3.6 Output feedback using an observer

3.6.1 Basic idea

In many situations, that the state is not measured entirely means that one cannot use state-feedback directly. In the case when only the output $y(t)$ is given, one can use the observer estimate $\hat{\mathbf{x}}(t)$ in lieu of the state $\mathbf{x}(t)$ and use our good old linear state-feedback technique.
Indeed, we replace the control law

$$u(t) = -\mathbf{K}\mathbf{x}(t) \tag{3.56}$$

with

$$u(t) = -\mathbf{K}\hat{\mathbf{x}}(t) \tag{3.57}$$

In doing so, note that we are hence combining both a state-feedback controller and an observer to end up with a controller referred to as *output feedback controller*, since it only uses the output for feedback as seen in the block diagram of figure 3.6.

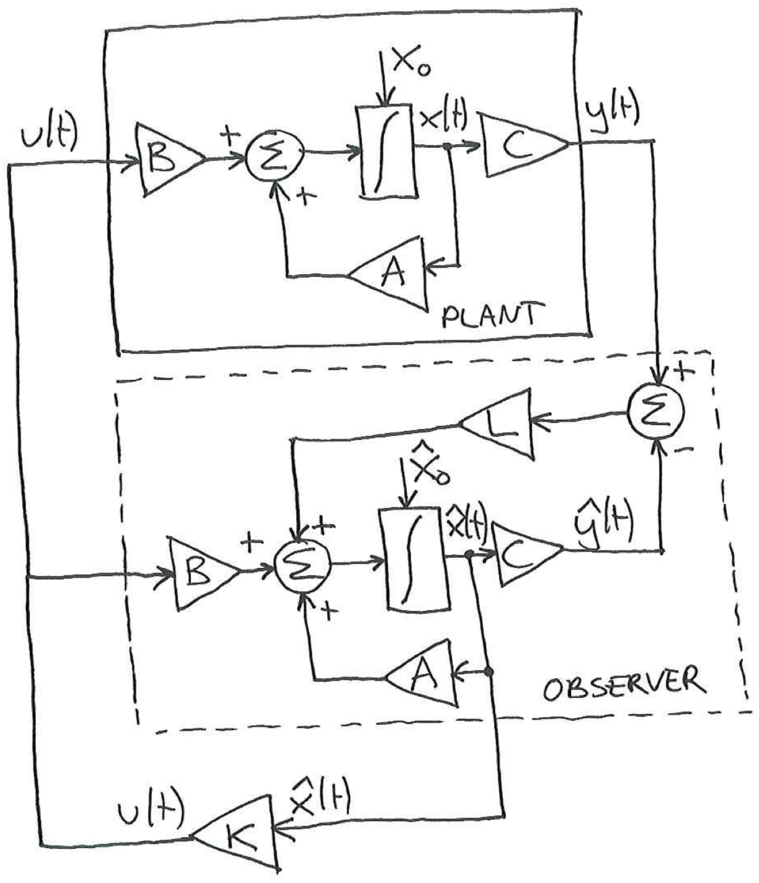

Figure 3.6: Output feedback controller combining linear state feedback controller and observer

Contrary to state-feedback controller (3.56), which is static, control law (3.57) represents what is called a dynamic feedback controller, since combining (3.57) with observer (3.27) gives the dynamic system

$$\begin{cases} \dot{\hat{\mathbf{x}}}(t) = (\mathbf{A} - \mathbf{LC} - \mathbf{BK})\,\hat{\mathbf{x}}(t) + \mathbf{L}y(t) \\ u(t) = -\mathbf{K}\hat{\mathbf{x}}(t) \end{cases}, \tag{3.58}$$

which is very easy to implement, for example with a single state-space block in Matlab/Simulink.

3.6.2 Application to tracking control

Once we are able to stabilize a system by output feedback, tracking control using only the output for measurement, where we want the system to follow a desired trajectory $\mathbf{x}_d(t)$, is not very far. Indeed, recall that for a tracking controller, we had

$$\Delta u(t) = -\mathbf{K}\Delta\mathbf{x}(t) \tag{3.59}$$

where $\Delta u(t)$ and $\Delta\mathbf{x}(t)$ were given by

$$\begin{cases} \Delta u(t) := u(t) - u_d(t) \\ \Delta\mathbf{x}(t) := \mathbf{x}(t) - \mathbf{x}_d(t) \end{cases}, \tag{3.60}$$

which gave us

$$u(t) = -\mathbf{K}\left(\mathbf{x}(t) - \mathbf{x}_d(t)\right) + u_d(t). \tag{3.61}$$

Now, as seen previously, replace $\mathbf{x}(t)$ with its estimate $\hat{\mathbf{x}}(t)$ so that we have

$$u(t) = -\mathbf{K}(\hat{\mathbf{x}}(t) - \mathbf{x}_d(t)) + u_d(t). \tag{3.62}$$

Note, as can be seen in the block diagram below (see figure 3.7), this last controller consists of 3 main subparts: a feedforward controller (providing desired trajectories), an observer and a linear state-feedback controller.

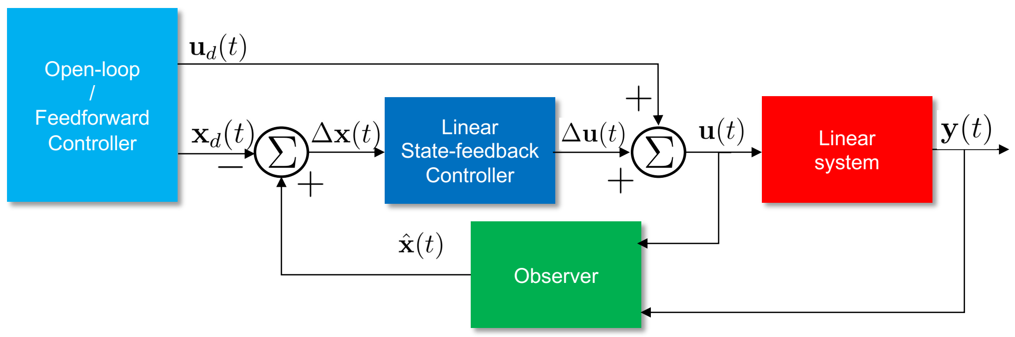

Figure 3.7: Block diagram of a tracking controller using only the output as measurement

3.6.3 Separation principle

Since, in the general controller structure, it is not $\mathbf{x}(t)$ but $\hat{\mathbf{x}}(t)$ that is passed to the tracking controller, and since $\hat{\mathbf{x}}(t)$ and $\mathbf{x}(t)$ will not be the same for some time (because $\hat{\mathbf{x}}_0 \neq \mathbf{x}_0$), then we would like to know what the influence of the estimation error on the tracking controller is.

Indeed, for a tracking controller **without observer**, the tracking error dynamics is:

$$\frac{d}{dt}(\Delta\mathbf{x}) = \mathbf{A}\Delta x - \mathbf{BK}\Delta\mathbf{x} \tag{3.63}$$

because

$$\Delta u = -\mathbf{K}\Delta\mathbf{x}. \tag{3.64}$$

Turning now to the influence of the observer, we replace $\mathbf{x}(t)$ by $\hat{\mathbf{x}}(t)$ to get the feedback law

$$u(t) = -\mathbf{K}\left(\hat{\mathbf{x}}(t) - \mathbf{x}_d(t)\right) + u_d(t). \tag{3.65}$$

Then, recalling that

$$\tilde{\mathbf{x}}(t) := \hat{\mathbf{x}}(t) - \mathbf{x}(t), \tag{3.66}$$

we get

$$\begin{aligned} u(t) &= -\mathbf{K}\left(\tilde{\mathbf{x}}(t) + \mathbf{x}(t) - \mathbf{x}_d(t)\right) + u_d(t) \\ &= -\mathbf{K}\left(\tilde{\mathbf{x}}(t) + \Delta\mathbf{x}(t)\right) + u_d(t) \\ &\Downarrow \\ \Delta u(t) &= -\mathbf{K}\Delta\mathbf{x}(t) - \mathbf{K}\tilde{\mathbf{x}}(t) \end{aligned} \tag{3.67}$$

This, in turn gives, for a tracking controller with observer, the following tracking error dynamics:

$$\frac{d}{dt}\left(\Delta\mathbf{x}(t)\right) = (\mathbf{A} - \mathbf{BK})\Delta\mathbf{x}(t) - \mathbf{BK}\tilde{\mathbf{x}}(t) \tag{3.68}$$

in equation (3.68), $\tilde{\mathbf{x}}(t)$ can be seen as an input. Hence, the evolution of the tracking error is influenced by the estimation error.

However, the estimation error is itself driven by the estimation error dynamics

$$\dot{\tilde{\mathbf{x}}}(t) = (\mathbf{A} - \mathbf{LC})\tilde{\mathbf{x}} \tag{3.69}$$

When equation (3.69) is stable, we have $\tilde{\mathbf{x}} \to 0$ as $t \to 0$. This also means that $\tilde{\mathbf{x}}$ will have less and less influence on the tracking error $\Delta\mathbf{x}(t)$ as time goes by. To summarize this mathematically, we put (3.68) and (3.69) together to tobtain:

$$\frac{d}{dt}\begin{bmatrix}\tilde{\mathbf{x}} \\ \Delta\mathbf{x}\end{bmatrix} = \begin{bmatrix}\mathbf{A} - \mathbf{LC} & 0 \\ -\mathbf{BK} & \mathbf{A} - \mathbf{BK}\end{bmatrix}\begin{bmatrix}\tilde{\mathbf{x}} \\ \Delta\mathbf{x}\end{bmatrix} \tag{3.70}$$

The above dynamics are stable if the eigenvalues of $\mathbf{A} - \mathbf{LC}$ are in the left-half-plane and the eigenvalues of $\mathbf{A} - \mathbf{BK}$ are there as well, since the estimation error dynamics are in cascade with the tracking error dynamics. Since only the feedback gain is present in equation (3.69), the tuning of $\mathbf{K}$ does not influence the stability of the estimation error dynamics.

Similarly, the tuning of $\mathbf{L}$ does not influence the dynamics of equation (3.68). This means that the gains $\mathbf{L}$ and $\mathbf{K}$ can be tuned separately. This is called the **Separation Principle**.

3.7 Active Disturbance Rejection Control

In the state-space techniques we have seen so far, whether it was state feedback or output feedback through the use of an observer, the main assumption or starting point, was that a model was available to us, so that we could use a state-space representation. While obtaining such a model is possible in practise, through model identification, it can be quite time-consuming, especially for orders higher than two.
The main advantage of PID controllers are their relative simplicity, as well as the fact that one does not necessarily need a model of the plant to use a PID controller. It is a model-free technique. Over the years, control specialists have looked for such model-free techniques for the state-space context.
One of these techniques originated in China in the 1980s, and has garnered a significant popularity in the last decade. Called Active Disturbance Rejection Control, and has the observer context as its core. To describe this model-free control technique, let us start with seeing how simple disturbances can be estimated using an observer.

3.7.1 Estimating disturbances

To begin with, as per our usual philosophy, let us start with the following very simple scalar example

$$\dot{x} = ax + bu + d \tag{3.71}$$

where the output of this scalar system is $y = x$ and where the term d is a constant but unknown disturbance (also a scalar). Note that this model is quite similar to the one used in our discussion on adding an integral term to state-feedback.
Our objective here is to design an observer that will not only give an estimate of x, but also an estimate of the constant disturbance. This is useful for feedback control, as well as in its own right, for monitoring purposes, for example.
To do so, noting that d is constant, notice that model (3.71) is equivalent to

$$\begin{cases} \dot{x} = ax + bu + d \\ \dot{d} = 0 \end{cases} \tag{3.72}$$

or, defining the new extra state variable $x_d := d$, so that we have the new state-space representation, in component form

$$\begin{cases} \dot{x} = ax + x_d + bu \\ \dot{x}_d = 0 \end{cases}, \tag{3.73}$$

where we still have $y = x$. The corresponding vectorial form of this linear state-space representation of dimension 2 is

$$\begin{cases} \dfrac{d}{dt}\begin{bmatrix} x \\ x_d \end{bmatrix} = \begin{bmatrix} a & 1 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} x \\ x_d \end{bmatrix} + \begin{bmatrix} b \\ 0 \end{bmatrix} u \\ y(t) = \begin{bmatrix} 1 & 0 \end{bmatrix}\begin{bmatrix} x \\ x_d \end{bmatrix} \end{cases} \tag{3.74}$$

The above system is easily checked to be observable. Consequently it is easy for us to design an observer that will give the estimate $\hat{x}$ of x as well as an estimate $\hat{x}_d = \hat{d}$ of constant disturbance d. Tuning this observer will not be done by using the original system parameters a and b alone, but by using the extended structure (3.74), ie we have the extended state-space representation matrices

$$\mathbf{A}_E = \begin{bmatrix} a & 1 \\ 0 & 0 \end{bmatrix}, \quad \mathbf{B}_E = \begin{bmatrix} b \\ 0 \end{bmatrix} \quad \text{and} \quad \mathbf{C}_E = \begin{bmatrix} 1 & 0 \end{bmatrix} \tag{3.75}$$

corresponding to the extended state $\mathbf{x}_E = [x, x_d]^T$.
Generalizing beyond the scalar case is pretty easy, starting with the model

$$\begin{cases} \dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}(u+d), & \mathbf{x}(0) = \mathbf{x}_0 \\ y = \mathbf{C}\mathbf{x} \end{cases}, \tag{3.76}$$

which, setting again $x_d := d$, gives

$$\begin{cases} \dot{\mathbf{x}}_E = \mathbf{A}_E\mathbf{x}_E + \mathbf{B}_E u \\ y = \mathbf{C}_E\mathbf{x}_E \end{cases}, \tag{3.77}$$

where we have

$$\mathbf{A}_E = \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ 0 & 0 \end{bmatrix}, \quad \mathbf{B}_E = \begin{bmatrix} \mathbf{B} \\ 0 \end{bmatrix} \quad \text{and} \quad \mathbf{C}_E = \begin{bmatrix} \mathbf{C} & 0 \end{bmatrix}. \tag{3.78}$$

Interestingly, $d(t)$ can also be something else than a mere constant. Indeed, as long as there is a dynamic model for the disturbance, the latter can be integrated into matrix $\mathbf{A}_E$.
Assume, for example, that our disturbance is a sine wave of unknown amplitude A and phase ϕ, but known frequency ω, ie we have $d(t) = A\sin(\omega t + \phi)$. In this case, $d(t)$ is the solution of the well-known second-order differential equation

$$\ddot{d} = -\omega^2 d \tag{3.79}$$

to which the following state-space representation corresponds

$$\begin{cases} \dot{\mathbf{x}}_d = \mathbf{A}_d\mathbf{x}_d \\ d = \mathbf{C}_d\mathbf{x}_d \end{cases}, \tag{3.80}$$

with

$$\mathbf{x}_d = \begin{bmatrix} x_{d,1} \\ x_{d,2} \end{bmatrix} := \begin{bmatrix} d \\ \dot{d} \end{bmatrix}, \quad \mathbf{A}_d = \begin{bmatrix} 0 & 1 \\ -\omega^2 & 0 \end{bmatrix} \quad \text{and} \quad \mathbf{C}_d = \begin{bmatrix} 1 & 0 \end{bmatrix}. \tag{3.81}$$

Combining then, as we have done for the scalar case, state vector $\mathbf{x}$ with disturbance state $\mathbf{x}_d$ to get the extended state $\mathbf{x}_E$, we get system (3.77) again, with the same matrices to the exception of matrix $\mathbf{A}_E$, which is given bybut this time with matrices

$$\mathbf{A}_E = \begin{bmatrix} \mathbf{A} & \mathbf{B}\mathbf{C}_d \\ 0 & \mathbf{A}_d \end{bmatrix}. \tag{3.82}$$

Note that the set of disturbances that can be considered is quite large. Indeed, one can indeed, following a similar reasoning as the above, have a disturbance consisting of a sine wave with a bias (ie plus a constant), or even several disturbances! As a final remark, one can also add disturbances directly on the output of the system, such as a constant bias on the sensor for example. In all these cases, one simply has to make sure that the extended system is indeed observable.

3.7.2 Extended-State Observers

One of the secrets of ADRC lies in the way the above extended-state observers are used in the context of model-free control. To see this, let us take a scalar example again. Indeed, consider the following first-order differential equation

$$\dot{y} = f(y, w, t) + bu \tag{3.83}$$

where we assume that parameter b is a known constant, while u is our usual control input. Note that we took the liberty to express the above equation with y as the differentiated variable as we have $y = x$. Variable $w(t)$ is an unknown time-varying disturbance. Finally, and very importantly, all we know about function $f(\bullet)$ is that is does not change too much/too fast according to y, w or t.
Let us rephrase this previous last sentence: we assume that function f is actually constant, but *on a very short time-horizon*. This means that, locally in time, we have $\dot{f} \approx 0$.
Hence we can define the following extended-state observer for model (3.83):

$$\begin{cases} \dot{\hat{y}} = \hat{f} + bu + l_1(y - \hat{y}) \\ \dot{\hat{f}} = l_2(y - \hat{y}) \end{cases} \tag{3.84}$$

or, in matrix form,

$$\begin{cases} \dot{\hat{\mathbf{x}}}_E = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \hat{\mathbf{x}}_E + \begin{bmatrix} b \\ 0 \end{bmatrix} u + \begin{bmatrix} l_1 \\ l_2 \end{bmatrix} (y - \hat{y}) \\ \hat{y} = \begin{bmatrix} 1 & 0 \end{bmatrix} \hat{\mathbf{x}}_E \end{cases} . \tag{3.85}$$

where $\hat{\mathbf{x}}_E = [\hat{y}, \hat{f}]^T$.
One might wonder how the above observer is able to do an OK job, since f is not really a constant after all. If tuned propertly (roughly speaking with gains l_i chosen high enough), the observer will quickly estimate 'constantÂ´f, even as it 'moves' to something else. In other words, as long as the observer dynamics are faster than the system dynamics, then $\hat{f}$ will be an acceptable estimate of f.
Many systems being of the second-order type (think Newton's second law, for example), models considered in ADRC are not only of the first order, as above, but modelled by the following second-order differential equation.

$$\ddot{y} = f(y, \dot{y}, w, t) + bu. \tag{3.86}$$

where all the terms are similarly described as for first-order case (3.83), with f also depending on $\dot{y}$. As an example, notice how our usual pendulum model fits the above class of systems.
One can easily obtain a (nonlinear) state-space representation from equation (3.86):

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = f(x_1, x_2, w, t) + bu \\ y = x_1 \end{cases} . \tag{3.87}$$

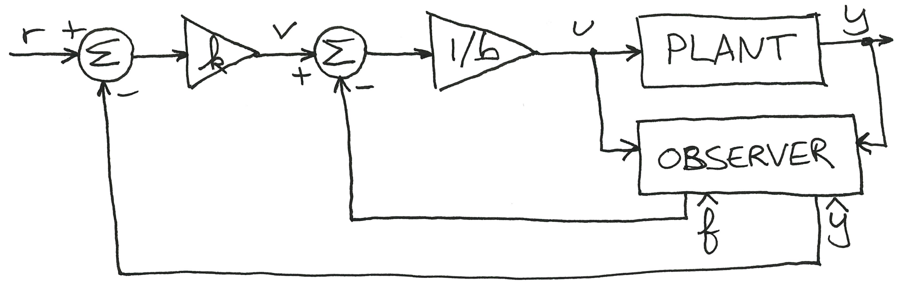

Figure 3.8: Linear ADRC structure for first-order plant

Following the same reasoning as in the scalar case, we get the extended-state observer (ESO)

$$\begin{cases} \dot{\hat{\mathbf{x}}}_E = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \hat{\mathbf{x}}_E + \begin{bmatrix} 0 \\ b \\ 0 \end{bmatrix} u + \begin{bmatrix} l_1 \\ l_2 \\ l_3 \end{bmatrix} (y - \hat{y}) \\ \hat{y} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \hat{\mathbf{x}}_E \end{cases} . \tag{3.88}$$

3.7.3 Model-free control using linear ADRC

We are now ready to actually use the above to do feedback control. This is actually quite simple. A bit similarly to what we have seen when we looked at feedback linearization, one can consider that our controller will have 2 parts: a cancelling term and a stabilizing one.
Thus, provided that $\hat{f} \approx f$, as obtained by an ESO, the first controller cancelling the unknown function f will be

$$u = \frac{-\hat{f} + v}{b}. \tag{3.89}$$

Indeed, putting this controller into, say, first-order system (3.83), we have

$$\dot{y} = v, \tag{3.90}$$

which is a simple integrator, while putting this controller into second-order system (3.86) or (3.87), we get

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = v \end{cases} , \tag{3.91}$$

ie a double-integrator! Note that, in each of these cases, ie first or second-order system, we have used the corresponding extended-state observer.
Now that we have cancelled the unknown function f, we simply stabilize the simple or double integrators with the observer-based controllers

$$v = -k\hat{y} \tag{3.92}$$

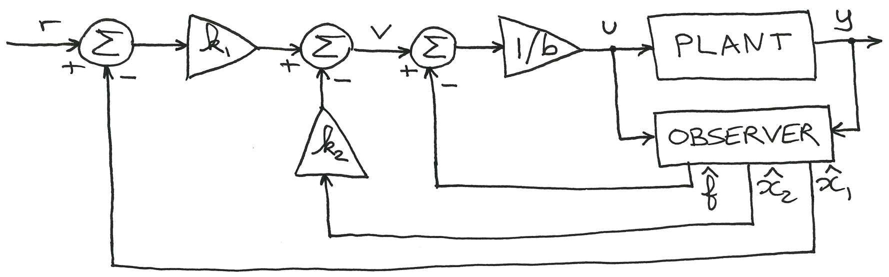

Figure 3.9: Linear ADRC structure for second-order plant

for the first-order example (with k a tuning gain), and

$$v = -k_1\hat{x}_1 - k_2\hat{x}_2. \tag{3.93}$$

The above can be further generalized to higher-order systems, but note that this is rarely done in practise, where it is mostly the second-order which is used. For implementation Figure 3.8 shows the linear ADRC structure for the first-order case, while Figure 3.9 shows the second-order one, where both diagrams also include a reference input.
Regarding tuning, the most well-known rule consists in simply choosing repeated poles for the observer (eg two -10's for a first-order system), and similarly for the pole(s) related to the controller. The poles of the observer are usually chosen to be between 3 and 10 times faster than those related to the controller.

As a conclusion, note that the above is yet another small introduction to a very important topic, incredibly useful in practical applications. We have just given the essence, and additional tricks exist so that ADRC performs even better. Generally, ADRC is known to often outperform PID controllers and it is surprisingly not always known in the industry, despite its simplicity.