AE353: Notes on Disturbance Rejection and Integral Action

(to be treated as an appendix to the presentation)

A. Borum T. Bretl M. Ornik P. Thangeda

1 Integral Action

Consider the state-space system

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx, \end{aligned} \tag{1}$$

where y and u are scalars. Consider controllers that have the form

$$u = -Kx + k_{\text{ref}} r + d \tag{2}$$

For this choice of u,

- x is the state of the system. It is known, but we cannot change it.
- K is a matrix that we get to choose.
- k_{ref} is a scalar that we get to choose.
- r is the reference. We may get to choose it, or it may be given to us.
- d is the disturbance. We **do not** know or get to choose the value d.

When $d = 0$, we've seen how to choose K and k_{ref} so that y, the output of our system, tracks the reference r in steady-state. We've also seen that when $d \neq 0$, the controller (2) is not able to track the reference. What change can we make to the controller (2) so that we can track the the reference r in steady-state?

We could try to add a term that is proportional to $y - r$, which is the difference between the output and the reference. However, in our state-space system, y is defined by $y = Cx$. Adding a term of the form $-k(y - r)$ to (2) therefore gives

$$\begin{aligned} u &= -Kx + k_{\text{ref}} r + d - k(y - r) \\ &= -(K - kC)x + (k_{\text{ref}} + k)r + d, \end{aligned}$$

which has the same form as (2). This approach, therefore, cannot compensate for the non-zero disturbance.

Let's try to add a term that not only penalizes the current difference between y and r, but also penalizes the *past* difference between y and r. One way to measure the past difference between y and r is to integrate the difference from the initial time, say $t_0 = 0$, to the current time, t, as in

$$\int_0^t (y(\tau) - r(\tau))\, d\tau.$$

This integral gives us a way of measuring the difference between the output and the reference at every time before the current time. Adding this to the controller (2) gives

$$u = -Kx + k_{\text{ref}} r + d - k_{\text{int}} \int_0^t (y(\tau) - r(\tau))\, d\tau,$$

where k_{int} is a scalar that we get to choose.

It may seem like this controller would be hard to implement, for example, in MATLAB. Do we have to evaluate the integral each time we want to compute a control input? If we are clever, the answer is no. Let's define a new variable $v(t)$, which is a function of time, so that $v(t)$ satisfies the differential equation

$$\dot{v}(t) = y(t) - r(t) \qquad\qquad v(0) = 0.$$

The solution of this differential equation is

$$v(t) = \int_0^t (y(\tau) - r(\tau))\, d\tau,$$

which is exactly the term in our controller! We can now write the controller as

$$u = -Kx + k_{\text{ref}} r + d - k_{\text{int}} v. \tag{3}$$

A controller of the form (3) is often called a controller *with integral action*.

If we apply this controller to the open-loop system described by (1), the closed-loop system is governed by the following four equations:

$$\dot{x} = Ax + Bu \tag{4}$$
$$\dot{v} = y - r \tag{5}$$
$$y = Cx \tag{6}$$
$$u = -Kx + k_{\text{ref}} r + d - k_{\text{int}} v. \tag{7}$$

When solving this closed-loop system, we can treat v simply as another state of the state-space system. **However, note that v is a variable that we introduced. It does not have any meaning in terms of the original, open-loop system (1). We created v ourselves in order to apply the controller (3).**

2 Block Matrix Multiplication

In the previous section, we stated that when solving (4)-(7), we can treat v simply as another state of the state-space system. To do this, it is convenient to create a new state variable,

$$z = \begin{bmatrix} x \\ v \end{bmatrix}.$$

We then have

$$\dot{z} = \begin{bmatrix} \dot{x} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} Ax + Bu \\ y - r \end{bmatrix}. \tag{8}$$

Expressions like the one above (after plugging in (6) and (7)) can often be simplified by writing the expression in terms of block matrices. As an example, consider the matrix

$$\begin{bmatrix} A_1 x + A_2 v \\ C_1 x + C_2 v \end{bmatrix}. \tag{9}$$

Even though A_1, A_2, C_1, and C_2 are matrices, we can rewrite (9) as

$$\begin{bmatrix} A_1 & A_2 \\ C_1 & C_2 \end{bmatrix} \begin{bmatrix} x \\ v \end{bmatrix}. \tag{10}$$

In other words, we can treat the blocks of this matrix as if they were scalars. You can check, by hand or in Matlab, that (9) and (10) are equivalent.

Note that multiplying block matrices in this way allows us to rewrite (8) as

$$\dot{z} = Ez + Fw,$$

where E and F are matrices and

$$w = \begin{bmatrix} r \\ d \end{bmatrix}.$$

This is a differential equation that we know how to solve in closed-form! We can also write

$$y = Gz,$$

where G is a matrix and y is the output of our original system (1).

3 Example and MATLAB Snippets

We show the relevance of integral action in a problem with disturbance in the control input using the following example. Consider a system

```
>> A = [0.9 −0.8 0.0; 0.1 0.8 0.6; 0.9 −0.1 0.2];
>> B = [0.3; 0.5; −0.7];
>> C = [0.0 0.8 −0.8];
>> D = [0.0];
```

with the control input

$$u = -Kx + -k_{\text{int}} v + k_{\text{ref}} r + d$$

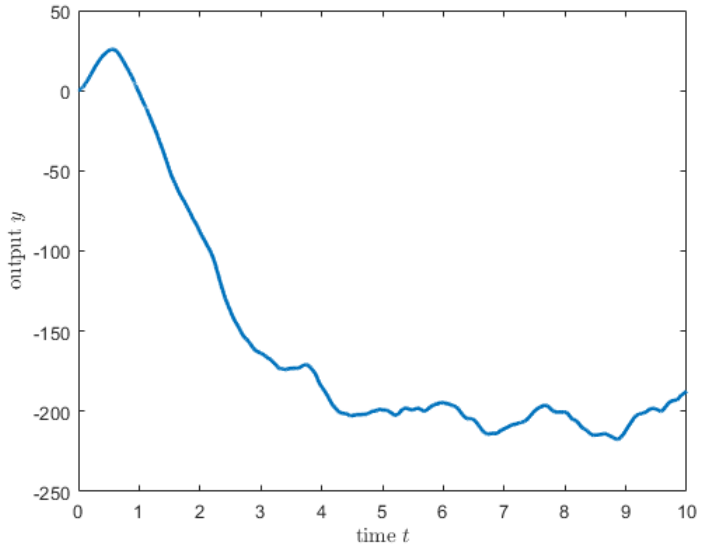
where d is the unknown, possibly time-varying disturbance. We want our output to track a reference $r = 2$. For this example, we assume that the random disturbance is is $d(t) = K_{\text{ref}} * rand(0, 1)$ where $rand(0, 1)$ is a random value between 0 and 1.

Let us first pretend that we do not know the disturbance d exists in the control input. In that case, to track the reference with zero steady-state error, we solve for K and K_{ref} using methods we discussed in the previous lectures. The following MATLAB code does exactly that and then plots the output y:

```
% initialize
x_init = [0;0];
ref = 2;

% Design K and Kref
K = place(A,B, [−3, −2, −1]); % randomly selected negative eigenvalues
Kref = −1/(C∗inv(A−B∗K)∗B);

[t,x]=ode45(@(t,x)(A−B∗K)∗x+B∗Kref∗ref + rand∗Kref,[0 10],[0;0;0]);
plot(t,C∗x');
xlabel('time $t$', 'Interpreter','latex');
ylabel('output $y$', 'Interpreter','latex');
```



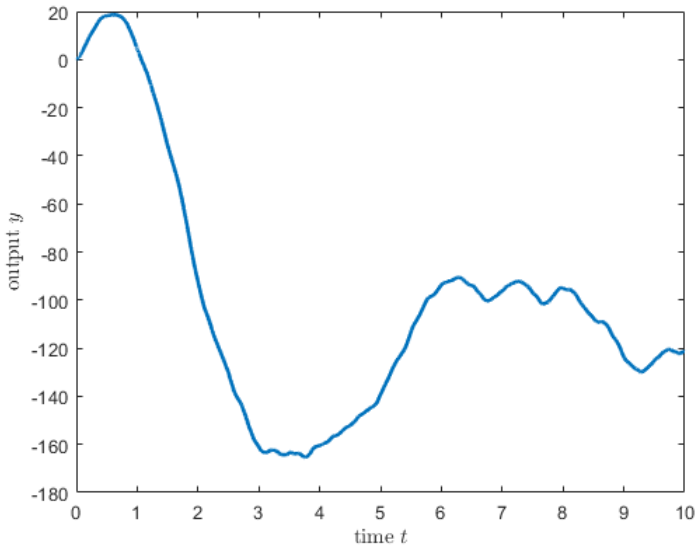
Clearly, the output y does not converge to the reference value of 2. Suspecting that there might be disturbance in the input, we introduce the component $-k(y-r)$ in the input u. For reasons discussed above, this will not any positive impact on the controller performance. We use the following MATLAB code to prove compute and plot the output y for $k = 3$ to prove that it is indeed the case.

```
% initialize
x_init = [0;0];
ref = 2;

% Design K and Kref
K = place(A,B, [−3, −2, −1]); % randomly selected negative eigenvalues
Kref = −1/(C∗inv(A−B∗K)∗B);
k = 3;

```

```
[t,x]=ode45(@(t,x)(A-B*K)*x + B*Kref*ref + rand*Kref - k*B*(C*x - ref),[0
    10],[0;0;0]);
plot(t,C*x');
xlabel('time $t$', 'Interpreter','latex');
ylabel('output $y$', 'Interpreter','latex');
```



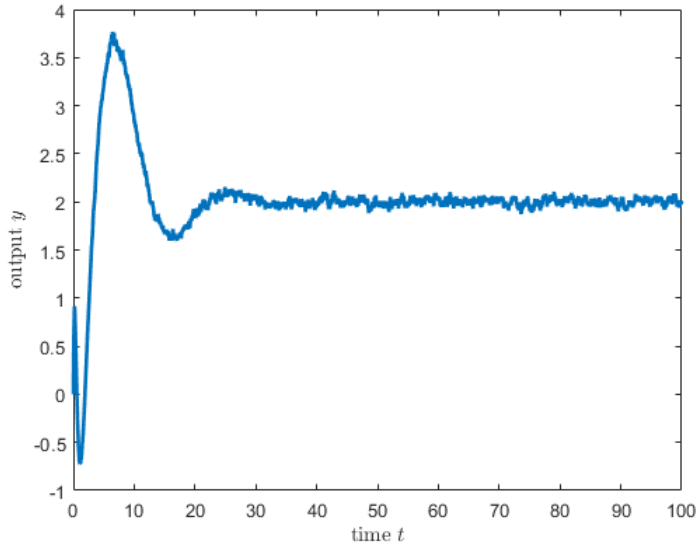
It is evident from the plot that the controller is still performing poorly. Finally, let us consider integral action as described above with $K_{\text{int}} = 1$. We extend the state-space model to include v and use z to denote our extended set of state variables.

```
% initialize
z_init = [0;0;0;0]; % z = [x;v]
ref = 2;
Kint = 1;

% define extended system zdot = Ez + Fw and y = Gz
E = [A-B*K, -B*Kint; C, 0];
F = [B*Kref, B; -1, 0];
G = [C, 0];

% check that the system is asymptotically stable
eig(E); % all eigenvalues must have strictly negative real parts

% solve for states and plot output
[t,z]=ode45(@(t,z)(E)*z + F*[ref; rand*Kref],[0 100],z_init);
plot(t,G*z');
xlabel('time $t$', 'Interpreter','latex');
ylabel('output $y$', 'Interpreter','latex');
```



As you can see in the plot above, the output indeed converges to the reference $r = 2$. Integral action works! (even when you don't know d!!). Note that if the disturbance was a constant value, the output y will exactly converge to the reference r after sometime. However, the disturbance in this case is time-varying (due to the *rand* component) and hence the output y is also a little noisy.