Example - Observer Based Design

Consider a system described by the differential equation

$$\ddot{y} = u \qquad \text{(Double integrator)}$$

To put this system into state space form let

$$\begin{aligned} x_1 &= y \\ x_2 &= \dot{y} \end{aligned} \Rightarrow \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} \dot{y} \\ \ddot{y} \end{pmatrix} = \begin{pmatrix} x_2 \\ u \end{pmatrix}$$

$$\therefore \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \end{pmatrix} u$$

$$y = (1 \;\; 0) \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

where $A = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}$, $B = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$, $C = (1 \;\; 0)$, $D = 0$

Suppose that the desired closed loop response of the output y to a reference input r is specified by the desired control characteristic equation

$$s^2 + 2\xi\omega_n s + \omega_n^2 = 0$$

where $\omega_n = 1$, $\xi = 0.707$.

Find the feedback gains K s.t. the eig $(A-BK)$ are at the roots of the desired control char equation.

The control gains are given by

$$K = (\bar{\alpha} - \bar{a})\,\mathcal{Q}^{-1}\mathcal{C}^{-1}$$

where $\bar{\alpha} = (1.414, 1)$ and where

$$\det(sI - A) = \det\begin{pmatrix} s & -1 \\ 0 & s \end{pmatrix} = s^2$$ implies that

$$\bar{a} = (0 \quad 0)$$

and $\mathcal{A} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \Rightarrow \mathcal{A}^{-1} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$

and where $\mathcal{C} = (B \quad AB) = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \Rightarrow \mathcal{C}^{-1} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$

$$\therefore \quad K = \left[(1.414 \quad 1) - (0 \quad 0)\right]\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

$$= \boxed{(1 \quad 1.414)}$$

The associated feedforward gain is

$$k_r = \frac{-1}{C(A-BK)^{-1}B} = \frac{-1}{(1 \quad 0)\left(\begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} - \begin{pmatrix} 0 \\ 1 \end{pmatrix}(1 \quad 1.414)\right)^{-1}\begin{pmatrix} 0 \\ 1 \end{pmatrix}}$$

$$= \frac{-1}{(1 \quad 0)\begin{pmatrix} 0 & 1 \\ -1 & -1.414 \end{pmatrix}^{-1}\begin{pmatrix} 0 \\ 1 \end{pmatrix}} = \frac{-1}{(1 \quad 0)\begin{pmatrix} -1.414 & -1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} 0 \\ 1 \end{pmatrix}} = \frac{-1}{(-1.414 \quad -1)\begin{pmatrix} 0 \\ 1 \end{pmatrix}}$$

$$= 1$$

In Matlab we could have used

```
AP.A = [0, 1; 0, 0];
AP.B = [0; 1];
AP.C = [1, 0];
AP.D = 0;

% pick desired char equation for system response
wn = 1;
zeta = 0.707;
control_poles = roots([1,2*zeta*wn,wn^2]);

% place control poles using feedback gain K
P.K = place(P.A,P.B,control_poles);

% feedforward gain
P.kr = -1/(P.C*inv(P.A-P.B*P.K)*P.B);
```

Suppose now that we want to place the poles of the observer such that the char eq of the observation error is governed by

$$s^2 + 2\zeta_o \omega_{n_o} s + \omega_{n_o}^2 = 0$$

where $\zeta_o = 0.707$, $\omega_{n_o} = 10$,

Then the desired char equation of the observation error is

$$s^2 + 14.14\, s + 100 = 0$$

The observer gains are given by

$$L = \mathcal{O}^{-1} \mathcal{C}^{-T} (\alpha_o - \bar{a})^T$$

where $\alpha_o = (14.14 \quad 100)$ and

$$\mathcal{O} = \begin{pmatrix} C \\ CA \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \Rightarrow \mathcal{O}^{-1} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

$$\therefore \boxed{L = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \left(\begin{pmatrix} 14.14 \\ 100 \end{pmatrix} - \begin{pmatrix} 0 \\ 0 \end{pmatrix} \right) = \begin{pmatrix} 14.14 \\ 100 \end{pmatrix}}$$

In Matlab, we could have used

```
% pick desired char equation for observation error
wn = 10;
zeta = 0.707;
observer_poles = roots([1,2*zeta*wn,wn^2]);

% place observer poles using observer gain L
P.L = place(P.A',P.C',observer_poles)';
```

The observer based controller is given by the equations

$$\dot{\hat{x}} = A\hat{x} + Bu + L(y - C\hat{x}) \quad (4.1)$$

$$u = -K\hat{x} + k_r r \quad (4.2)$$

The following Matlab code implements this controller

```
function u=ctrl(in,P)
    y_d    = in(1);
    y      = in(2);
    t      = in(3);
    x      = in(4:5);

    % define and initialize persistent variables
(1) persistent xhat_        % estimated state (for observer)
    persistent u            % delayed input (for observer)

    N = 10; % number of integration steps for each sample

    % initialize persistent variables
    if t==0,
       xhat_  = zeros(2,1);   (2)
       u      = 0;
    end
    % solve observer differential equations
(3) for i=1:N,
        xhat_ = xhat_ + P.Ts/N*(P.A*xhat_ + P.B*u + P.L*(y-P.C*xhat_));
    end
    xhat = xhat_(1:2);
    % feedback controller
(4) u = P.kr*y_d - P.K*xhat;

end
```

(1) We will need to retain in memory both the state xhat and the old control variable u

(2) The observer is initialized as $\hat{x}(0) = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$.

The input at time $t=0$ is initialized to be

$u(0) = 0$

(3) Solution to the differential equation

$$\dot{\hat{x}} = A\hat{x} + Bu + L(y - C\hat{x}) \quad (5.1)$$

$\dot{\hat{x}}$ is approximated as

$$\dot{\hat{x}} \simeq \frac{\hat{x}_K - \hat{x}_{K-1}}{T}$$

where T is the time between $K-1^{th}$ and K^{th} samples

$\therefore$ (5.1) becomes

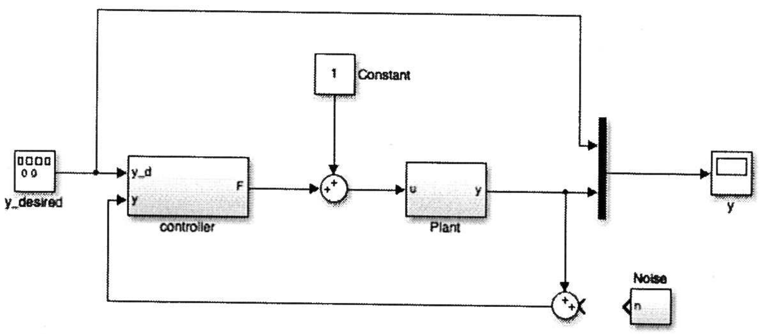
$$\hat{x}_K = \hat{x}_{K-1} + T\left(A\hat{x}_{K-1} + Bu_{K-1} + L\left(y_K - C\hat{x}_{K-1}\right)\right) \quad (5.2)$$

(5.2) is approximation of the true solution of (5.1).
The smaller T, the better the approximation.
Since the control code is called every sample instant with sample rate T_s, we improve the approximation in (3) by increasing the sample rate by $N=10$.

(4) Direct implementation of (4.1)

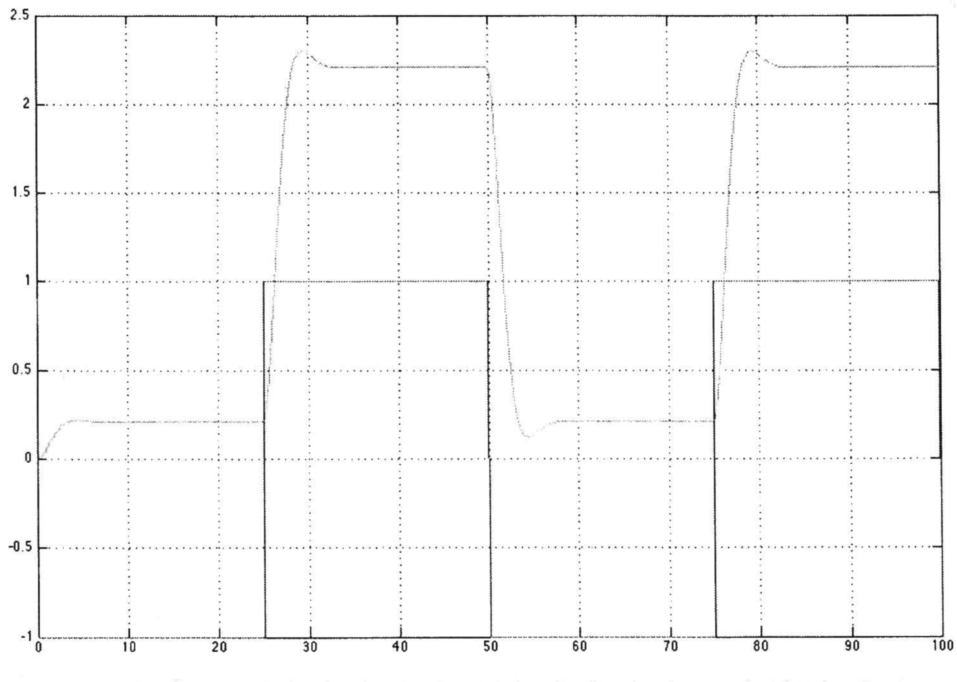
2. Adding an Integrator

Suppose now that there is an input disturbance to the system, like that shown in the simulink diagram below



The input disturbance may be caused by a variety of external physical phenomena and is assumed unknown to the controller.

With the input disturbance, there will be a steady state error in the output as shown below



To remove steady state error we can add an integrator to the controller.

Let $$x_I = \int_{-\infty}^{t} (y(\tau) - r(\tau))\, d\tau$$

$$\Rightarrow \quad \dot{x}_I = y - r = Cx - r$$

Therefore given the original system dynamics

$$\dot{x} = Ax + Bu$$

we augment the state with the integrator state to get

$$x_{int} = \begin{pmatrix} x \\ x_I \end{pmatrix}$$

where

$$\dot{x}_{int} = \begin{pmatrix} \dot{x} \\ \dot{x}_I \end{pmatrix} = \begin{pmatrix} A & 0 \\ C & 0 \end{pmatrix} \begin{pmatrix} x \\ x_I \end{pmatrix} + \begin{pmatrix} B \\ 0 \end{pmatrix} u + \begin{pmatrix} 0 \\ -1 \end{pmatrix} r \qquad (7.1)$$

The controller will be of the form

$$u = -Kx + k_I \int (y - r)\, dt$$

$$= -Kx - k_I x_I$$

and we wish to use the same pole placement tools that we used to design K without the integrator.

Noting that $$u = -\begin{pmatrix} K & k_I \end{pmatrix} \begin{pmatrix} x \\ x_I \end{pmatrix} \triangleq -K_{int} x_{int}$$

If we let $$A_{int} = \begin{pmatrix} A & 0 \\ C & 0 \end{pmatrix}, \quad B_{int} = \begin{pmatrix} B \\ 0 \end{pmatrix}$$

then (7.1) becomes

$$\dot{x}_{int} = A_{int} x_{int} + B_{int} u + \begin{pmatrix} 0 \\ -1 \end{pmatrix} r$$

when $u = -K_{int} X_{int}$ we have

$$\dot{X}_{int} = (A_{int} - B_{int} K_{int}) X_{int} + \begin{pmatrix} 0 \\ -1 \end{pmatrix} r$$

where we wish to select K_{int} to places the poles of the augmented system at specified locations

We can find the new gains by using the matlab code

```
% gains if there is an integrator
integrator_pole = -.5;
P.A_int = [P.A, [0;0]; P.C, 0];
P.B_int = [P.B; 0];
P.C_int = [P.C, 0; ];
P.K_int = place(P.A_int,P.B_int,[control_poles; integrator_pole]);
P.K_wi = P.K_int(1:2);
P.ki = P.K_int(3);
```

The feedback controller with the integrator can be implemented by the following matlab code

```
function u=ctrl(in,P)
    y_d     = in(1);
    y       = in(2);
    t       = in(3);
    x       = in(4:5);
    % define and initialize persistent variables
    persistent xhat_       % estimated state (for observer)
    persistent u           % delayed input (for observer)
    persistent integrator
(1) persistent error_d1

    N = 10; % number of integration steps for each sample
    % initialize persistent variables
    if t==0,
        xhat_   = zeros(2,1);
        u       = 0;
(2)     integrator = 0;
        error_d1   = 0;
    end
    % solve observer differential equations
    for i=1:N,
        xhat_ = xhat_ + P.Ts/N*(P.A*xhat_ + P.B*u + P.L*(y-P.C*xhat_));
    end
    xhat = xhat_(1:2);
    % implement integrator
(3) error = y - y_d;
    integrator = integrator + (P.Ts/2)*(error+error_d1);
    error_d1 = error;
(4) % feedback controller
    u = - P.K_wi*xhat - P.ki*integrator;
end
```

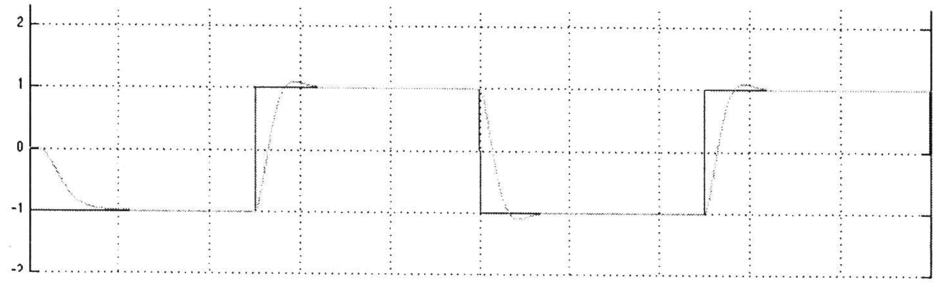
(1) We need two persistent variable associated with the integrator

(2) The integrator and the error, delayed by 1 are initialized to zero

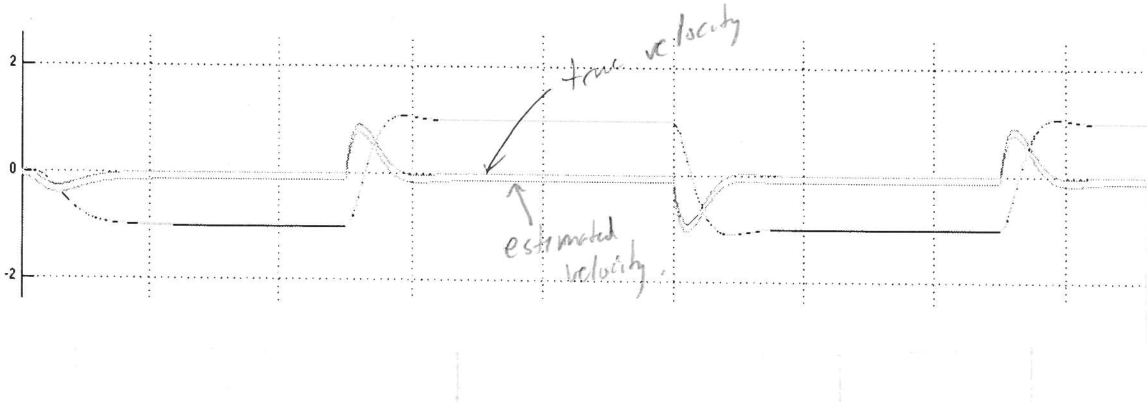
(3) Implementation of the integrator using trapezoidal rule

(4) Implementation of state feedback with integrator.

The output response is shown below where it can be seen that the integrator removes the steady state error in the output



However, if we look at the estimation error, we see a non-zero steady state bias in the estimation error:



The reason for the bias in the estimation error can be understood as follows.

The true system, with the input disturbance is

$$\dot{x} = Ax + B(u+d)$$

the observer is

$$\dot{\hat{x}} + A\hat{x} + Bu + L(y - C\hat{x})$$

Note the absence of d

$\therefore$ the observation error $e_x = x - \hat{x}$ is governed by

$$\dot{e}_x = \dot{x} - \dot{\hat{x}} = Ax + Bu + Bd - A\hat{x} - Bu - LCe_x$$

$$= (A - LC)e_x + \underline{\underline{Bd}}$$

therefore d is a constant input into the estimation error.

To mitigate this effect, we will use the observer to estimate, not only x, but d as well.

Since d is constant we have $\dot{d} = 0$

Therefore the system dynamics are given by

$$\dot{x} = Ax + Bu + Bd$$
$$\dot{d} = 0$$

Letting $x_{dist} \triangleq \begin{pmatrix} x \\ d \end{pmatrix}$ be the augmented state, The system equations become

$$\dot{X}_{dist} = \begin{pmatrix} \dot{x} \\ \dot{d} \end{pmatrix} = \begin{pmatrix} A & B \\ 0 & 0 \end{pmatrix}\begin{pmatrix} x \\ d \end{pmatrix} + \begin{pmatrix} B \\ 0 \end{pmatrix} u$$

Defining $A_{dist} \triangleq \begin{pmatrix} A & B \\ 0 & 0 \end{pmatrix}$, $B_{dist} \triangleq \begin{pmatrix} B \\ 0 \end{pmatrix}$

$C_{dist} \triangleq (C \ \ 0)$

we have

$$\dot{X}_{dist} = A_{dist} X_{dist} + B_{dist} u$$

$$y = C_{dist} X_{dist}$$

We now build an observer for the augmented system as

$$\dot{\hat{X}}_{dist} = A_{dist} \hat{X}_{dist} + B_{dist} u + L_{dist}(y - C_{dist}\hat{X}_{dist})$$

where L_{dist} is selected to place $eig(A_{dist} - L_{dist} C_{dist})$ at specified locations.

The Matlab code to find L_{dist} is

```
% observer gain if estimating the input disturbance
input_disturbance_pole = -1;
P.A_dis = [P.A, P.B; zeros(1,3)];
P.B_dis = [P.B; 0];
P.C_dis = [P.C, 0];
P.L_dis = place(P.A_dis',P.C_dis',[observer_poles;input_disturbance_pole])';
```

After the disturbance is estimated, we can use it in the controller to subtract out the input disturbance:

$$u = -K\hat{x} + k_r r - \hat{d} \qquad (12.1)$$

where $\hat{X}_{dist} = \begin{pmatrix} \hat{x} \\ \hat{d} \end{pmatrix}$

The resulting control code is on the next page

1. The observer used A_{dist}, B_{dist}, C_{dist} instead of A, B, C

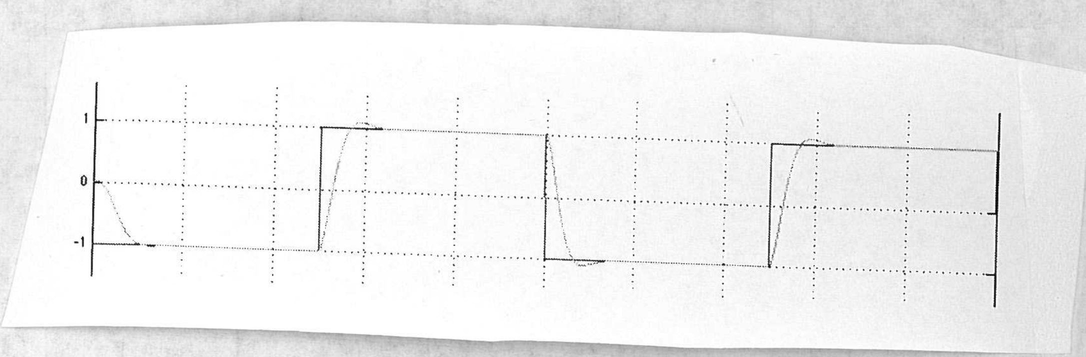
2. $\hat{x}$ is the first two elements of $\hat{x}_{dist}$
$\hat{d}$ is the third element of $\hat{x}_{dist}$

3. controller that uses $\hat{d}$ as in (12.1)

```
function u=ctrl(in,P)
    y_d     = in(1);
    y       = in(2);
    t       = in(3);
    x       = in(4:5);
    % define and initialize persistent variables
    persistent xhat_       % estimated state (for observer)
    persistent u           % delayed input (for observer)

    N = 10; % number of integration steps for each sample
    % initialize persistent variables
    % initialize persistent variables
    if t==0,
       xhat_  = zeros(3,1);
       u      = 0;
    end
    % solve observer differential equations
    for i=1:N,
(1)    xhat_ = xhat_ + P.Ts/N*(P.A_dis*xhat_ + P.B_dis*u + P.L_dis*(y-
P.C_dis*xhat_));
    end
    xhat = xhat_(1:2);
(2) disturbance_estimate = xhat_(3)
    % feedback controller
(3) u = P.kr*y_d - P.K*xhat - disturbance_estimate;
end
```

The output and estimated states are shown below



Note zero tracking error even without an integrator

Note zero bias in the state estimates