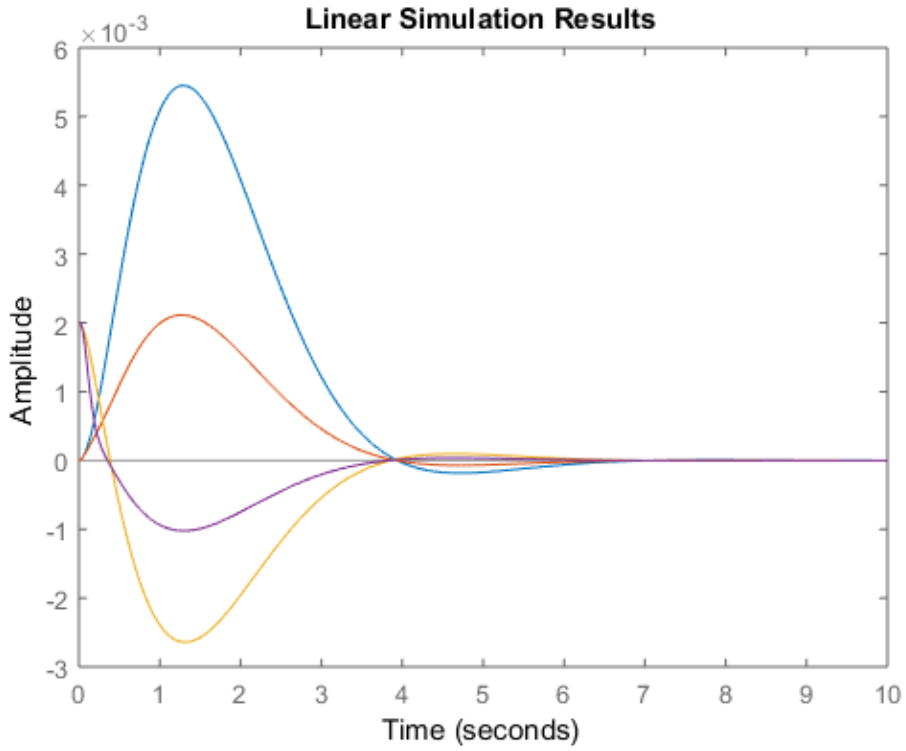
```
% make the observer poles a multiple of the plant observes, my multiplying
% Pol by an element in co before evaluating L.
co = [2, 5, 10, 20];

% For each multiple of Pol, plot the response against the plant for an
% initial state offset -> delta by 0.002

for i = 1:4
    % outputing delta
    C = [0,0,1,0];
    % place poles of observer at a multiple of Pol
    L = place(A', C', co(i)*Pol)';              % poles at eig(A-LC);
    A_p = [A -B*k; L*C A-B*k-L*C];              % observer A
    B_p = [B*kr; B*kr];                         % observer B
    C_p = [C zeros(size(C))];                   % observer C

    % retrieve the state-space model for an output of delta only
    if i == 1
        sys_delta = ss(A_star, B_star, C, D);        % system which outputs plant delta
    end
    sys_delta_hat = ss(A_p, B_p, C_p, 0);        % system which outputs observer delta
    % retrieve the state-space model for an output of phi
    C = [1,0,0,0];
    C_p_hat = [ C, zeros(size(C))];
    if i == 1
        sys_phi = ss(A_star, B_star, C, D);        % system which outputs plant delta
    end
    sys_phi_hat = ss(A_p, B_p, C_p_hat, 0);        % system which outputs observer delta
    figure(9); hold on;
    % plot response on phi
    if i == 1
        % plot the plant response, only once
        lsim(sys_phi, no_input, t, [0, 0, 0.002, 0]);
    end
    lsim(sys_phi_hat, no_input, t, [0, 0, 0.002, 0, zeros(1,4)]);
```

```
    hold off;
    figure(10); hold on;
    % plot reponse on delta
    if i == 1
        % plot the plant response, only once
        lsim(sys_delta, no_input, t, [0, 0, 0.002, 0]);
    end
    lsim(sys_delta_hat, no_input, t, [0, 0, 0.002, 0, zeros(1,4)]);
    hold off;

end
```





```
i = 1;
    % outputing delta
    C = [0,0,1,0];
    % place poles of observer at a multiple of Pol
    L = place(A', C', [-15+7.5i -15-7.5i -20+10i -20-10i])';              % poles at eig(A-LC);
    A_p = [A -B*k; L*C A-B*k-L*C];              % observer A
    B_p = [B*kr; B*kr];                         % observer B
    C_p = [C zeros(size(C))];                   % observer C

    % retrieve the state-space model for an output of delta only
    if i == 1
        sys_delta = ss(A_star, B_star, C, D);        % system which outputs plant delta
    end
    sys_delta_hat = ss(A_p, B_p, C_p, 0);        % system which outputs observer delta
    % retrieve the state-space model for an output of phi
    C = [1,0,0,0];
    C_p_hat = [ C, zeros(size(C))];
    if i == 1
        sys_phi = ss(A_star, B_star, C, D);        % system which outputs plant delta
    end
    sys_phi_hat = ss(A_p, B_p, C_p_hat, 0);        % system which outputs observer delta
    figure(11); hold on;
    % plot response on phi
    if i == 1
        % plot the plant response, only once
        lsim(sys_phi, no_input, t, [0, 0, 0.002, 0]);
    end
    lsim(sys_phi_hat, no_input, t, [0, 0, 0.002, 0, zeros(1,4)]);
    hold off;
    figure(11); hold on;
    % plot reponse on delta
    if i == 1
        % plot the plant response, only once
        lsim(sys_delta, no_input, t, [0, 0, 0.002, 0]);
    end
    lsim(sys_delta_hat, no_input, t, [0, 0, 0.002, 0, zeros(1,4)]);
    hold off;
```



Published with MATLAB® R2015a