E102 Inverted Pendulum Project

Felipe Borja[1], Casey Gardner[1]

I. Introduction

The state space control of a simple inverted pendulum is a dynamic problem with multiple applications as a model, and is thus well studied. In this paper, we generalize this model by introducing a translating base and integral action to account for the effect of a disturbance on the pendulum itself. Fig. 1 represents this model in its entirety (as well as defines specific constraints for response time and distance travelled). In this case, the base must move 1 meter within 10 seconds, without overshooting its target.

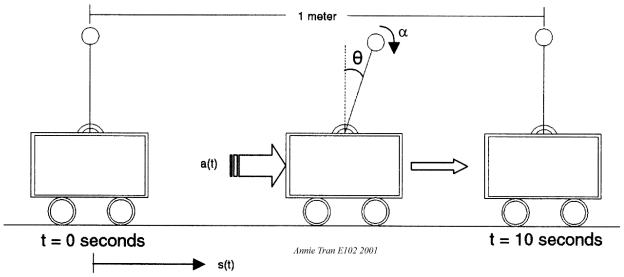

Fig. 1: Diagram of Inverted Pendulum on Cart

II. State Space Design of a Linear Controller for a Linearized Plant

A. Linearized State Space Equations

The governing equations for the system are given by Eqs. 1 and 2. For the purpose of this paper, $s(t)$ refers to the position of the cart, while $\theta(t)$ refers to the angular displacement of the inverted pendulum. An angular acceleration disturbance, $\alpha(t)$ is also incorporated.

$$\frac{d^2\theta}{dt^2} = \frac{g}{L}\sin\theta(t) - \frac{a(t)}{L}\cos\theta(t) + \alpha(t) \tag{1}$$

$$\frac{d^2s(t)}{dt^2} = a(t) \tag{2}$$

[1]Felipe Borja and Casey Gardner are both senior Engineering majors at Harvey Mudd College in Claremont, CA.

Using small angle (linear) approximations, the system can also be expressed in state space matrix form as $\dot{\mathbf{x}} = A\mathbf{x} + Bu$ and $y = C\mathbf{x} + Du$, as in Eqs. 3 and 4:

$$\begin{bmatrix}\dot{\theta}\\ \ddot{\theta}\\ \dot{s}\\ \ddot{s}\end{bmatrix} = \begin{bmatrix}0 & 1 & 0 & 0\\ \frac{g}{L} & 0 & 0 & 0\\ 0 & 0 & 0 & 1\\ 0 & 0 & 0 & 0\end{bmatrix}\begin{bmatrix}\theta\\ \dot{\theta}\\ s\\ \dot{s}\end{bmatrix} + \begin{bmatrix}0\\ -\frac{1}{L}\\ 0\\ 1\end{bmatrix}\ddot{s} \tag{3}$$

$$\begin{bmatrix}\theta\\ s\end{bmatrix} = \begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 0 & 1 & 0\end{bmatrix}\begin{bmatrix}\theta\\ \dot{\theta}\\ s\\ \dot{s}\end{bmatrix} + \begin{bmatrix}0\\ 0\end{bmatrix}\ddot{s} \tag{4}$$

B. Stability, Controllability, and Observability

Using Eqs. 3 and 4, we can establish the stability, controllability, and observability of the linearized system. A causal continuous LTI system is stable if the real parts of the eigenvalues of the system matrix **A** are all less than zero. The linearized system is unstable, with eigenvalues of $s_{1,2} = \pm 4.4272, s_{3,4} = 0$.

A system is controllable if the system matrix **A** can be transformed into control canonical form, which is only possible when the controllability matrix M_c (as seen in Eq. 5) has full rank n; in this case, $n = 4$, so the linearized system is controllable.

$$M_c = \begin{bmatrix}B & AB & A^2B & A^3B\end{bmatrix} \tag{5}$$

A system is observable if the system matrix **A** can be transformed into observer canonical form, which is only possible when the observability matrix M_o (as seen in Eq. 6) has full rank n; in this case, $n = 4$, so the linearized system is also observable.

$$M_o = \begin{bmatrix}C\\ CA\\ CA^2\\ CA^3\end{bmatrix} \tag{6}$$

C. Feedback Control with Integral Action

We begin by implementing integral action control on the translation of the cart using pole placement design. Although the disturbance is an angular acceleration acting on the pendulum, the cart is what we hope to

control. The first step of this process is to augment the original state vector with the integral of the error, as seen in Eq. 7:

$$\dot{\xi} = \begin{bmatrix} x_I \\ \mathbf{x} \end{bmatrix} \tag{7}$$

Next, we create the augmented matrices that correspond to Eq. 7, and are shown in Eqs. 8, 9, 10, and 11.

$$\mathcal{A} = \begin{bmatrix} 0 & \textbf{-C} \\ \mathbf{0} & \mathbf{A} \end{bmatrix} \tag{8}$$

$$\mathcal{B} = \begin{bmatrix} -D \\ \mathbf{B} \end{bmatrix} \tag{9}$$

$$\mathcal{B}_r = \begin{bmatrix} 1 \\ \mathbf{0} \end{bmatrix} \tag{10}$$

$$\mathcal{B}_w = \begin{bmatrix} -D \\ \mathbf{B} \end{bmatrix} \tag{11}$$

Our approach to pole placement was to assume that the behavior of the overall system would be dominated by two "slower" poles, and essentially design a canonical 2nd-order system (In total, we will eventually need to place five poles: four system poles and an additional one due to the integral controller, called K_i). Given the "no-overshoot" and "10 second response" specifications discussed in Section I, and after some fine tuning, we found that the system parameters $\omega_n = 0.8$ rad/s and $\zeta = 1.01$ gave the desired response. This corresponded to poles of: $s_1 = -0.6946, s_2 = -0.9214$. In order to ensure the other two system poles would not contribute significantly to the overall response, we multiplied the first two poles by 5 (a constant found through tuning), to get $s_3 = -3.4729, s_4 = -4.6071$. Lastly, we set $s_{K_i} = -8.5$ to make this the fastest pole of all.

Since the original system's poles led to unstable behavior, we placed our new poles using MATLAB's $acker$ function. The $acker$ function took as inputs $\mathcal{A}$, $\mathcal{B}$, and a row vector containing our five chosen poles; it output an augmented version of **K**, our linear controller. The first element of this vector is K_i and the remaining elements make up **K**. Once we have K_i and **K** we can calculate our feedback matrix $\mathcal{A}_f$ as seen in Eq. 12

$$\mathcal{A}_f = \begin{bmatrix} -DK_i & \textbf{-C} + D\mathbf{K} \\ \mathbf{B}K_i & \mathbf{A} - \mathbf{BK} \end{bmatrix} \tag{12}$$

D. Observer Design

The next step is to design an observer. We found the poles of our observer gain vector **L** by multiplying the poles of our original system by a constant, to speed them up. In our case, we used a factor of 50 to choose our observer poles, which allows the observer to quickly respond to disturbances (differences between the expected output and the measured output). The observer gain, **L**, was found using MATLAB's $place$ function. This function took in A^T, C^T, and the chosen observer poles as inputs and output **L**. Ideally, the poles of the observer would be very fast, even several orders of magnitude faster than the controller poles; however, during physical implementation, this becomes impractical because it requires increased computational effort from extremely quick sampling and calculation.

III. Simulation of Control of a Nonlinear Plant with a Linear Controller

MATLAB and Simulink (developed by MathWorks) were used to simulate the fully controlled and observed nonlinear system, with both the inverted pendulum and the cart. All figures referenced in the following section can be found in Section VII. Fig. 5 is the full system modeled in Simulink. Note the subsystem labeled **Cart Pendulum** in Fig. 5, this represents the nonlinear cart-pendulum plant shown in Fig. 6. Note the other subsystem, labeled **Observer** in Fig. 5; this represents the observer and is shown in Fig. 7.

The variables K_i, **K**, the outputs in Fig. 5 and the outputs in Fig. 6 are the same as those discussed in the previous sections. Additionally, the constants $g = 9.8$ m/s^2 and $L_{pendulum} = 0.5$ m were also incorporated. A disturbance block, labelled *alpha* and a saturation block were also used to fully model the system. The latter was used to approximate the maximum acceleration of the cart. Lastly, as this is a nonlinear system that was designed assuming linearization, some amount of fine-tuning through trial-and-error was required to reach the values (of poles, constant factors, etc.) discussed above.

The system was simulated for a disturbance of $\alpha = 0.5$ rad/s^2, and also for larger disturbances to determine the maximum allowable disturbance before the system became unstable.

IV. Discussion and Conclusions

Fig. 2 shows the response of the nonlinear system to an $\alpha(t) = 0.5$ rad/s^2 disturbance. Note how the cart is able to adapt to the disturbance, and recover enough to reach a final steady state position 1 meter away. It does

so in less than 10 seconds, and without overshooting the desired position, thus fulfilling the specifications. Furthermore, this does not require a saturation of the acceleration, meaning that the system can handle larger disturbances.

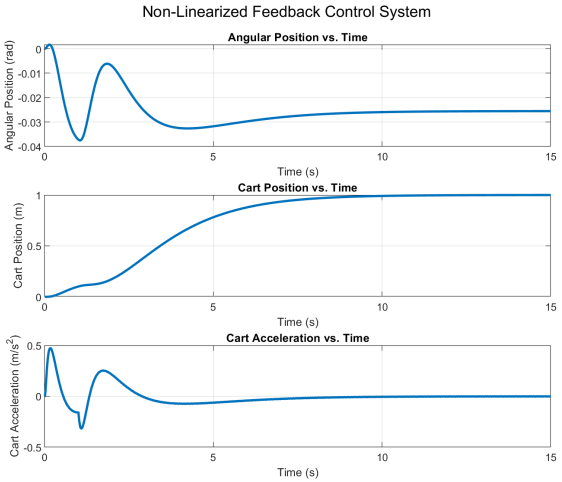

Fig. 2: Plots for angular position, cart position, and cart acceleration in response to an external disturbance of $\alpha(t) = 0.5$ rad/s^2

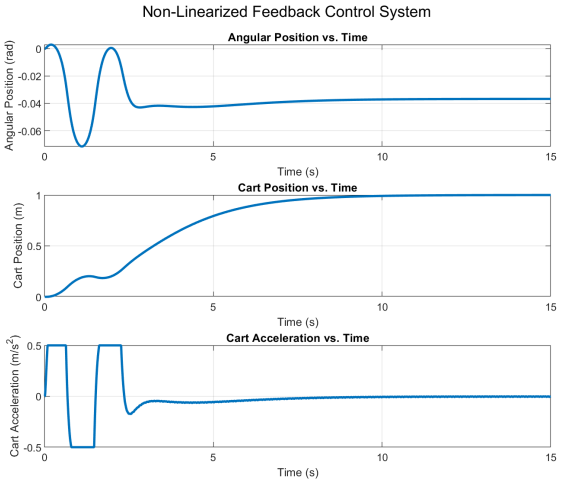

Fig. 3: Plots for angular position, cart position, and cart acceleration in response to an external disturbance of $\alpha(t) = 0.72$ rad/s^2

Fig. 3 shows such a (larger) disturbance. By incrementing the disturbance, we found that $\alpha(t) = 0.72$ rad/s^2 was the maximum this system, could handle and remain stable, as it is currently designed and simulated. Note the larger swing of the angular position of the pendulum, and the highly saturated acceleration of the cart over time. Both of these represent an increasingly unstable system. Despite these differences, the path taken by the cart remains surprisingly similar as a function of time.

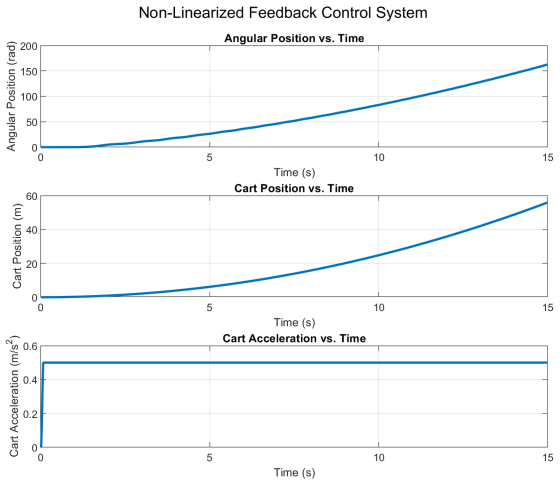

Fig. 4: Plots for angular position, cart position, and cart acceleration in response to an external disturbance of $\alpha(t) = 0.9$ rad/s^2

Finally, Fig. 4 shows the response to an even larger disturbance ($\alpha(t) = 0.9$ rad/s^2), one that makes the system wholly unstable. The cart accelerates to saturation immediately, but is unable to counterbalance the motion of the pendulum, which begins accelerating exponentially. In an effort to control this, the cart continues to accelerate, and thus it is no surprise that after 10 seconds, it has far overshot the desired position, with no signs of stopping.

In conclusion, these results make clear that we have been able to analyze the stability, controllability, and observability of a nonlinear system and design both an integral action linear controller and an observer to meet certain specifications. The inverted-cart pendulum, as mentioned at the beginning of this paper, can serve as a potential model for many systems; the present effort contributes to the body of control engineering literature that can be used in such applications.

V. Acknowledgements

We would like to thank Prof. Qimin Yang, Prof. Timothy Tsai, the Tau-Beta-Pi tutors, and our fellow students who helped us through the assignment. We would also like to thank Prof. Anthony Bright for his E102 Modern Control Engineering Class Notes.

We sampled heavily from these notes, both in our procedure and in our writeup.

VI. Appendix: MATLAB Code

A. *inverted_pendulum_final.m*

```
% inverted_pendulum_final.m
% Felipe Borja and Casey Gardner
% E102 Midterm Project, 23 April 2019

%clear

%% Part 0: Defining Constants
% Define Constants
L = 0.5;          % in m
g = 9.8;          % in m/s^2
alpha = 0.5;      % in rad/s^2
ICo = 0;          % observer initial condition

wn = 0.8;         % nat frequency of secondOrderSys
zeta = 1.01;      % damping ratio of secondOrderSys
pfactor = 5;      % factor for other two system poles
poleki = -8.5;    % integral controller pole

%% Part 1: Defining the State Space Matrices
% Define State-space matrices
A = [0 1 0 0;(g/L) 0 0 0; 0 0 0 1; 0 0 0 0];
B = [0; -(1/L); 0; 1];
C = [1 0 0 0; 0 0 1 0];
D = [0; 0];

%% Part 2: Establish stability, controllability, observability
disp(' ')
A_eig = eig(A);
% stable if eigenvalues < 0
if (max(A_eig) <= 0); disp('Original system is stable')
else; disp('Original system is UNstable')
end

Mc = [B (A*B) (A^2*B) (A^3*B)]; % controllability matrix
%controllable if full rank
if(rank(Mc) == size(Mc,2)); disp('System is controllable');
else; disp('System is NOT controllable')
end

Mo = [C; (C*A); (C*A^2); (C*A^3)]; % observability matrix
% observable if full rank
if (rank(Mo)==size(Mo,2)); disp('System is observable');
else; disp('System is NOT observable')
end

%% Part 3.1: Design integral action sys with feedback
% We only need integral control on translational displacement
%       disturbance is angular, but we are controlling translation
%       so we take the bottom row of C and D
C_bot = C(2,:);
D_bot = D(2,:);
% Calculate augmented matrices for integral control
A_aug = [0 -C_bot; zeros(4,1) A];
B_aug = [-D_bot; B];
Br_aug = [1; zeros(4,1)];
Bw_aug = B_aug;
C_aug = [zeros(2,1) C];

% We assume that the behavior of the system is dominated by the
%       behavior of two secondOrderSys poles, sop1 and sop2
[ sop1, sop2, so_respInfo ] = secondOrderStep( wn, zeta, 0 );
% Define poles to place, pki = poles that include ki
pki = [poleki sop1 sop2 pfactor*sop1 pfactor*sop2];
% pki = [-8.5000   -0.6946   -0.9214   -3.4729   -4.6071];
% Extract 4 "system" poles (not poleki) for later evaluation
p1 = pki(1,2);
p2 = pki(1,3);
p3 = pki(1,4);
p4 = pki(1,5);

% Use pole placement to find k_i and k_bar
kpoles = acker(A_aug, B_aug, pki);
% Extract integral controller gain and controller vector
ki = kpoles(1);
kbar = kpoles(2:end);

%{
% Define feedback matrix, to check correct pole placement
A_f = [-D_bot*ki    -C_bot + D_bot*kbar; ...
        B*ki                A-B*kbar ];
% Check the stability of feedback matrix
newpol = eig(A_f);
if (max(real(newpol)) <= 0); disp('Feedback system is stable')
else; disp('Feedback system is UNstable')
end
%}
disp(' ') % nice printing space

%{
% Plot the step response for four chosen "system" poles
%       note that only the "settling time" is accurate
%       to predict full behavior, we'd need the numerator of the TF
s = tf('s');
tcl = 1/( (s-p1)*(s-p2)*(s-p3)*(s-p4) );
[y, t] = step(tcl);
figure(1)
plot(t,y)
xlabel('Time (s)')
ylabel('Unit Step Response')
title(sprintf('Step Response with Chosen Poles: [%.2f, %.2f, %.2f,
    %.2f]',...
    p1, p2, p3, p4))
grid on
stats = stepinfo(tcl, 'SettlingTimeThreshold', 0.001);
%}

%% Part 3.2: Design observer
% Observer poles are much faster than controller poles,
%       not including pole corresponding to ki
% pe = poles of l_bar
pe = 50*pki(1,2:end);
lbar = place(A', C', pe)';

tspan = 15; % how long to simulate in Simulink

%% Non-linear Simulink Simulation

disp('Simulating nonlinear system...')
sim('pendulum_control_final', tspan)
figure(3)
clf
subplot(3,1,1)
plot(tout, yout.signals(1).values', 'LineWidth', 2)
title('Angular Position vs. Time')
xlabel('Time (s)')
ylabel('Angular Position (rad)')
grid on

subplot(3,1,2)
plot(tout, yout.signals(2).values', 'LineWidth', 2)
title('Cart Position vs. Time')
xlabel('Time (s)')
ylabel('Cart Position (m)')
grid on

subplot(3,1,3)
plot(tout, yout.signals(3).values', 'LineWidth', 2)
title('Cart Acceleration vs. Time')
xlabel('Time (s)')
ylabel('Cart Acceleration (m/s^2)')
grid on
sgtitle({'Non-Linearized Feedback Control System'})
set(gcf, 'color', 'w')

disp(' ') % nice printing space
```

B. *secondOrderStep.m*

```
function [ p1, p2, respInfo ] = secondOrderStep( wn, z, plotStep )
%secondOrderStep returns the two poles and the results of stepinfo()
      given
%the natural frequency (rad/s) and damping ratio (zeta) of a
      canonical
%second order system
%   plotStep = 1 also plots the response of this system (else = no
      plot)
    %define the system
    s = tf('s');
    num = wn^2;
    denom = s^2 + 2*z*wn*s + wn^2;
    tcl = num/denom;

    if plotStep == 1
        figure(10)
        clf
        [y, t] = step(tcl);
        plot(t,y)
        xlabel('Time (s)')
        ylabel('Unit Step Response')
        title(sprintf('Step Response with wn = %.2f, zeta = %.2f'
      ,...
            wn, z))
        grid on
    end
    respInfo = stepinfo(tcl, 'SettlingTimeThreshold', 0.01);

    %find the pole values, using quadratic formula
    a = 1;
    b = 2*z*wn;
    c = wn^2;
    p1 = (-b + sqrt(b^2 - 4*a*c)) / (2*a);
    p2 = (-b - sqrt(b^2 - 4*a*c)) / (2*a);

end
```

VII. Appendix: Simulink Diagrams

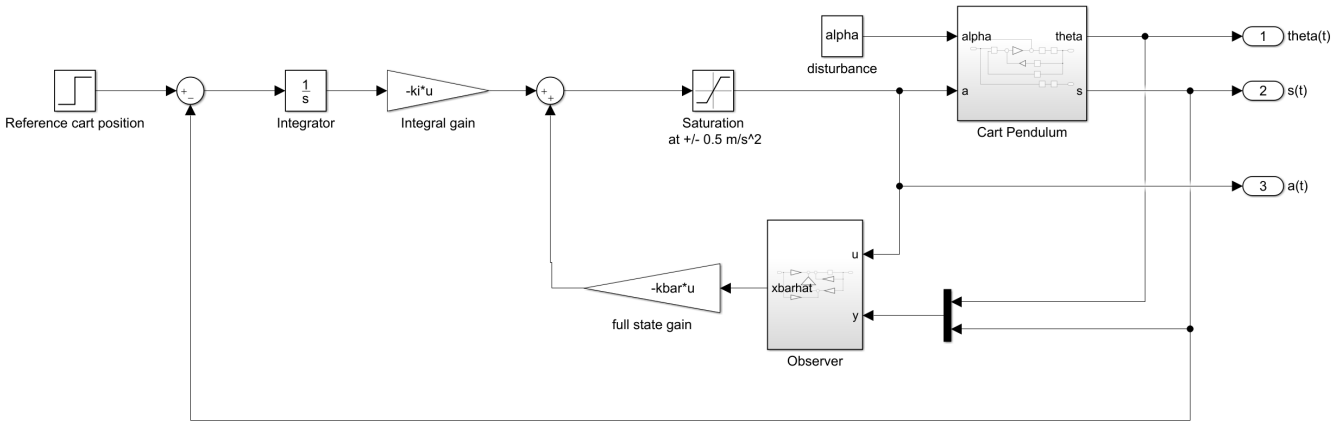

Fig. 5: Simulink model of linearized feedback control system for cart-pendulum system.

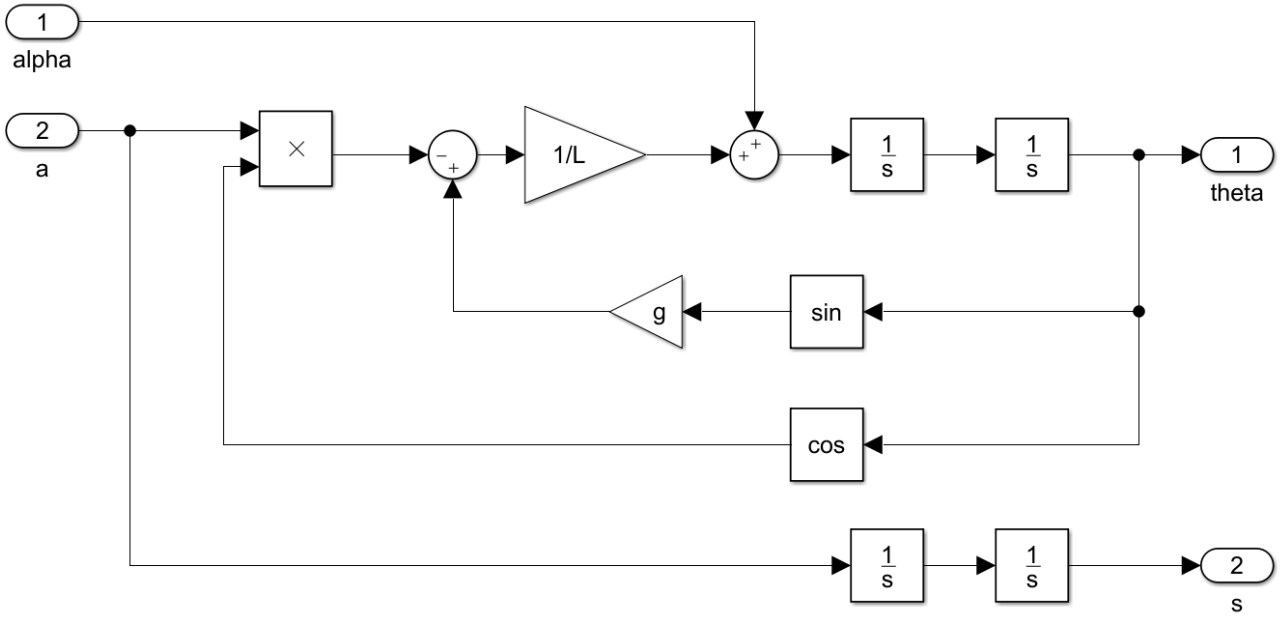

Fig. 6: Simulink model of plant for cart-pendulum system.

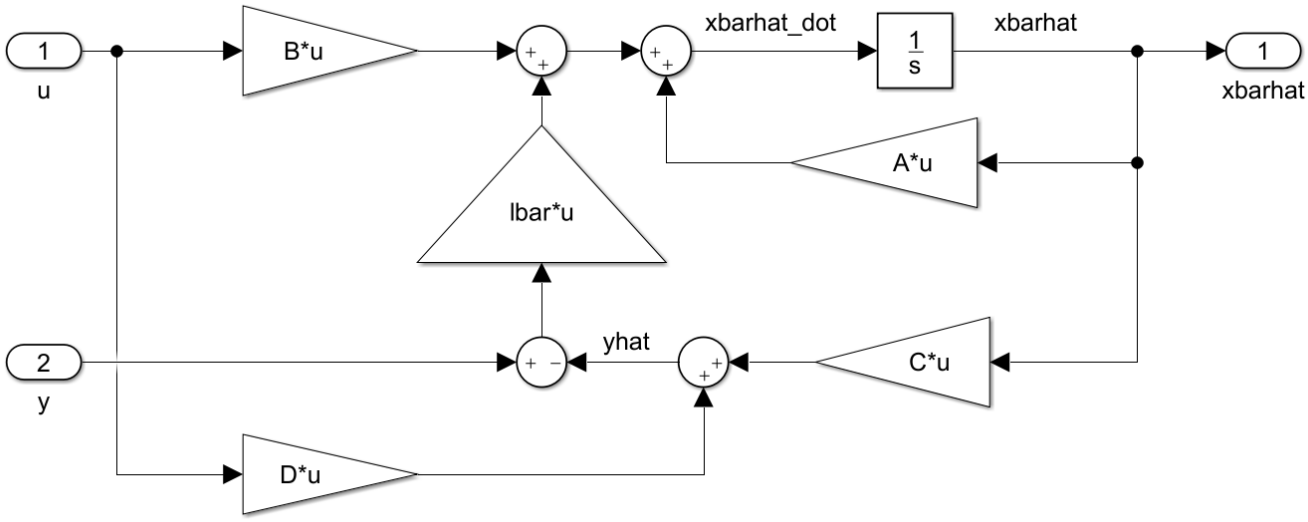

Fig. 7: Simulink model of observer for linearized feedback control system.