Notes for Discrete-Time Control Systems (ECE-520)

Fall 2010

by
R. Throne

The major sources for these notes are

- *Modern Control Systems*, by Brogan, Prentice-Hall, 1991.
- *Discrete-Time Control Systems*, by Ogata. Prentice-Hall, 1995.
- *Computer Controlled Systems*, by Åström and Wittenmark. Prentice-Hall, 1997.
- *Analog and Digital Control System Design*, by C. T. Chen. Sanders College Publishing. 1993.
- *Digital Control of Dynamic Systems*, by Franklin, Powell, and Workman. Addison Wesley, 1998.
- *Modern Control Systems*, by Dorf and Bishop. Prentice-Hall, 2008.

Contents

1 z-Transforms

In this course we will assume we are sampling the continuous time signal $x(t)$ at a uniform sampling rate. The time interval between samples will be denoted by T. Thus we will denote the discrete-time (sampled) values as $x(0)$, $x(T)$, x(2T),$\ldots, x(kT)$. This is shown graphically in Figure 1. Sometimes we remove the explicit dependence on the sampling interval T and just write $x(0)$, $x(1)$, $x(2)$, $\ldots$, $x(k)$, since the sample interval is the same for all of the different samples.

1.1 Special Functions

Just as in continuous-time, there are certain special functions that are used very often in discrete-time. Specifically, we will be concerned with the unit impulse function, the unit step function, and the unit ramp function.

The *unit impulse* or *delta* function is defined as

$$\delta(k) = \begin{cases} 1 & k = 0 \\ 0 & k \neq 0 \end{cases}$$

or

$$\delta(n-k) = \begin{cases} 1 & k-n = 0 \\ 0 & k-n \neq 0 \end{cases}$$

The *unit step* or *Heaviside* function is defined as

$$u(k) = \begin{cases} 1 & k \geq 0 \\ 0 & k < 0 \end{cases}$$

or

$$u(n-k) = \begin{cases} 1 & n-k \geq 0 \\ 0 & n-k < 0 \end{cases}$$

The *unit ramp* function is defined as

$$r(k) = ku(k)$$

or

$$r(n-k) = (n-k)u(n-k)$$

While there are other special function, these are the special functions we will be utilizing the most.

1.2 Impulse Response and Convolution

The unit impulse response of a Linear Time-Invariant (LTI) system, $h(k)$, is the response of the system at rest (no initial energy, the initial conditions are all zero), to a unit impulse at time zero. Symbolically, we can write

$$\delta(k) \rightarrow h(k)$$

Since the system is assumed to be time-invariant we also have

$$\delta(n-k) \rightarrow h(n-k)$$

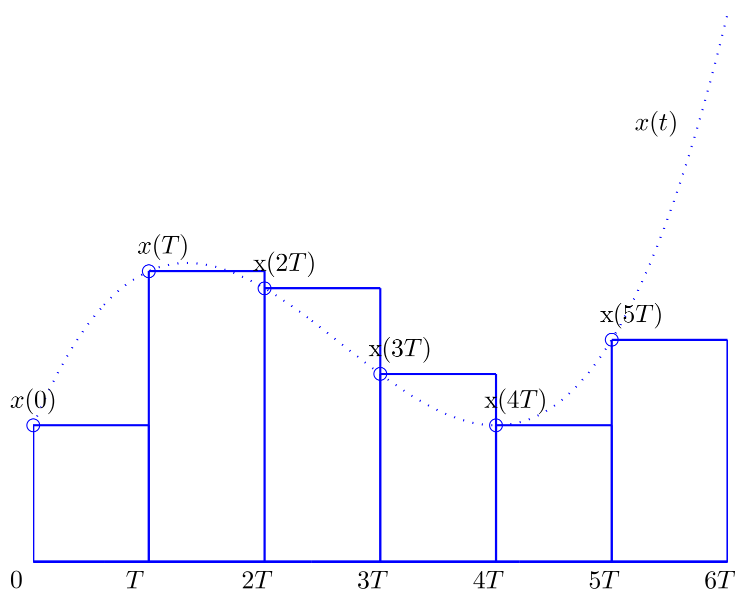

Figure 1: Continuous-time signal $x(t)$ and its samples $x(kT)$. We assume the samples are taken at the beginning of each time interval.

Since the system is also linear, we have

$$\alpha\delta(k) \rightarrow \alpha h(k)$$

and

$$\alpha\delta(n-k) \rightarrow \alpha h(n-k)$$

Now we can write $x(n)$ as

$$x(n) = \ldots + x(-2)\delta(n+2) + x(-1)\delta(n+1) + x(0)\delta(n) + x(1)\delta(n-1) + x(2)\delta(n-2) + \ldots$$

If $x(n)$ is the input to an LTI system, then we can compute the output as

$$y(n) = \ldots + x(-2)h(n+2) + x(-1)h(n+1) + x(0)h(n) + x(1)h(n-1) + x(2)h(n-2) + \ldots$$

since we treat the $x(k)$'s as constants, and the system is only responding the the impulses. We can write this expression as

$$y(n) = \sum_{k=-\infty}^{\infty} x(k)h(n-k)$$

or, by making a change of variable in the sum, as

$$y(n) = \sum_{k=-\infty}^{\infty} x(n-k)h(k)$$

Thus the output of an LTI system is the *convolution* of the input with the impulse response of the system,

$$\begin{aligned} y(n) &= h(n) \star x(n) = \sum_{k=-\infty}^{\infty} h(k)x(n-k) \\ &= x(n) \star h(n) = \sum_{k=-\infty}^{\infty} x(k)h(n-k) \end{aligned}$$

If both the system and input are *causal* (both are zero for $n \leq 0$), then we can simplify the convolution sums as

$$\begin{aligned} y(n) &= h(n) \star x(n) = \sum_{k=0}^{n} h(k)x(n-k) \\ &= x(n) \star h(n) = \sum_{k=0}^{n} x(k)h(n-k) \end{aligned}$$

1.3 A Useful Summation

In what follows, we will make extensive use of a simple summation formula, which you probably saw in high school. Let's denote the sum S_n as

$$S_n = 1 + a + a^2 + \ldots + a^{n-1} + a^n = \sum_{k=0}^{k=n} a^k$$

If we multiply by a we have

$$\begin{aligned} aS_n &= a\left(1 + a + a^2 + \ldots + a^{n-1} + a^n\right) \\ &= a + a^2 + a^3 + \ldots + a^n + a^{n+1} \\ &= S_n - 1 + a^{n+1} \end{aligned}$$

Solving for S_n we have

$$\begin{aligned} aS_n - S_n &= a^{n+1} - 1 \\ S_n &= \frac{a^{n+1} - 1}{a - 1} \end{aligned}$$

or

$$\sum_{k=0}^{k=n} a^k = \frac{1 - a^{n+1}}{1 - a}$$

Note that if $n = \infty$ and $|a| < 1$, then we have

$$\sum_{k=0}^{\infty} a^k = \frac{1}{1-a}$$

Now let's do some convolution examples.

Example. Assume the input to a system is $x(n) = u(n)$ and the impulse response of the system is $h(n) = u(n)$. Determine the output of the system. We have

$$\begin{aligned} y(n) &= x(k) \star h(k) \\ &= \sum_{k=-\infty}^{\infty} x(k)h(n-k) \\ &= \sum_{k=-\infty} u(k)u(n-k) \\ &= \sum_{k=0}^{n} (1)(1) \\ &= (n+1)u(n) \end{aligned}$$

Note that we need the final $u(n)$ since our method is only valid if $n \geq 0$.

Example. Assume the input to a system is $x(n) = \alpha^n u(n)$ and the impulse response is $h(n) = u(n)$. Determine the output of the system. We have

$$\begin{aligned} y(n) &= x(k) \star h(k) \\ &= \sum_{k=-\infty}^{\infty} x(k)h(n-k) \\ &= \sum_{k=-\infty} \alpha^k u(k)u(n-k) \\ &= \sum_{k=0}^{n} \alpha^k \\ &= \frac{1-\alpha^{n+1}}{1-\alpha} u(n) \end{aligned}$$

Note that we need the final $u(n)$ since our method is only valid if $n \geq 0$.

Example. Assume the input to a system is $x(n) = \alpha^{n+2}u(n+2)$ and the impulse response is $h(n) = \beta^{n-1}u(n-1)$. Determine the output of the system. We have

$$\begin{aligned} y(n) &= x(k) \star h(k) \\ &= \sum_{k=-\infty}^{\infty} x(k)h(n-k) \\ &= \sum_{k=-\infty} \alpha^{k+2}u(k+2)\beta^{n-k-1}u(n-k-1) \end{aligned}$$

Now let's look at our sums. We know $u(k+2) = 1$ for $k \geq -2$ and $u(n-k-1) = 1$ for $n-k-1 \geq 0$ or $n-1 \geq k$. Hence our sum is

$$y(n) = \sum_{k=-2}^{n-1} \alpha^{k+2}\beta^{n-k-1}$$

In order to use our summation formula, let's change variables in this sum. We want the lower limit to be zero, so let's define $l = k+2$, or $k = l-2$. In terms of l, our new sum is

$$y(n) = \sum_{l=0}^{l=n+1} \alpha^{l}\beta^{n-l+1}$$

$$
\begin{aligned}
&= \beta^{n+1}\sum_{l=0}^{n+1}\left(\frac{\alpha}{\beta}\right)^{l} \\
&= \beta^{n+1}\left[\frac{1-(\frac{\alpha}{\beta})^{n+2}}{1-\frac{\alpha}{\beta}}\right]u(n+1)
\end{aligned}
$$

Example. Assume the input to a system is $x(n) = \alpha^{-n}u(-n)$ and the impulse response is $h(n) = \beta^{n-1}u(n-1)$. Determine the output of the system. We have

$$
\begin{aligned}
y(n) &= x(k) \star h(k) \\
&= \sum_{k=-\infty}^{\infty} x(k)h(n-k) \\
&= \sum_{k=-\infty} \alpha^{-k}u(-k)\beta^{n-k-1}u(n-k-1)
\end{aligned}
$$

We know that $u(-k) = 1$ for $k \leq 0$ and $u(n-k-1) = 1$ for $n-k-1 \geq 0$ or $k \leq n-1$. Now we have to be very careful since our answers will depend on what we assume about n. For $n \geq 1$, the most restrictive condition is $k \leq 0$ Our new sum becomes

$$
\begin{aligned}
y(n) &= \sum_{k=-\infty}^{0} \alpha^{-k}\beta^{n-k-1} \\
&= \beta^{n-1}\sum_{k=-\infty}^{0}(\alpha\beta)^{-k}
\end{aligned}
$$

Letting $l = -k$ and assuming the sum is *absolutely convergent* so that we can rearrange the order of the summation, we have

$$
y(n) = \beta^{n-1}\sum_{l=0}^{l=\infty}(\alpha\beta)^{l} = \beta^{n-1}\frac{1}{1-\alpha\beta}
$$

Next, let's look at what happens for $n \leq 0$. Here the more restrictive condition is $k \leq n-1$, and we have the sum

$$
\begin{aligned}
y(n) &= \sum_{k=-\infty}^{n-1} \alpha^{-k}\beta^{n-k-1} \\
&= \beta^{n-1}\sum_{k=-\infty}^{n-1}(\alpha\beta)^{-k}
\end{aligned}
$$

Letting $l = k-(n-1) = k-n+1$ we have

$$
y(n) = \beta^{n-1}(\alpha\beta)^{-(n-1)}\sum_{l=-\infty}^{0}(\alpha\beta)^{-l}
$$

Finally, let $p = -l$ and assume absolute convergence, and we have

$$
y(n) = \alpha^{-n+1}\frac{1}{1-\alpha\beta}
$$

Combining our two regions we have

$$
y(n) = \beta^{n-1}\frac{1}{1-\alpha\beta}u(n-1) + \alpha^{-n+1}\frac{1}{1-\alpha\beta}u(-n)
$$

Note that there is no overlap in the definition regions (values of n).

1.4 z-Transforms

In continuous-time we can either compute the output of a system using the convolution directly, or we can resort to transform methods. In continuous-time control systems we generally utilize Laplace transforms and block diagrams, since this is a much easier way to determine the output than by using convolution in the time domain. Similarly, in discrete-time we can utilize the z-transform to avoid using the convolution sum. Just as with Laplace transforms, we can define one and two-sided z-transforms.

The *one-sided* z-transform is defined as

$$X(z) = \mathcal{Z}\{x(nT)\} = \sum_{k=0}^{\infty} x(kT)z^{-k}$$

and the *two-sided* z-transform is defined as

$$X(z) = \mathcal{Z}\{x(nT)\} = \sum_{k=-\infty}^{\infty} x(kT)z^{-k}$$

Just as with Laplace transforms, the only difference between the one and two-sided z-transform is the lower limit. Often we do not directly utilize the dependence on T and leave it out of the summation. In this course we are generally concerned with causal systems and causal inputs, so we generally will use the one-sided z-transform. Finally, the sum often only converges for values of $|z|$ within a specific region. This is called the *region of convergence* or *ROC*. Now let's do some examples.

Example. Determine the one-sided z-transform of $x(n) = \delta(n-l)$. We have

$$X(z) = \sum_{k=0}^{k=\infty} \delta(k-l)z^{-k} = z^{-l}$$

This sum converges for all values of $|z|$, so the region of convergence is the entire z-plane.

Example. Determine the one-sided z-transform of $x(n) = u(n)$, the unit step function. We have

$$X(z) = \sum_{k=0}^{\infty} u(k)z^{-k} = \sum_{k=0}^{\infty} \left(\frac{1}{z}\right)^{k} = \frac{1}{1-z^{-1}} = \frac{z}{z-1}$$

This is only true if the sum is absolutely convergent, which means $|\frac{1}{z}| < 1$ or if $|z| > 1$. Hence the ROC is defined by $|z| > 1$.

Example. Determine the one-sided z-transform of $x(n) = nu(n)$, the unit ramp function. We have

$$X(z) = \sum_{k=0}^{\infty} ku(k)z^{-k} = \sum_{k=0}^{\infty} kz^{-k}$$

At this point, we need to use something we all ready know, and some calculus. For the unit step we have

$$\frac{z}{z-1} = \sum_{k=0}^{\infty} z^{-k}$$

Taking derivatives of both sides with respect to z yields

$$\begin{aligned}
\frac{d}{dz}\left[\frac{z}{z-1}\right] &= \frac{(z-1)-z}{(z-1)^2} = \frac{-1}{(z-1)^2} \\
\frac{d}{dz}\left[\sum_{k=0}^{\infty} z^{-k}\right] &= \sum_{k=0}^{\infty} -kz^{-k-1} = -z^{-1}\sum_{k=0}^{\infty} kz^{-k}
\end{aligned}$$

Hence we have

$$\frac{-1}{(z-1)^2} = -z^{-1}\sum_{k=0}^{\infty} kz^{-k}$$

or

$$\sum_{k=0}^{\infty} kz^{-k} = \frac{z}{(1-z)^2}$$

which is the result we wanted. Just as with the z-transform of the unit step, the ROC for this function is $|z| > 1$.

<u>**Example.**</u> Determine the one-sided z-transform for the function $x(n) = a^n u(n)$. We have

$$\begin{aligned}
X(z) &= \sum_{k=-\infty}^{\infty} a^k u(k) z^{-k} \\
&= \sum_{k=0}^{\infty} \left(\frac{a}{z}\right)^k \\
&= \frac{1}{1-\frac{a}{z}} = \frac{z}{z-a}
\end{aligned}$$

The region of convergence is given by $|\frac{a}{z}| < 1$ or $|z| > |a|$.

<u>**Example.**</u> Determine the z-transform of the $x(t) = e^{-at}u(t)$ when $x(t)$ is sampled with sampling interval T. The sampled version of the signal is given by $x(nT) = e^{-anT}u(nT) = \left(e^{-aT}\right)^n u(n)$. Computing the z-transform we have

$$\begin{aligned}
X(z) &= \sum_{k=-\infty}^{\infty} \left(e^{-aT}\right)^k u(k) z^{-k} \\
&= \sum_{k=0}^{\infty} \left(\frac{1}{e^{aT}z}\right)^k \\
&= \frac{1}{1-\frac{1}{e^{aT}z}} \\
&= \frac{z}{z-e^{-aT}}
\end{aligned}$$

The region of convergence is given by $|\frac{1}{e^{aT}z}| < 1$ or $e^{-aT} < |z|$.

Example. Find the one-sided z-transform of $x(n) = a^n \cos(bnT)u(n)$. We have

$$X(z) = \sum_{k=-\infty}^{\infty} a^k \cos(bkT)u(k)z^{-k} = \sum_{k=0}^{\infty} \left(\frac{a}{z}\right)^k \cos(bkT)$$

At this point we need to use Euler's identity,

$$\cos(bkT) = \frac{e^{jbkT} + e^{-jbkT}}{2} = \frac{1}{2}\left(e^{jbT}\right)^k + \frac{1}{2}\left(e^{-jbT}\right)^k$$

Inserting this identity into our sum we have

$$\begin{aligned} X(z) &= \frac{1}{2}\sum_{k=0}^{\infty}\left(\frac{a}{z}e^{jbT}\right)^k + \frac{1}{2}\sum_{k=0}^{\infty}\left(\frac{a}{z}e^{-jbT}\right)^k \\ &= \frac{1}{2}\frac{1}{1-\frac{ae^{jbT}}{z}} + \frac{1}{2}\frac{1}{1-\frac{ae^{-jbT}}{z}} \end{aligned}$$

After some algebra we arrive at

$$X(z) = \frac{z(z - a\cos(bT))}{z^2 - 2a\cos(bT)z + a^2}$$

the region of convergence is given by

$$|\frac{ae^{jbT}}{z}| < 1, \quad |\frac{ae^{-jbT}}{z}| < 1$$

or $|z| > |a|$ since $|e^{jbT}| = |e^{-jbT}| = 1$.

1.5 z-Transform Properties

Just as with Laplace and Fourier transforms, there are a number of useful properties of z-transforms. We will cover a few of them in this section. Most of the properties we will cover are straightforward to prove, and you should be able to prove most of them. Unless otherwise stated, we will assume only one-sided z transforms.

Linearity. If we have the z-transform pairs $g(n) \leftrightarrow G(z)$ and $h(n) \leftrightarrow H(z)$, then for all constants a and b we have

$$\mathcal{Z}\left\{ag(n) + bh(n)\right\} = aG(z) + bH(z)$$

Multiplication by a^n. If $g(n) \leftrightarrow G(z)$, then $a^n g(n) \leftrightarrow G(\frac{z}{a})$.
Proof:

$$\begin{aligned} \mathcal{Z}\left\{a^n g(n)\right\} &= \sum_{k=0}^{\infty} a^k g(k) z^{-k} \\ &= \sum_{k=0}^{\infty} g(k)\left(\frac{z}{a}\right)^{-k} \\ &= G(\frac{z}{a}) \end{aligned}$$

Shifting. If $g(n) \leftrightarrow G(z)$ and $g(n)$ is causal, then

$$\begin{aligned}\mathcal{Z}\{g(n-l)\} &= z^{-l}G(z) \\ \mathcal{Z}\{g(n+l)\} &= z^{l}\left[G(z) - \sum_{k=0}^{l-1} g(k)z^{-k}\right]\end{aligned}$$

Proof of first part: Taking the z-transforms we have

$$\begin{aligned}\mathcal{Z}\{g(n-l)\} &= \sum_{k=0}^{\infty} g(k-l)z^{-k} \\ &= \sum_{k=l}^{\infty} g(k-l)z^{-k}\end{aligned}$$

since $g(n)$ is causal. Now let $m = k - l$, or $k = m + l$. The sum then becomes

$$\begin{aligned}\mathcal{Z}\{g(n-l)\} &= \sum_{m=0}^{\infty} g(m)z^{-(m+l)} \\ &= z^{-l}\sum_{m-0}^{\infty} g(m)z^{-m} \\ &= z^{-l}G(z)\end{aligned}$$

Proof of the second part: Let's start by trying to find the z-transform of $g(n+1)$. We have then

$$\begin{aligned}\mathcal{Z}\{g(n+1)\} &= \sum_{k=0}^{\infty} g(k+1)z^{-k} \\ &= g(1) + g(2)z^{-1} + g(3)z^{-2} + g(4)z^{-3} + g(5)z^{-4} + \dots\end{aligned}$$

Now we know

$$\begin{aligned}G(z) &= \sum_{k=0}^{\infty} g(k)z^{-k} \\ &= g(0) + g(1)z^{-1} + g(2)z^{-2} + g(3)z^{-3} + g(4)z^{-4} + g(5)z^{-5} + \dots\end{aligned}$$

Rearranging this expression we get

$$\begin{aligned}zG(z) &= g(0)z + g(1) + g(2)z^{-1} + g(3)z^{-2} + g(4)z^{-3} + g(5)z^{-4} + \dots \\ zG(z) - g(0) &= g(1) + g(2)z^{-1} + g(3)z^{-2} + g(4)z^{-3} + g(5)z^{-4} + \dots \\ &= \mathcal{Z}\{g(n+1)\}\end{aligned}$$

Now consider taking the z-transform of $g(n+2)$. We have

$$\begin{aligned}\mathcal{Z}\{g(n+2)\} &= \sum_{k=0}^{\infty} g(k+2)z^{-k} \\ &= g(2) + g(3)z^{-1} + g(4)z^{-2} + g(5)z^{-3} + \dots\end{aligned}$$

Starting from the definition of $G(z)$ from above

$$G(z) = g(0) + g(1)z^{-1} + g(2)z^{-2} + g(3)z^{-3} + g(4)z^{-4} + g(5)z^{-5} + \dots$$

we can find

$$\begin{aligned} z^2G(z) &= g(0)z^2 + g(1)z + g(2) + g(3)z^{-1} + g(4)z^{-2} + g(5)z^{-3} + \dots \\ z^2G(z) - z^2g(0) - zg(1) &= g(2) + g(3)z^{-1} + g(4)z^{-2} + g(5)z^{-3} + \dots \\ &= \mathcal{Z}\left\{g(n+2)\right\} \end{aligned}$$

In general we have

$$\begin{aligned} \mathcal{Z}\left\{g(n+l)\right\} &= \sum_{k=0}^{\infty} g(k+l)z^{-k} \\ &= g(l) + g(l+1)z^{-1} + g(l+2)z^{-2} + g(l+3)z^{-3} + \dots \end{aligned}$$

and

$$\begin{aligned} G(z) &= g(0) + g(1)z^{-1} + g(2)z^{-2} + \dots + g(l-1)z^{l-1} + g(l)z^{-l} + g(l+1)z^{-l-1} + g(l+2)z^{-l-2} + \dots \\ z^lG(z) &= z^lg(0) + z^{l-1}g(1) + z^{l-2}g(2) + \dots + z^{-1}g(l-1) + g(l) + g(l+1)z^{-1} + g(l+2)z^{-2} + \dots \end{aligned}$$

Rearranging we get

$$\begin{aligned} z^lG(z) - z^lg(0) - z^{l-1}g(1) + z^{l-2}g(2) + \dots + z^{-1}g(l-1) &= g(l) + g(l+1)z^{-1} + g(l+2)z^{-2} + \dots \\ z^l\left[G(z) - \sum_{k=0}^{l-1} g(k)z^{-k}\right] &= \mathcal{Z}\left\{g(n+l)\right\} \end{aligned}$$

Final Value Theorem. Just as with Laplace transforms, the Final Value Theorem is very useful for determining steady state errors in discrete-time control systems. If we assume $g(k)$ is causal, and all poles of $G(z)$ are inside the unit circle (with exception of a possible simple pole at $z = 1$), then

$$\lim_{t\to\infty} x(t) = \lim_{z\to 1}\left[1 - z^{-1}\right]G(z) = \lim_{z\to 1}\frac{z-1}{z}G(z)$$

Example. If $g(t) = (1 - e^{-3t})u(t)$, then $G(z) = \frac{z(1-e^{-3T})}{(z-1)(z-e^{-3T})}$. By the Final Value Theorem we have

$$\lim_{t\to\infty} g(t) = 1 = \lim_{z\to 1}\frac{z-1}{z}G(z) = \frac{1-e^{-3T}}{1-e^{-3T}} = 1$$

since all of the poles of the system are within the unit circle with the exception of a simple pole at $z = 0$.

Example. If $g(t) = e^{-t}u(t)$, then $G(z) = \frac{z}{z-e^{-T}}$. By the Final Value Theorem

$$\lim_{t\to\infty} g(t) = 0 = \lim_{z\to 1}\frac{z-1}{z}G(z) = 0$$

since all of the poles of the system are within the unit circle.

Example. If $g(t) = u(t)$, then $G(z) = \frac{z}{z-1}$. By the Final Value Theorem

$$\lim_{t \to \infty} g(t) = 1 = \lim_{z \to 1} \frac{z-1}{z} G(z) = 1$$

Note that in this example there is a simple pole at $z = 1$.

Initial Value Theorem. Just as with laplace transforms, the Initial Value Theorem for z-transforms is very useful for determining the initial required control effort for different types of controllers. The Initial Value Theorem states that if the limits exist, then

$$\lim_{t \to 0} g(t) = \lim_{z \to \infty} G(z)$$

Example. If $g(t) = (1 - e^{-3t})u(t)$, then $G(z) = \frac{z(1-e^{-3T})}{(z-1)(z-e^{-3T})}$. By the Initial Value Theorem we have

$$\lim_{t \to 0} g(t) = 0 = \lim_{z \to \infty} G(z) = 0$$

Example. If $g(t) = e^{-t}u(t)$, then $G(z) = \frac{z}{z-e^{-T}}$. By the Initial Value Theorem

$$\lim_{t \to 0} g(t) = 1 = \lim_{z \to \infty} G(z) = 1$$

Table 1 summarizes some common z-transform pairs, while Table 2 summarizes some common z-transform properties.

1.6 Inverse z-Transforms

There are four common methods for going from the z-domain back to the (discrete) time domain. These are

- Computational methods (Matlab/Maple, etc)
- Contour integration in the complex plane
- Partial fractions
- Long division

In this course we will emphasize the use of partial fractions, which you should be pretty familiar with. It is generally easiest to find the partial fraction expansion for $\frac{G(z)}{z}$ when you want to find the inverse z-transform of $G(z)$, since simple poles have the form $a^k u(k) \leftrightarrow \frac{z}{z-a}$.

Example. Determine $g(n)$ for $G(z) = \frac{z(z-4)}{(z-3)(z-2)}$. We have

$$\frac{G(z)}{z} = \frac{z-4}{(z-3)(z-2)} = \frac{A}{z-3} + \frac{B}{z-2}$$

$X(s)$	$x(t)$	$x(nT)$ or $x(n)$	$X(z)$
—	—	$\delta(n)$	1
—	—	$\delta(n-l)$	z^{-l}
$\frac{1}{s}$	u(t)	$u(nT)$	$\frac{z}{z-1}$
$\frac{1}{s+a}$	$e^{-at}u(t)$	$e^{-anT}u(nT)$	$\frac{z}{z-e^{-aT}}$
$\frac{1}{s^2}$	$tu(t)$	$(nT)u(nT)$	$\frac{Tz}{(z-1)^2}$
$\frac{1}{s^3}$	$\frac{1}{2}t^2u(t)$	$\frac{1}{2}(nT)^2u(nT)$	$\frac{T^2z(z+1)}{(z-1)^3}$
$\frac{a}{s(s+a)}$	$(1-e^{-at})u(t)$	$(1-e^{-anT})u(nT)$	$\frac{z(1-e^{-aT})}{(z-1)(z-e^{-aT})}$
$\frac{b-a}{(s+a)(s+b)}$	$(e^{-at}-e^{-bt})u(t)$	$(e^{-anT}-e^{-bnT})u(nT)$	$\frac{z(e^{-aT}-e^{-bT})}{(z-e^{-aT})(z-e^{-bT})}$
$\frac{1}{(s+a)^2}$	$te^{-at}u(t)$	$(nT)e^{-anT}u(nT)$	$\frac{zTe^{-aT}}{(z-e^{-aT})^2}$
$\frac{s}{(s+a)^2}$	$(1-at)e^{-at}u(t)$	$(1-anT)e^{-anT}u(nT)$	$\frac{z[z-(1+aT)e^{-aT}]}{(z-e^{-aT})^2}$
$\frac{\omega}{s^2+\omega^2}$	$\sin(\omega t)u(t)$	$\sin(\omega nT)u(nT)$	$\frac{z\sin(\omega T)}{z^2-2z\cos(\omega T)+1}$
$\frac{s}{s^2+\omega^2}$	$\cos(\omega t)u(t)$	$\cos(\omega nT)u(nT)$	$\frac{z[z-\cos(\omega T)]}{z^2-2\cos(\omega T)+1}$
		$b^nu(n)$	$\frac{z}{z-b}$
		$nb^nu(n)$	$\frac{bz}{(z-b)^2}$
		$(1-b^n)u(n)$	$\frac{z(1-b)}{(z-1)(z-b)}$

Table 1: Common (one sided) z-transforms. Often we set $T = 1$ and $e^{-aT} = b$ to find results when the sampling time is not important.

$x(t)$	$x(nT)$	$\mathcal{Z}[x(t)]$ or $\mathcal{Z}[x(nT)]$
$\alpha x(t)$	$\alpha x(nT)$	$\alpha X(z)$
$\alpha x_1(t)+\beta x_2(t)$	$\alpha x_1(nT)+\beta x_2(nT)$	$\alpha X_1(z)+\beta X_2(z)$
$x(t+T)$	$x(nT+T)$	$zX(z)-zx(0)$
$x(t+2T)$	$x(nT+2T)$	$z^2X(z)-z^2x(0)-zx(T)$
$x(t+kT)$	$x(nT+kT)$	$z^kX(z)-z^kx(0)-z^{k-1}x(T)-z^{k-2}X(2T)-\ldots-zx(kT-T)$
$x(t-kT)$	$x(nT-kT)$	$z^{-k}X(z)$
$tx(t)$	$(nT)x(nT)$	$-Tz\frac{d}{dz}X(z)$
$e^{-at}x(t)$	$e^{-anT}x(nT)$	$X(ze^{aT})$
—	$a^nx(nT)$	$X(\frac{z}{a})$
$x(0)$	$x(0)$	$\lim_{z\to\infty}X(z)$ if the limit exists
$x(\infty)$	$x(\infty)$	$\lim_{z\to 1}[(1-z^{-1}]X(z)$ if all poles of $X(z)$ are inside unit the unit circle (a simple pole at $z=1$ is allowed)
—	$\sum_{k=0}^{n}x(k)$	$\frac{z}{z-1}X(z)$
$x(t)\star h(t)$	$x(nT)\star h(nT)$	$X(z)H(z)$

Table 2: Common properties of (one sided) z-transforms. Often we set $T = 1$ and $e^{-aT} = b$ to find results when the sampling time is not important

Using the "cover up" method we quickly arrive at

$$A = \frac{3-4}{3-2} = -1 \quad B = \frac{2-4}{2-3} = 2$$

This leads to

$$\frac{G(z)}{z} = \frac{-1}{z-3} + \frac{2}{z-2}$$

or

$$G(z) = -\frac{z}{z-3} + 2\frac{z}{z-2}$$

From this form we can quickly conclude that

$$g(n) = -(3)^n u(n) + 2(2)^n u(n)$$

The first few terms of this impulse response are $g(0) = 1$, $g(1) = 1$, $g(2) = -1$, and $g(3) = -11$.

Example. Determine $g(n)$ if $G(z) = \frac{z-4}{(z-3)(z-2)}$. For this $G(z)$, there is no z in the numerator to divide by to create the $\frac{G(z)}{z}$. One solution is to multiple by $\frac{z}{z}$ and then do the partial fractions, as follows:

$$G(z) = \frac{z-4}{(z-3)(z-2)} = \frac{z-4}{(z-3)(z-2)}\frac{z}{z}$$

so we have

$$\frac{G(z)}{z} = \frac{z-4}{z(z-3)(z-2)} = \frac{A}{z} + \frac{B}{z-3} + \frac{C}{z-2}$$

Using the "cover up" method we get

$$A = \frac{-4}{(-3)(-2)} = \frac{-2}{3}, \quad B = \frac{3-4}{(3-2)(3)} = \frac{-1}{3}, \quad C = \frac{2-4}{(2-3)(2)} = 1$$

so

$$G(z) = -\frac{2}{3} - \frac{1}{3}\frac{z}{z-3} + \frac{z}{z-2}$$

which yields

$$g(n) = -\frac{2}{3}\delta(n) - \frac{1}{3}(3)^n u(n) + (2)^n u(n)$$

If we compute the first few terms of $g(n)$ we get $g(0) = 0$, $g(1) = 1$, $g(2) = 1$, $g(3) = -1$, $g(4) = -11$.

Example. Determine $g(n)$ if $G(z) = \frac{z-4}{(z-3)(z-2)}$. For this $G(z)$, there is no z in the numerator to divide by to create the $\frac{G(z)}{z}$. One solution is to construct and $F(z)$ so that $G(z) = z^{-1}F(z)$, as follows:

$$F(z) = \frac{z(z-4)}{(z-3)(z-2)}$$

so we have

$$\frac{F(z)}{z} = \frac{z-4}{(z-3)(z-2)} = \frac{A}{z-3} + \frac{B}{z-2}$$

Using the "cover up" method we get

$$F(z) = -\frac{z}{z-3} + 2\frac{z}{z-2}$$

and

$$f(n) = -(3)^n u(n) + 2(2)^n u(n)$$

Since we know $G(z) = z^{-1}F(z)$, we have $g(n) = f(n-1)$ using the delay property. Hence

$$g(n) = -(3)^{n-1}u(n-1) + 2(2)^{n-1}u(n-1)$$

Although this answer looks much different than in the previous example, the two solutions are in fact the same.

Example. Determine $g(n)$ for $G(z) = \frac{z(-5z+22)}{(z+1)(z-2)^2}$. We have

$$\frac{G(z)}{z} = \frac{-5z+22}{(z+1)(z-2)^2} = \frac{A}{z+1} + \frac{B}{z-2} + \frac{C}{(z-2)^2}$$

Using the "cover-up" method we can quickly find A and C,

$$A = \frac{-5(-1)+22}{(-1-2)^2} = \frac{27}{9} = 3, \quad C = \frac{-5(2)+22}{(2+1)} = \frac{12}{3} = 4$$

To get B, we multiply both sides of the original equation by z and then let $z \to \infty$, to get

$$0 = A + B, \quad B = -3$$

So we have

$$G(z) = 3\frac{z}{z+1} - 3\frac{z}{z-2} + 4\frac{z}{(z-2)^2}$$

and

$$g(n) = 3(-1)^n u(n) - 3(2)^n u(n) + 4n(2)^{n-1}u(n)$$

The first few terms of the impulse response are $g(0) = 0$, $g(1) = -5$, $g(2) = 7$, $g(3) = 21$, and $g(4) = 83$.

While we can normally use partial fractions to determine the time response for all sample times, sometimes we only want to determine the time response for the first few terms. We can use the method of long division to determine the first few terms of the time domain response by utilizing the fact that if we compute the z-transform of $g(n)$ we have

$$G(z) \quad = \quad g(0) + g(1)z^{-1} + g(2)z^{-2} + g(3)z^{-3} + g(4)z^{-4} + \ldots$$

This method of long division is particularly important as a method to check your answer. However, it is not very useful for determining a closed form solution for all time.

Example. Use long division to determine the first few terms of the impulse response for the transfer function $G(z) = \frac{z(z-4)}{(z-3)(z-2)}$. We first need to multiply out the numerator and the denominator, so we have $G(z) = \frac{z^2-4z}{z^2-5z+6}$. Next we do the long division,

$$
\begin{array}{r|lllll}
 & 1 & +z^{-1} & -z^{-2}-11z^{-3} & & \\
\cline{2-6}
z^2-5z+6 & z^2 & -4z & & & \\
 & z^2 & -5z & +6 & & \\
\cline{2-4}
 & & z & -6 & & \\
 & & z & -5 & +6z^{-1} & \\
\cline{3-5}
 & & & -1 & -6z^{-1} & \\
 & & & -1 & +5z^{-1} & -6z^{-2} \\
\cline{4-6}
 & & & & -11z^{-1} & +6z^{-2}
\end{array}
$$

Hence we have the approximations

$$
\begin{aligned}
G(z) &= \frac{z(z-4)}{z^2-5z+6} = 1 + z^{-1} - z^{-2} - 11z^{-3} + \cdots \\
g(n) &= \delta(n) + \delta(n-1) - \delta(n-2) - 11\delta(n-3) + \cdots
\end{aligned}
$$

which agrees with our previous result.

Example. Use long division to determine the first few terms of the impulse response for the transfer function $G(z) = \frac{z-4}{(z-3)(z-2)}$. We first need to multiply out the denominator, so we have $G(z) = \frac{z-4}{z^2-5z+6}$. Next we do the long division,

$$
\begin{array}{r|lllll}
 & z^{-1} & +z^{-2} & -z^{-3}-11z^{-4} & & \\
\cline{2-6}
z^2-5z+6 & z & -4 & & & \\
 & z & -5 & +6z^{-1} & & \\
\cline{2-4}
 & & 1 & -6z^{-1} & & \\
 & & 1 & -5z^{-1} & +6z^{-2} & \\
\cline{3-5}
 & & & -z^{-1} & -6z^{-2} & \\
 & & & -z^{-1} & +5z^{-2} & -6z^{-3} \\
\cline{4-6}
 & & & & -11z^{-2} & +6z^{-3}
\end{array}
$$

Hence we have the approximations

$$
\begin{aligned}
G(z) &= \frac{z-4}{z^2-5z+6} = z^{-1} + z^{-2} - z^{-3} - 11z^{-4} + \cdots \\
g(n) &= \delta(n-1) + \delta(n-2) - \delta(n-3) - 11\delta(n-4) + \cdots
\end{aligned}
$$

which agrees with our previous result.

Example. Use long division to determine the first few terms of the impulse response for the transfer function $G(z) = \frac{z(-5z+22)}{(z+1)(z-2)^2}$. We first need to multiply out the numerator and denominator, so we have $G(z) = \frac{-5z^2+22z}{z^3-3z^{-2}+4}$. Next we do the long division,

$$
\begin{array}{r|llllll}
 & -5z^{-1} & +7z^{-2} & \multicolumn{2}{l}{+21z^{-3}+83z^{-4}} & & \\
\cline{2-7}
z^3-3z^2+4 & -5z^2 & +22z & & & & \\
 & -5z^2 & +15z & & -20z^{-1} & & \\
\cline{2-5}
 & & 7z & & +20z^{-1} & & \\
 & & 7z & -21 & & +28z^{-2} & \\
\cline{3-6}
 & & & 21 & +20z^{-1} & -28z^{-2} & \\
 & & & 21 & -63z^{-1} & & +84z^{-3} \\
\cline{4-7}
 & & & & 83z^{-1} & -28z^{-3} & -84z^{-3}
\end{array}
$$

Hence we have the approximations

$$
\begin{aligned}
G(z) &= \frac{z(-5z+22)}{z^3-3z^2+4} = -5z^{-1}+7z^{-2}+21z^{-3}+83z^{-4}+\cdots \\
g(n) &= -5\delta(n-1)+7\delta(n-2)+21\delta(n-3)+83\delta(n-4)+\cdots
\end{aligned}
$$

which agrees with our previous result.

1.7 Second Order Transfer Functions with Complex Conjugate Poles

Consider the continuous-time impulse response response that corresponds to a transfer function with complex conjugate poles at $-a \pm jb$,

$$
G(s) = \frac{r}{(s+a)^2+b^2} \quad \leftrightarrow \quad g(t) = re^{-at}\cos(bt+\theta)u(t)
$$

If we look at samples of the impulse response, we get

$$
\begin{aligned}
g(nT) &= re^{-anT}\cos(bnT+\theta)u(nT) \\
&= r\left(e^{-aT}\right)^n \cos((bT)n+\theta)u(nT) \\
&= r\gamma^n\cos(\beta n+\theta)u(n)
\end{aligned}
$$

Hence, the discrete-time impulse response that corresponds to a continuous-time system with complex conjugate poles is of the form

$$
g(n) \;=\; r\gamma^n\cos(\beta n+\theta)u(n)
$$

As you might expect, this form shows up a great deal when we are modeling many types of systems. Hence we should know the corresponding z-transform of this impulse response. As you will see, it is much more complex than you might expect.

Using Euler's identity we can write $g(n)$ as

$$
g(n) \;=\; \frac{r}{2}\gamma^n e^{j\beta n}e^{j\theta} + \frac{r}{2}\gamma^n e^{-j\beta n}e^{-j\theta}
$$

and taking the z-transform of this we have

$$\begin{aligned}
G(z) &= \frac{r}{2}\sum_{k=0}^{\infty}\gamma^k e^{j\beta k}e^{j\theta}z^{-k} + \frac{r}{2}\sum_{k=0}^{\infty}\gamma^k e^{-j\beta k}e^{-j\theta}z^{-k} \\
&= \frac{r}{2}e^{j\theta}\sum_{k=0}^{\infty}\left(\frac{\gamma e^{j\beta}}{z}\right)^k + \frac{r}{2}e^{-j\theta}\sum_{k=0}^{\infty}\left(\frac{\gamma e^{-j\beta}}{z}\right)^k \\
&= \frac{re^{j\theta}}{2}\left[\frac{1}{1-\gamma e^{j\beta}z^{-1}}\right] + \frac{re^{-j\theta}}{2}\left[\frac{1}{1-\gamma e^{-j\beta}z^{-1}}\right] \\
&= \frac{re^{j\theta}}{2}\left[\frac{z}{z-\gamma e^{j\beta}}\right] + \frac{re^{-j\theta}}{2}\left[\frac{z}{z-\gamma e^{-j\beta}}\right] \\
&= \frac{re^{j\theta}}{2}\left[\frac{z(z-\gamma e^{-j\beta})}{(z-\gamma e^{j\beta})(z-\gamma e^{-j\beta})}\right] + \frac{re^{-j\theta}}{2}\left[\frac{z(z-\gamma e^{j\beta})}{(z-\gamma e^{j\beta})(z-\gamma e^{-j\beta})}\right] \\
&= \frac{r}{2}\left[\frac{z^2e^{j\theta} - z\gamma e^{-j\beta}e^{j\theta} + z^2e^{-j\theta} - z\gamma e^{j\beta}e^{-j\theta}}{z^2 - \gamma z e^{j\beta} - \gamma z e^{-j\beta} + \gamma^2}\right] \\
&= \frac{r}{2}\left[\frac{z^2\left(e^{j\theta}+e^{-j\theta}\right) - z\gamma\left(e^{j(\beta-\theta)}+e^{-(\beta-\theta)}\right)}{z^2 - \gamma z\left(e^{j\beta}+e^{-j\beta}\right) + \gamma^2}\right] \\
&= r\left[\frac{z^2\cos(\theta) - z\gamma\cos(\beta-\theta)}{z^2 - 2\gamma z\cos(\beta) + \gamma^2}\right]
\end{aligned}$$

Unfortunately, what we really need is a form that we can (easily?) use in doing partial fractions. This is going to take a bit more work. Let's write

$$G(z) = r\left[\frac{z^2\cos(\theta) - z\gamma\cos(\beta-\theta)}{z^2 - 2\gamma z\cos(\beta) + \gamma^2}\right] = \frac{Az^2+Bz}{z^2+2Cz+\gamma^2}$$

This second form is much easier to use when doing partial fractions, but we need to be able to relate the parameters A, B, and C, to the parameters r, θ, and β. Clearly we have already identified the parameter γ.

Let's start with the denominator. Equating powers of z in the denominator we get

$$-2C = 2\gamma\cos(\beta)$$

which means

$$\beta = \cos^{-1}\left(\frac{-C}{\gamma}\right)$$

Next we identify

$$\begin{aligned}
A &= r\cos(\theta) \\
B &= -r\gamma\cos(\beta-\theta)
\end{aligned}$$

Expanding the cosine expression out we get

$$\begin{aligned}
B &= -r\gamma\cos(\beta-\theta) \\
&= -r\gamma\cos(\beta)\cos(\theta) - r\gamma\sin(\beta)\sin(\theta) \\
&= -\gamma\cos(\beta)\left(r\cos(\theta)\right) - r\gamma\sin(\beta)\sin(\theta) \\
&= CA - r\gamma\sin(\beta)\sin(\theta)
\end{aligned}$$

But

$$\begin{aligned}\frac{-C}{\gamma} &= \cos(\beta)\\ \sin(\beta) &= \sqrt{1-\cos^2(\beta)} = \sqrt{1-\frac{C^2}{\gamma^2}} = \frac{1}{\gamma}\sqrt{\gamma^2-C^2}\end{aligned}$$

So

$$B = CA - \gamma\left(\frac{1}{\gamma}\sqrt{\gamma^2-C^2}\right) r\sin(\theta)$$

or

$$r\sin(\theta) = \frac{CA-B}{\sqrt{\gamma^2-C^2}}$$

Recall that we also have $r\sin(\theta) = A$, so dividing these we get

$$\tan(\theta) = \frac{\sin(\theta)}{\cos(\theta)} = \frac{CA-B}{A\sqrt{\gamma^2-C^2}}$$

or

$$\theta = \tan^{-1}\left(\frac{CA-B}{A\sqrt{\gamma^2-C^2}}\right)$$

Finally we have

$$\begin{aligned}\sqrt{(r\sin(\theta))^2+(r\cos(\theta))^2} &= r = \sqrt{\left(\frac{CA-B}{\sqrt{\gamma^2-C^2}}\right)^2 + A^2}\\ &= \sqrt{\frac{A^2C^2-2ACB+B^2+A^2\gamma^2-C^2A^2}{\gamma^2-C^2}}\end{aligned}$$

or

$$r = \sqrt{\frac{A^2\gamma^2+B^2-2ABC}{\gamma^2-C^2}}$$

In summary we have

$$\begin{aligned}g(n) = r\gamma^n\cos(\beta n+\theta)u(n) \;\;&\leftrightarrow\;\; G(z) = \frac{Az^2+Bz}{z^2+2Cz+\gamma^2}\\ r &= \sqrt{\frac{A^2\gamma^2+B^2-2ABC}{\gamma^2-C^2}}\\ \theta &= \tan^{-1}\left(\frac{CA-B}{A\sqrt{\gamma^2-C^2}}\right)\\ \beta &= \cos^{-1}\left(\frac{-C}{\gamma}\right)\end{aligned}$$

Now we have all of the parameters we need to solve some examples.

Example. Determine the impulse and unit step responses of the system with transfer function

$$G(z) \;=\; \frac{-0.535z}{z^2 - 1.597z + 0.671}$$

To determine the impulse response, we first equate this transfer function with our standard form,

$$G(z) \;=\; \frac{-0.535z}{z^2 - 1.597z + 0.671} = \frac{Az^2 + Bz}{z^2 + 2Cz + \gamma^2}$$

From this we can immediately determine $A = 0$, $B = -0.535$, $\gamma = \sqrt{0.671} = 0.819$, $-2C = 1.597$ so $C = -0.7895$. Now we compute the remaining parameters,

$$\begin{aligned}
\beta &= \cos^{-1}\left(\frac{-C}{\gamma}\right) = \cos^{-1}\left(\frac{0.7985}{0.819}\right) = 0.224 \\
\theta &= \tan^{-1}\left(\frac{CA - B}{A\sqrt{\gamma^2 - C}}\right) = \tan^{-1}\left(\frac{0.535}{0}\right) = \frac{\pi}{2} = 1.570 \\
r &= \sqrt{\frac{A^2\gamma^2 + B^2 - 2ABC}{\gamma^2 - C^2}} = \sqrt{\frac{0.2862}{0.03340}} = 2.9274
\end{aligned}$$

Hence the impulse response of the system is

$$g(n) = 2.927(0.819^n)\cos(0.224n + 1.570)u(n)$$

The initial value of this system is $g(0) = 2.450\cos(1.570) = 0$, so the system is initially at rest.

To determine the unit step response, we have

$$\begin{aligned}
Y(z) &= G(z)\frac{z}{z-1} \\
\frac{Y(z)}{z} &= G(z)\frac{1}{z-1} = \frac{-0.535z}{(z^2 - 1.597z + 0.671)(z-1)} = \frac{Az + B}{z^2 - 1.597z + 0.671} + \frac{D}{z-1}
\end{aligned}$$

Using the cover-up method we can easily determine that $D = -7.230$. Multiplying both sides of the equation by z and letting $z \to \infty$, we get $A = -D = 7.230$. Finally, setting $z = 0$ we get $B = -4.851$. Now we need to recompute our parameters

$$\begin{aligned}
\beta &= \cos^{-1}\left(\frac{-C}{\gamma}\right) = 0.224 \\
\theta &= \tan^{-1}\left(\frac{CA - B}{A\sqrt{\gamma^2 - C}}\right) = -0.6110 \\
r &= \sqrt{\frac{A^2\gamma^2 + B^2 - 2ABC}{\gamma^2 - C^2}} = 8.827
\end{aligned}$$

The step response is then

$$y(n) \;=\; 8.827(0.819^n)\cos(0.224n - 0.6110)u(n) - 7.230u(n)$$

1.8 Solving Difference Equations

Just as we can use Laplace transforms to solve differential equations with initial conditions, in discrete-time we can use z-transforms to solve difference equations with initial conditions. The general idea is to take the z-transform of each term of the difference equation, then solve. For the low order difference equations we will be solving it is useful to summarize the following identities again:

$$\begin{aligned}
\mathcal{Z}\{x(k+2)\} &= z^2X(z) - z^2x(0) - zx(1) \\
\mathcal{Z}\{x(k+1)\} &= zX(z) - zx(0) \\
\mathcal{Z}\{x(k)\} &= X(z) \\
\mathcal{Z}\{x(k-1)\} &= z^{-1}X(z) \\
\mathcal{Z}\{x(k-2)\} &= z^{-2}X(z)
\end{aligned}$$

It is important to remember that the lower case letters represent the time-domain and the capitol letters represent the z-domain. After we have solved the difference equation, it is a good idea to check our answer by using the difference equation to compute the answer and compare this with the answer we get using the z-transform. Finally, sometimes we break the solution into two different parts: the *Zero Input Response (ZIR)* is the response of the system due to initial conditions alone (there is no, or zero input), and the *Zero State Response (ZSR)* is the response of the system due to the input only, assuming all initial conditions (all initial states) are zero. The system transfer function is determined from the ZSR.

Example. Find the ZIR, ZSR, and solve the difference equation

$$y(n+2) - 5y(n+1) + 6y(n) = x(n)$$

with initial conditions $y(0) = 4$ and $y(1) = 3$ and input $x(n) = u(n)$, the unit step function. Taking z-transforms of each term, we have

$$\left[z^2Y(z) - z^2y(0) - zy(1)\right] - 5\left[zY(z) - zy(0)\right] + 6Y(z) = X(z)$$

Note that we have not simplified the difference equation yet. Rearranging this equation we get

$$\left[z^2 - 5z + 6\right]Y(z) = X(z) + \left[z^2y(0) + zy(1) - 5zy(0)\right]$$

or

$$Y(z) = \underbrace{\frac{X(z)}{(z-3)(z-2)}}_{ZSR} + \underbrace{\frac{z^2y(0) + zy(1) - 5zy(0)}{(z-3)(z-2)}}_{ZIR}$$

At this point we can determine the transfer function from the ZSR and corresponding impulse response,

$$H(z) = \frac{1}{(z-3)(z-2)}, \quad h(n) = 3^{n-1}u(n-1) - 2^{n-1}u(n-1) = \frac{1}{6}\delta(n) + \frac{1}{3}3^nu(n) - \frac{1}{2}2^nu(n)$$

The ZSR is then given by

$$Y_{ZSR} = \frac{X(z)}{(z-3)(z-2)} = \frac{z}{(z-1)(z-2)(z-3)} = \frac{1}{2}\frac{z}{z-1} - \frac{z}{z-2} + \frac{1}{2}\frac{z}{z-3}$$

or

$$y_{ZSR}(n) = \frac{1}{2}u(n) - 2^n u(n) + \frac{1}{2}3^n u(n)$$

The ZIR is given by

$$Y(z) = \frac{z^2 y(0) + zy(1) - 5zy(0)}{(z-3)(z-2)} = \frac{4z^2 - 17z}{(z-3)(z-2)} = -5\frac{z}{z-3} + 9\frac{z}{z-2}$$

or

$$y_{ZIR}(n) = -5(3^n)u(n) + 9(2^n)u(n)$$

Finally the total solution is given by the sum of the ZIR and ZSR,

$$\begin{aligned} y(n) &= y_{ZIR}(n) + y_{ZSR}(n) \\ &= [-5(3^n)u(n) + 9(2^n)u(n)] + \left[\frac{1}{2}u(n) - 2^n u(n) + \frac{1}{2}3^n u(n)\right] \\ &= \left[(-5 + \frac{1}{2})3^n + (9-1)2^n + \frac{1}{2}\right] u(n) \\ &= \left[-4.5(3^n) + 8(2^n) + \frac{1}{2}\right] u(n) \end{aligned}$$

Now to check our answer, we know that $y(0) = 4$ and $y(1) = 3$. To generate the other values of $y(n)$, let's rewrite the difference equation as

$$y(n+2) = 5y(n+1) - 6y(n) + x(n)$$

and remember that $x(n) = u(n) = 1$ for $n \geq 0$. Now we can make a table,

DE Prediction	*Closed Form*
$y(0) = 4$	$y(0) = -4.5(3^0) + 8(2^0) + 0.5 = 4$
$y(1) = 3$	$y(1) = -4.5(3) + 8(2) + 0.5 = 3$
$y(2) = 5y(1) - 6y(0) + 1 = 5(3) - 6(4) + 1 = -8$	$y(2) = -4.5(3^2) + 8(2^2) + 0.5 = -8$
$y(3) = 5y(2) - 6y(1) + 1 = 5(-8) - 6(3) + 1 = -57$	$y(3) = -4.5(3^3) + 8(2^3) + 0.5 = -57$
$y(4) = 5y(3) - 6y(2) + 1 = 5(-57) - 6(-8) + 1 = -236$	$y(4) = -4.5(3^4) + 8(2^4) + 0.5 = -236$

Example. Find the ZIR, ZSR, and solve the difference equation

$$x(n+2) - x(n+1) + 0.25x(n) = f(n-1) - f(n)$$

with initial conditions $x(0) = 0$ and $x(1) = 1$ and input $f(n) = u(n)$, the unit step function. *Note that we must write the difference equation so that the input does not include initial conditions!* We assume all *initial conditions* are associated with the states of the system, in this case the $x(n)$. Taking z-transforms of each term, we have

$$\left[z^2 X(z) - z^2 x(0) - zx(1)\right] - [zX(z) - zx(0)] + 0.25X(z) = z^{-1}F(z) - F(z)$$

Note that we have not simplified the difference equation yet. Rearranging this equation we get

$$\left[z^2 - z + \frac{1}{4}\right] X(z) = \left[z^{-1} - 1\right] F(z) + \left[z^2 x(0) + zx(1) - zx(0)\right]$$

or

$$X(z) = \underbrace{F(z)\frac{z^{-1}-1}{(z-\frac{1}{2})^2}}_{ZSR} + \underbrace{\frac{z^2x(0)+zx(1)-zx(0)}{(z-\frac{1}{2})^2}}_{ZIR}$$

At this point we can determine the transfer function from the ZSR and corresponding impulse response,

$$H(z) = \frac{z^{-1}-1}{(z-\frac{1}{2})^2} = \frac{1-z}{z(z-\frac{1}{2})^2}, \quad h(n) = 4\delta(n-1) - 4(\frac{1}{2})^{n-1}u(n-1) + (n-1)(\frac{1}{2})^{n-2}u(n-2)$$

The ZSR is then given by

$$X_{ZSR} = \frac{X(z)(z^{-1}-1)}{(z-\frac{1}{2})^2} = \frac{z(z^{-1}-1)}{(z-\frac{1}{2})^2} = -\frac{1}{(z-\frac{1}{2})^2} = -(n-1)(\frac{1}{2})^{n-2}u(n-1)$$

The ZIR is given by

$$X(z) = \frac{z^2x(0)+zx(1)-zx(0)}{(z-\frac{1}{2})^2} = \frac{z}{(z-\frac{1}{2})^2} = n(\frac{1}{2})^{n-1}u(n)$$

Finally the total solution is given by the sum of the ZIR and ZSR,

$$\begin{aligned} y(n) &= y_{ZIR}(n) + y_{ZSR}(n) \\ &= \left[n(\frac{1}{2})^{n-1}u(n)\right] - \left[(n-1)(\frac{1}{2})^{n-2}u(n-1)\right] \\ &= (\frac{1}{2})^{n-2}\left[1-\frac{n}{2}\right]u(n-2) + \delta(n-1) \end{aligned}$$

Now to check our answer, we know that $x(0) = 0$ and $x(1) = 1$. To generate the other values of $x(n)$, let's rewrite the difference equation as

$$x(n+2) = x(n+1) - 0.25x(n) + f(n-1) - f(n)$$

and remember that $f(n) = u(n) = 1$ for $n \geq 0$. Now we can make a table,

DE Prediction	*Closed Form*
$x(0) = 0$	$x(0) = 0$
$x(1) = 1$	$x(1) = 1$
$x(2) = x(1) - \frac{1}{4}x(0) + 0 - 1 = 1 - 0 + 0 - 1 = 0$	$x(2) = (\frac{1}{2})^0\left[1-\frac{2}{2}\right] = 0$
$x(3) = x(2) - \frac{1}{4}x(1) + 1 - 1 = -\frac{1}{4}$	$x(3) = (\frac{1}{2})^1\left[1-\frac{3}{2}\right] = -\frac{1}{4}$
$x(4) = x(3) - \frac{1}{4}x(2) + 1 - 1 = -\frac{1}{4}$	$x(4) = (\frac{1}{2})^2\left[1-\frac{4}{2}\right] = -\frac{1}{4}$
$x(5) = x(4) - \frac{1}{4}x(3) + 1 - 1 = -\frac{1}{4} + (\frac{1}{4})^2 = -3(\frac{1}{4})^2$	$x(5) = (\frac{1}{2})^3\left[1-\frac{5}{2}\right] = -3(\frac{1}{2})^4$

1.9 Asymptotic Stability

In continuous-time, asymptotic stability is based on the behavior of the impulse response as $t \rightarrow \infty$. There are three cases:

- If $|h(t)| \rightarrow \infty$ as $t \rightarrow \infty$, then the system is *unstable*. This means at least one pole of the system is in the (open) right half plane.
- If $|h(t)| \rightarrow 0$ as $t \rightarrow \infty$, then the system is *stable*. This means all of the poles of the system are in the (open) left half plane.
- If $|h(t)| \leq M$ for some constant M as $t \rightarrow \infty$, then the system is *marginally stable*. This means there are some isolated poles on the $j\omega$ axis and, if there are poles not on the $j\omega$ axis these poles are in the left half plane.

We can easily modify these criteria to discrete-time, and look at the behavior of the impulse response as $k \rightarrow \infty$:

- If $|h(k)| \rightarrow \infty$ as $k \rightarrow \infty$, then the system is *unstable*. This means at least one pole of the system is outside the unit circle (at least one pole has a magnitude larger than one.)
- If $|h(k)| \rightarrow 0$ as $k \rightarrow \infty$, then the system is *stable*.This means all of the poles of the system are inside the unit circle (all poles have a magnitude less than one).
- If $|h(k)| \leq M$ for some constant M as $k \rightarrow \infty$, then the system is *marginally stable*. This means there are some isolated poles on the unit circle and, if there are poles not on the unit circle these poles are inside the unit circle.

1.10 Mapping Poles and Settling Time

Let's assume we have the continuous-time time transfer function

$$H(s) = \frac{1}{(s+a)(s+b)}$$

with corresponding impulse response

$$h(t) = \frac{1}{b-a}e^{-at}u(t) + \frac{1}{a-b}e^{-bt}u(t)$$

Now we will assume we sample this transfer function with sampling interval T, so we have

$$h(kT) = \frac{1}{b-a}e^{-akT}u(kT) + \frac{1}{a-b}e^{-bkT}u(kT)$$

Taking z-transforms of this we get

$$\begin{aligned} H(z) &= \frac{1}{b-a}\frac{z}{z-e^{-aT}} + \frac{1}{a-b}\frac{z}{z-e^{-bT}} \\ &= \frac{1}{b-a}\left[\frac{z}{z-e^{-aT}} - \frac{z}{z-e^{-bT}}\right] \\ &= \frac{1}{b-a}\frac{z(z-e^{-bT}) - z(z-e^{-aT})}{(z-e^{-aT})(z-e^{-bT})} \\ &= \frac{1}{b-a}\frac{z(e^{-aT}-e^{-bT})}{(z-e^{-aT})(z-e^{-bT})} \end{aligned}$$

From this we notice two things:

- The continuous-time poles at $-a$ and $-b$ have been mapped to discrete-time poles at e^{-aT} and e^{-bT}, where T is the sampling interval. Although we have only shown this for a system with two poles, it is true in general.

- Although the continuous-time system has no finite zeros, the discrete-time system has a zero at the origin. Although we know how the poles will map from continuous-time to discrete-time, it is not as obvious how the zeros will map and/or if new zeros will be introduced.

- If the dominant (real) pole in the continuous-time domain is at $-a$ ($|a| < |b|$), the dominant pole in the discrete-time domain will be at e^{-aT} since $e^{-bT} < e^{-aT}$, i.e. the pole at e^{-bT} is closer to the origin than the pole at e^{-aT}.

Next, let's look at the continuous-time transfer function

$$H(s) = \frac{1}{s+a}$$

and it's discrete-time equivalent

$$H(z) = \frac{z}{z - e^{-aT}}$$

In the continuous-time domain, the 2% *settling-time* is usually estimated to be four time constants, so we would estimate $T_s \approx \frac{4}{a}$. Hence to achieve a desired settling time we would need $a = \frac{4}{T_s}$. The pole at $-a$ is mapped to a pole at $e^{-aT} = e^{-4T/T_s}$ in discrete-time. Hence to achieve a settling time of T_s, any discrete-time pole p must have a *magnitude* less than or equal to e^{-4T/T_s}, or

$$|p| \leq e^{-4T/T_s}$$

We can rewrite this expression into a slightly more useful form as

$$T_s \approx \frac{-4T}{\ln(|p|)}$$

This form can be used to find the settling time for discrete-time poles.

Example. Assume we have the discrete-time transfer function $H(z) = \frac{z}{z-0.2}$ and we know the sampling interval is $T = 0.3$ sec. What is the estimated (2%) settling time? Here $|p| = 0.2$, so $T_s \approx \frac{-4(0.3)}{\ln(0.2)} = 0.745$ seconds. To check this, we know the impulse response will be $h(k) = (0.2)^k u(k)$. We have then $h(0T) = 1$, $h(1T) = 0.2$, $h(2T) = 0.04$, and $h(3T) = 0.008$. Clearly the settling time is somewhere between $2T = 0.6$ seconds and $3T = 0.9$ seconds.

Example. Assume we have the discrete-time transfer function $H(z) = \frac{z}{z-0.67}$ and we know the sampling interval is $T = 0.1$ seconds. What is the estimated (2%) settling time? Here $|p| = 0.67$, so $T_s \approx \frac{-4(0.1)}{\ln(0.67)} = 0.998$ seconds. To check this we know the impulse response will be $h(k) = (0.67)^k u(k)$. We have then $h(0T) = 1$, $h(1T) = 0.67$, $h(2T) = 0.449$, $h(3T) = 0.301$, $h(4T) = 0.202$, $h(5T) = 0.135$, $h(6T) = 0.090$, $h(7T) = 0.061$, $h(8T) = 0.041$, $h(9T) = 0.027$,

$h(10T) = 0.0182$. The settling time is approximately $10T = 1.0$.

Example. Assume we have the discrete-time transfer function $H(z) = \frac{z}{(z-0.1)(z+0.2+0.1j)(z+0.2-0.1j)}$ and we know the sampling interval is $T = 0.25$ seconds. What is the estimated (2%) settling time? There are three poles in the system, one pole with magnitude 0.1 and two (complex conjugate) poles with magnitude 0.2236. Just as from continuous-time, the *dominant* pole (or poles) is the pole (or poles) with the slowest response time. In discrete-time, the further away from the origin, the slower the response. Hence in this case the dominant poles are the complex conjugate poles with magnitude $|p| = 0.2236$. Then $T_s \approx \frac{-4(0.25)}{\ln(0.2236)} = 0.667$ seconds.

1.11 Sampling Plants with Zero Order Holds

Most of the system (plants) we will be trying to control in the world are inherently continuous-time, and we would like to use discrete-time control. The biggest issue is how we are going to be able to model the continuous-time plant as a discrete-time plant. The most common method is to assume we have discrete-time signals, $u_d(kT)$, which we model as impulse signals which that only exist at the sample times kT. Specifically, we can write $u_d(kT)$ as

$$u_d(kT) = \ldots + u_d(-T)\delta(t+T) + u_d(0)\delta(t) + u_d(T)\delta(t-T) + u_d(2T)\delta(t-2T) + \ldots$$

where the delta functions are continuous-time delta functions. This is shown graphically in Figure 2.

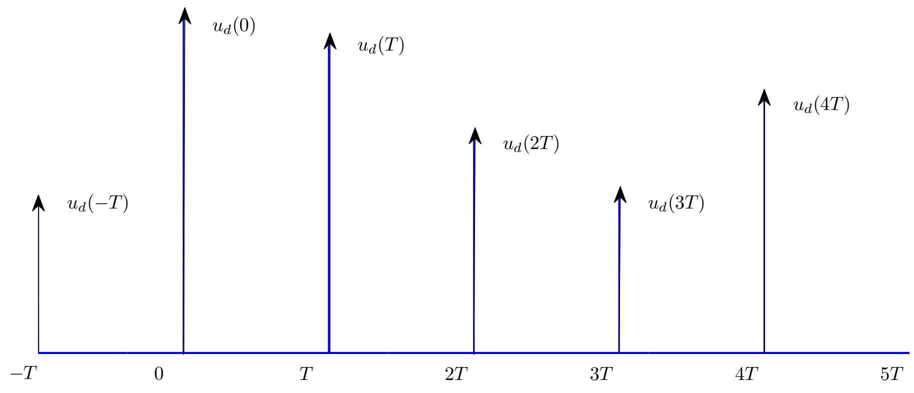

Figure 2: Impulse model of a discrete-time signal.

From these impulse signals we need to generate a continuous-time signal as an input to the plant, $G_p(s)$. The most common way to do this is by the use of a *zero order hold* (zoh), which constructs a continuous-time signal by holding the value of $u(kT)$ constant for time intervals $kT \leq t < (k+1)T$. The impulse response of the zero order hold is $g_{zoh}(t) = u(t) - u(t-T)$. Let's call the continuous-time output of the zero order hold $u_c(t)$, so we have

$$\begin{aligned} u_c(t) &= g_{zoh}(t) * u_d(kT) \\ &= g_{zoh}(t) * [\ldots + u_d(-T)\delta(t+T) + u_d(0)\delta(t) + u_d(T)\delta(t-T) + u_d(2T)\delta(t-2T) + \ldots] \\ &= \ldots g_{zoh}(t) * [u_d(-T)\delta(t+T)] + g_{zoh}(t) * [u_d(0)\delta(t)] + g_{zoh}(t) * [u_d(T)\delta(t-T)] + \ldots \end{aligned}$$

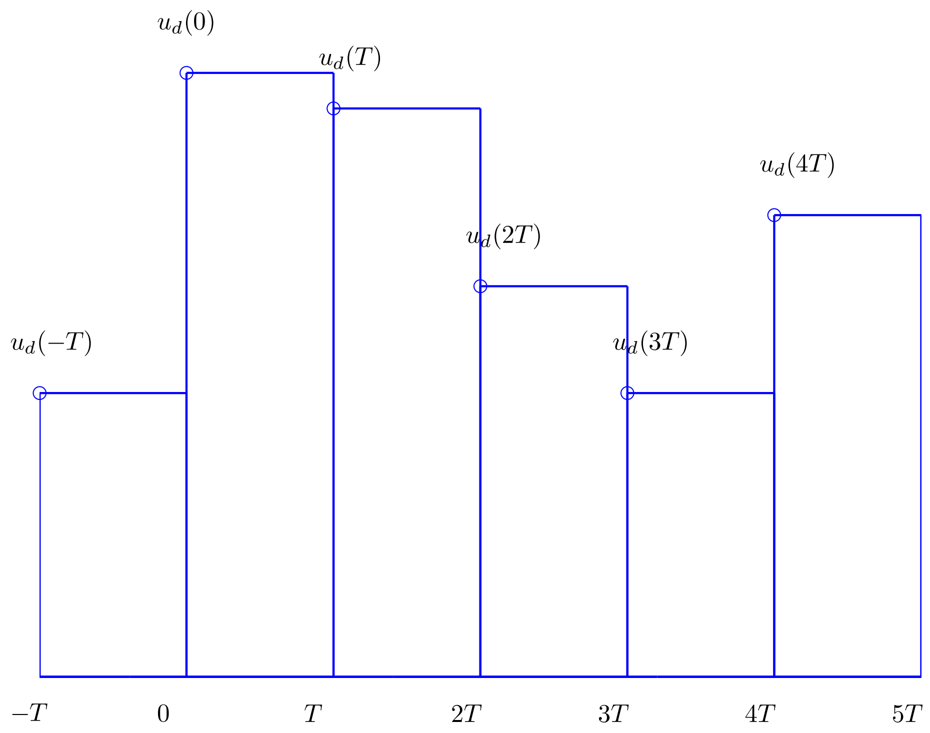

Figure 3: Continuous-time signal $u_c(t)$ as input to the plant. The value of the signal is held constant over the sample interval T.

How let's look at some of these convolutions,

$$\begin{aligned}
g_{zoh}(t) * [u_d(-T)\delta(t+T)] &= u_d(-T)g_{zoh}(t) * \delta(t+T) = u_d(-T)g_{zoh}(t+T) = u_d(-T)[u(t+T) - u(t)] \\
g_{zoh}(t) * [u_d(0)\delta(t)] &= u_d(0)g_{zoh}(t) * \delta(t) = u_d(0)g_{zoh}(t) = u_d(0)[u(t) - u(t-T)] \\
g_{zoh}(t) * [u_d(T)\delta(t-T)] &= u_d(T)g_{zoh}(t) * \delta(t-T) = u_d(0)g_{zoh}(t-T) = u_d(T)[u(t-T) - u(t-2T)]
\end{aligned}$$

Hence we have the input to the continuous-time plant

$$u_c(t) = \ldots + u_d(-T)[u(t+T) - u(t)] + u_d(0)[u(t) - u(t-T)] + u_d(T)[u(t-T) - u(t-2T)] + \ldots$$

This is a staircase function, where we have "filled in" the times between samples with the value of the sample at the beginning of the interval. This is shown graphically in Figure 3. It should be pointed out that a zero order hold is not the only method that can be used to convert a discrete-time signal to a continuous-time signal, but it is the most commonly used method, and is the method we will use when we convert continuous-time state equations to discrete-time state equations.

Now that we know how to convert our discrete-time signal to a continuous-time signal for our continuous-time plant, we need to figure out how to model the zero order hold and the plant as an equivalent discrete-time plant. This will allow us to do all of our design in the discrete-time domain. Figure 4 shows a simple single loop system.

What we want is the discrete-time transfer function that corresponds to the product $G_{zoh}(s)G_p(s)$, or

$$G_p(z) = \mathcal{Z}\left\{G_{zoh}(s)G_p(s)\right\}$$

Using properties of the Laplace transform we have

$$G_{zoh}(s) = \mathcal{L}\left\{g_{zoh}(t)\right\} = \mathcal{L}\left\{u(t) - u(t-T)\right\} = \frac{1}{s} - \frac{e^{-sT}}{s}$$

So we have

$$G_p(z) = \mathcal{Z}\left\{\left[\frac{1}{s} - \frac{e^{-sT}}{s}\right] G_p(s)\right\}$$

We can use this formula directly, but it there is a common trick we can use to make things a bit easier. Let's write

$$G_{zoh}(s)G_p(s) = \left[\frac{1}{s} - \frac{e^{-sT}}{s}\right] G_p(s) = \left[1 - e^{-sT}\right] G_i(s)$$

where we have defined

$$G_i(s) = \frac{G_p(s)}{s}$$

Now we know from Laplace transforms that e^{-sT} corresponds to a time delay of T, hence we have

$$\mathcal{L}^{-1}\left\{e^{-sT}G_i(s)\right\} = g_i(t-T)$$

and then taking z-transforms

$$\mathcal{Z}\left\{g_i(t-T)\right\} = \mathcal{Z}\left\{g_i(kT-T)\right\} = z^{-1}G_i(z)$$

Where

$$G_i(z) = \mathcal{Z}\left\{g_i(kT)\right\}$$

Finally, we will use the Linearity property of the z-transform,

$$G_p(z) = \mathcal{Z}\left\{G_i(s) - e^{-sT}G_i(s)\right\} = G_i(z) - z^{-1}G_i(z) = \left[1 - z^{-1}\right] G_i(z)$$

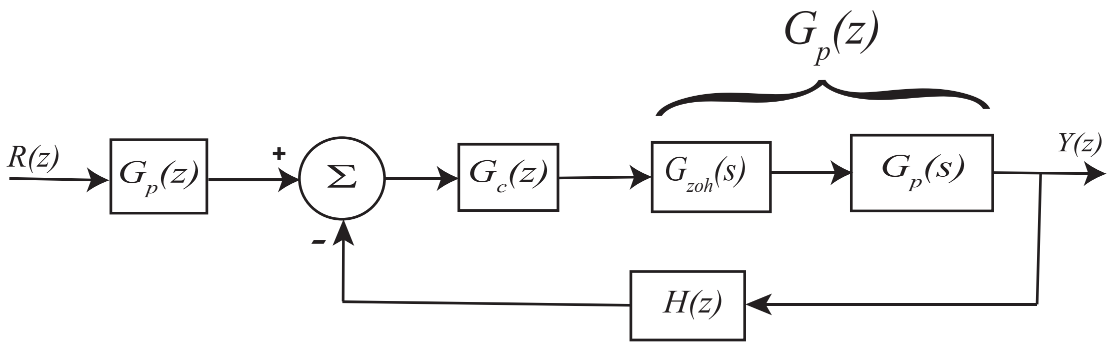

Figure 4: Discrete-time closed loop control system with a zero order hold and a continuous-time plant. We want the equivalent discrete-time plant $G_p(z)$.

which is the answer we want. In summary, the discrete-time transfer function that models the zero order hold and the continuous- time plant is given by

$$G_p(z) = \left[1 - z^{-1}\right] \mathcal{Z}\left\{\frac{G_p(s)}{s}\right\}$$

Example. Consider the continuous-time plant with transfer function $G_p(s) = \frac{1}{s+1}$. What is the equivalent discrete-time transfer function for the plant in series with a zero order hold? We have

$$G_p(z) = \left[1 - z^{-1}\right] \mathcal{Z}\left\{\frac{1}{s(s+1)}\right\}$$

so

$$G_i(s) = \frac{1}{s(s+1)} = \frac{1}{s} - \frac{1}{s+1}$$

and

$$g_i(t) = u(t) - e^{-t}u(t)$$

The sampled version is given as

$$g_i(kT) = u(kT) - e^{-kT}u(kT)$$

with z-transform

$$G_i(z) = \frac{z}{z-1} - \frac{z}{z-e^{-T}} = \frac{z(1-e^{-T})}{(z-1)(z-e^{-T})}$$

Finally, we have

$$G_p(z) = \frac{z-1}{z}G_i(z) = \frac{1-e^{-T}}{z-e^{-T}}$$

Example. Consider the continuous-time plant with transfer function $G_p(s) = \frac{1}{s}$. What is the equivalent discrete-time transfer function for the plant in series with a zero order hold?We have

$$G_p(z) = \left[1 - z^{-1}\right] \mathcal{Z}\left\{\frac{1}{s^s}\right\}$$

so

$$G_i(s) = \frac{1}{s^2}$$

and

$$g_i(t) = tu(t)$$

The sampled version is given as

$$g_i(kT) = kTu(kT)$$

with z-transform

$$G_i(z) = \frac{Tz}{(z-1)^2}$$

Finally, we have

$$G_p(z) = \frac{z-1}{z}G_i(z) = \frac{T}{z-1}$$

1.12 Final Notes

We have focused on using transfer functions in terms of z, such as

$$H(z) = \frac{K(z-a)(z-b)}{(z-c)(z-d)} = \frac{K(z^2-(a+b)z+ab)}{z^2-(c+d)z+cd}$$

This form is useful for determining poles and zeros and determining the time-response of the system. However, when implementing controllers (filters) in discrete-time, it is far more convenient to write things in terms of z^{-1}, which represents a delay of one time sample. For implementation purposes, we would write this transfer function as

$$H(z) = \frac{K(1-(a+b)z^{-1}+abz^{-2})}{1-(c+d)z^{-1}+cdz^{-2}} = \frac{K(1-az^{-1})(1-bz^{-1})}{(1-cz^{-1})(1-dz^{-1})}$$

You should be prepared to see and deal with the two equivalent forms of writing the transfer function.

2 Transfer Function Based Discrete-Time Control

Just as with continuous-time control, we can choose to utilize a transfer function approach or a state variable approach to control. In this chapter we will utilize the transfer function approach, while in the remainder of these notes we will utilize the state variable approach. In this brief chapter we first discuss implementing discrete-time transfer functions, then talk about some common conventions used in control systems, and then discuss common control algorithms.

2.1 Implementing Discrete-Time Transfer Functions

Just as in continuous-time we will represent signals in either the time domain or the transform domain, which in this case is the z-transform domain. A simple input-output transfer function block is shown in Figure 5. While in the continuous time-domain it might be somewhat difficult to solve for the output in terms of the input in the time-domain, in the discrete-time domain we can just use the recursive relationship from the transfer function. Specifically, if we have the transfer function $G(z)$, then we can write it as

$$G(z) \quad = \quad \frac{Y(z)}{U(z)} = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + \cdots + b_m z^{-m}}{1 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_n z^{-n}}$$

where $n \geq m$. Converting this to the time domain, the output $y(k)$ can be written in terms of previous output values and the current and previous input values as

$$y(k) = -a_1 y(k-1) - a_2 y(k-2) + \cdots - a_n y(k-n) + b_0 u(k) + b_1 u(k-1) + ... + b_m u(k-m)$$

This is an Infinite Impulse Response (IIR) filter, since the output at any time depends on both previous outputs and previous inputs. If the output depended only on previous inputs it would be a Finite Impulse Response (FIR) filter.

Figure 5: Discrete-time transfer function, $G(z) = \frac{Y(z)}{U(z)}$. Discrete-time transfer functions are often implemented in terms of the difference equation they represent.

2.2 Not Quite Right

At this point we need to stop and think a little bit about what we are doing, and the common conventions used in control systems. Let's assume we have a simplified version of the above expression, which we will write using the explicit dependence on the sample interval T,

$$y(kT) \quad = \quad -a_1 y((k-1)T) + b_0 u(kT) + b_1 u((k-1)T)$$

Now we have to examine what do we mean by $y(kT)$. Does this mean that this value of y is known at the beginning of the sample interval? This would appear to be correct, but then there is a problem. How can $y(kT)$ be the value of y at the beginning of the sample interval and yet it depends on $u(kT)$, the input at the beginning of the sample interval? Logically, we need to know the value of the input at the beginning of the sample interval $u(kT)$ *and then* we can compute $y(kT)$. There are two ways around this problem. The first, and cleanest, approach is to assume that $b_0 = 0$. This means the output at time kT only depends on the previous outputs and previous inputs. Alas, in case you have not noticed, people in controls really just want to screw with your mind, so this is not the usual assumption. In controls we usually assume that as long as y is known by the end of the interval (before the next sample time), then it is ok to call it $y(kT)$. Thus, for our example, we might assume $u(kT)$, $u((k-1)T)$, and $y((k-1)T)$ are known at time kT. We then do some calculations to compute y, and if these calculations are finished before the next sample interval, we will call this output value $y(kT)$. When we deal with state variable feedback systems, we will see how we can account for this delay if we want to. Next, we want to look at a basic control loop.

A common configuration for a discrete-time control system is shown in Figure 6. In this simple figure there is a discrete-time prefilter, a discrete-time controller, and a discrete-time representation of a plant. In this figure, we are representing all of the signals in the z-domain, as is the usual custom. In Figure 7 we have represented the signals in the discrete-time domain. If you think about this figure for a while, it should begin to be a little bit disturbing. Why? Well, $e(kT) = G_{pf}r(kT) - y(kT)$, $e(k)$ is the input to the controller which produces the control signal $u(kT)$, which goes into the plant to produce the output $y(kT)$. So now we have $y(kT)$ depends on $e(kT)$ which depends on $y(kT)$. Simulink refers to this as an *algebraic loop.* Often in control systems, if the sampling interval is small enough, then $y(kT)$ is very close to $y((k-1)T)$ and $y((k+1)T)$, so this approximation is not too far off, and the notation is much easier. For some parts of this course, however, we will explicitly insert a delay between the output and the error signal, so the error will be $e(k) = G_{pf}r(k) - y(k-1)$, as is shown in Figure 8. Next we need to discuss some common control configurations.

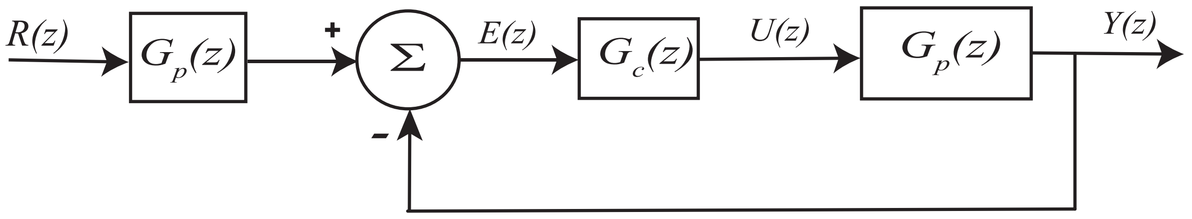

Figure 6: A common discrete-time feedback system written in the z-domain.

2.3 PID and Constant Prefilters

We will first determine how to compute the value of the constant prefilter, though it is generally better to include an integrator in your system than to depend on a prefilter. For the configuration

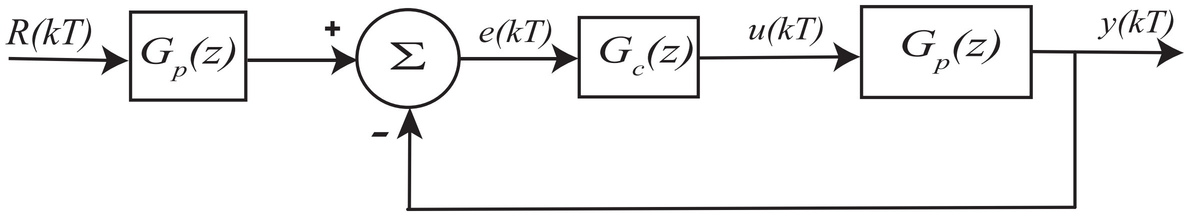

Figure 7: A common discrete-time feedback system written in the time domain.

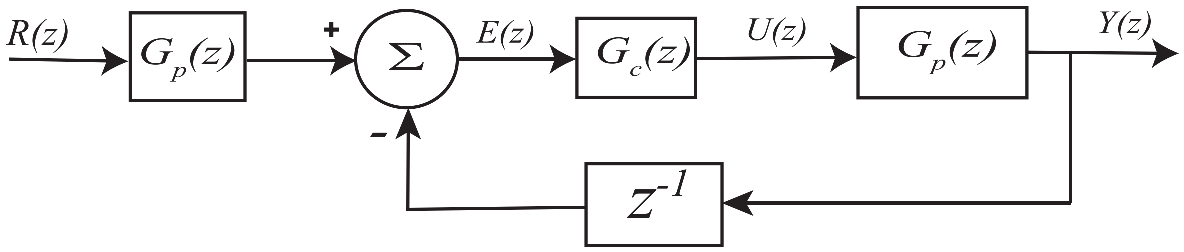

Figure 8: A common discrete-time feedback system written with a delay between the output signal and the error signal.

in Figure 9 we have the closed loop transfer function

$$G_0 = \frac{G_{pf}G_c(z)G_p(z)}{1 + H(z)G_c(z)G_p(z)}$$

Assuming the system is stable, the final value for a unit step input is determined by

$$\begin{aligned}\lim_{t\to\infty} y(t) &= \lim_{z\to 1} \frac{z-1}{z} G_0(z)R(z) \\ &= \lim_{z\to 1} \frac{z-1}{z} G_0(z)\frac{z}{z-1} = G_0(1)\end{aligned}$$

If the input is a unit step, then we want $\lim_{t\to\infty} y(t) = 1$, so we want $G_0(1) = 1$. Solving for the prefilter we have

$$G_{pf} = \frac{1 + G_c(1)G_p(1)H(1)}{G_c(1)G_p(1)}$$

Now will determine the discrete-time form of a PID controller. For a *proportional controller*, the control signal is directly proportional to the error signal, so we have

$$U(z) = k_p E(z)$$

or

$$G_c(z) = \frac{U(z)}{E(z)} = k_p$$

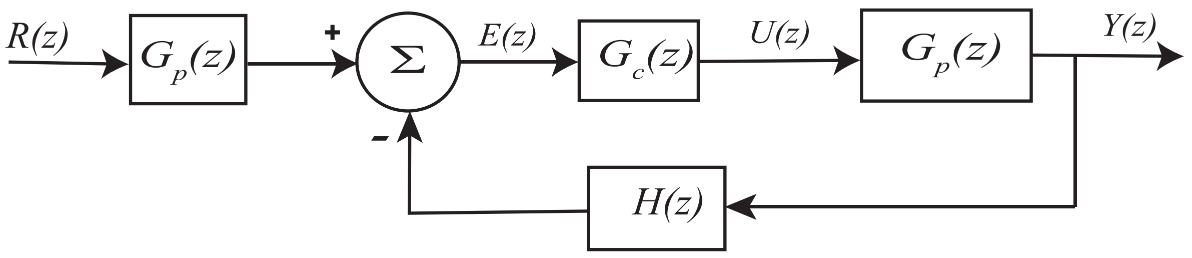

Figure 9: A general discrete-time feedback system written in the z-domain.

which is the same form as we have in continuous time. For an *integral controller* it is probably easiest to start with the continuous-time form,

$$G_c(s) \;=\; k_i \frac{1}{s}$$

In the previous chapter we have the relationship

$$\frac{1}{s} \leftrightarrow u(t) \;=\; u(kT) \leftrightarrow \frac{z}{z-1} = \frac{1}{1-z^{-1}}$$

So a discrete-time integral controller has the form

$$G_c(z) \;=\; k_i \frac{1}{1-z^{-1}}$$

Finally, to get a *derivative controller*, we start with a simple approximation of a derivative, and then take the z-transform,

$$\begin{aligned} u(t) &= k_d \dot{e}(t) \\ &\approx k_d \frac{e(kT) - e((k-1)T)}{T} \end{aligned}$$

or

$$\begin{aligned} U(z) &= \approx k_d \frac{E(z) - z^{-1}E(z)}{T} \\ &= \frac{k_d}{T}(1-z^{-1})E(z) \end{aligned}$$

It is common to just remove the explicit dependence on the sample interval and just write the derivative controller as

$$G_c(z) \;=\; k_d(1-z^{-1})$$

It should be somewhat comforting that our approximation for the derivative is the inverse (reciprocal) of our expression for the integral. Finally, we can write our discrete-time PID controller as

$$G_c(z) \;=\; k_p + k_i \frac{1}{1-z^{-1}} + k_d(1-z^{-1})$$

Often root locus methods are used for designing closed loop control systems. The construction of the root locus plot is identical to that for continuous-time systems, but the interpretation is different. For continuous-time systems, the system is stable if the poles of the closed loop system are all in the open left-half plane, while for discrete-time systems the system is stable is the poles of the closed loop system are inside the open unit circle. For a continuous-time system, the further the poles are from the $j\omega$ axis, the faster the response is, while for discrete-time systems, the closer to the origin the poles are the faster the response is.

In addition to using the root locus plot, or a tool like Matlab's *sisotool* to construct the root locus plot, we can also design controllers algebraically to place the closed loop poles exactly where we want them to be. This is discussed in the next section.

2.4 Placing Closed Loop Poles

In this section we will utilize an algebraic technique to design a controller to place the closed loop poles where we want them to be. This technique is very similar to utilizing Diophantine equations to place closed loop poles, which you may have studied in continuous-time controls class. In both of our examples we will assume a plant of the form

$$G_p(z) = \frac{b_0^p z + b_1^p}{z + a_1^p}$$

Here we have assumed that $a_0^p = 1$. Let's assume we are utilizing a closed loop control scheme like that shown in Figure 8 where $H(z) = z^{-1}$. For now we will assume a controller of the form,

$$G_c(z) = \frac{b_0^c z + b_1^c}{z + a_1^c}$$

The closed loop transfer function is

$$G_0(z) = \frac{G_c(z)G_p(z)}{1 + G_c(z)G_p(z)z^{-1}}$$

So the characteristic equation is

$$\begin{aligned}\Delta(z) &= 1 + G_c(z)G_p(z)z^{-1} \\ &= 1 + \left(\frac{b_0^c z + b_1^c}{z + a_1^c}\right)\left(\frac{b_0^p z + b_1^p}{z + a_1^p}\right) z^{-1}\end{aligned}$$

The closed-loop poles are the locations where $\Delta(z) = 0$. In order to solve for $\Delta(z) = 0$, we can rewrite the last equation as

$$\Delta(z) = z\left[(z + a_1^p)(z + a_1^c)\right] + \left[(b_0^c z + b_1^c)(b_0^p z + b_1^p)\right] = 0$$

Expanding this out we have

$$\Delta(z) = z^3 + \left[a_1^p + a_1^c + b_0^c b_0^p\right] z^2 + \left[a_1^p a_1^c + b_1^c b_0^p + b_1^p b_0^c\right] z + \left[b_1^c b_1^p\right] z^0$$

Now since this is a third order polynomial, we need to find three pole locations we would like our closed loop system to have. Specifically, if we assume we want the closed loop poles to be a μ_1, μ_2, and μ_3, then the desired closed loop poles are determined by

$$\Delta(z) \;=\; (z-\mu_1)(z-\mu_2)(z-\mu_3) = z^3 + d_1 z^2 + d_2 z + d_3$$

Equating coefficients we have

$$\begin{aligned} a_1^p + a_1^c + b_0^p b_0^c &= d_1 \\ a_1^p a_1^c + b_0^p b_1^c + b_1^p b_0^c &= d_2 \\ b_1^p b_1^c &= d_3 \end{aligned}$$

We can write this in matrix form then as

$$\begin{bmatrix} 1 & b_0^p & 0 \\ a_1^p & b_1^p & b_0^p \\ 0 & 0 & b_1^p \end{bmatrix} \begin{bmatrix} a_1^c \\ b_0^c \\ b_1^c \end{bmatrix} = \begin{bmatrix} d_1 - a_1^p \\ d_2 \\ d_3 \end{bmatrix}$$

Example. Assume our plant is given by $G_p(z) = \frac{2z+0.5}{z-0.5}$ and our controller is of the form $G_c(z) = \frac{b_0^c z + b_1^c}{z + a_1^c}$. We can then choose to assign the three closed loop poles wherever we want. Let's assume we want the closed loop poles at 0.1, 0.2, and 0.3, so that we want

$$\begin{aligned} \Delta(z) &= (z-0.1)(z-0.2)(z-0.3) \\ &= z^2 - 0.6z^2 + 0.11z - 0.006 \end{aligned}$$

We then have, $d_1 = -0.6$, $d_2 = 0.11$, $d_3 = -0.006$, $a_1^p = 0.25$, $b_0^p = 2$, and $b_1^p = 0.5$. Solving the above matrix equation we find the controller to be

$$G_c(z) \;=\; \frac{0.0465z - 0.012}{z - 0.443}$$

and the closed loop transfer function to be

$$G_0(z) \;=\; \frac{z(2z+0.5)(0.0465z - 0.012)}{z^3 - 0.6z^2 + 0.11z - 0.006}$$

Note that we have introduced a zero into our system. Often you need to simulate the system to see if any of the zero's you introduced into the system significantly alter the desired closed-loop system behavior. Also, it is important to note that this method does not guarantee that the controller will be stable. It is always important to be sure the controller you implement is stable. If you go though this method and find the controller is not stable, then you will need to change the location of the closed loop poles.

Next, let's assume we want to introduce a controller to make our system a type one system. In this case we need to increase both the order of the numerator and the denominator in the controller, as well as being sure the controller has a pole a 1 (to make a type one system). Hence we assume the following form for the controller,

$$G_c(z) \;=\; \frac{b_0^c z^2 + b_1^c z + b_2^c}{(z-1)(z-\alpha)}$$

The unknowns in our controller are then b_0^c, b_1^c, b_2^c, and α. The characteristic equation for this system is then

$$\begin{aligned}\Delta(z) &= z\left[(z+a_1^p)(z-1)(z-\alpha)\right] + \left[(b_0^p z + b_1^p)(b_0^c z^2 + b_1^c z + b_2^c)\right] \\ &= z^4 + \left[a_1^p - (1+\alpha) + b_0^p b_0^c\right] z^3 + \left[-(1+\alpha)a_1^p + \alpha + b_1^p b_0^c + b_1^c b_0^p\right] z^2 \\ &\quad + \left[a_1^p \alpha + b_1^p b_1^c + b_0^p b_2^c\right] z + \left[b_1^p b_2^c\right] z^0\end{aligned}$$

Our closed loop system will then need four poles, μ_1, μ_2, μ_3, and μ_4, so the desired characteristic equation is

$$\begin{aligned}\Delta(z) &= (z-\mu_1)(z-\mu_2)(z-\mu_3)(z-\mu_4) \\ &= z^4 + d_1 z^3 + d_2 z^2 + d_3 z + d_4\end{aligned}$$

Equating powers of z as before, we end up with the following matrix equations to be solved

$$\begin{bmatrix} -1 & b_0^p & 0 & 0 \\ (1-a_1^p) & b_1^p & b_0^p & 0 \\ a_1^p & 0 & b_1^p & b_0^p \\ 0 & 0 & 0 & b_1^p \end{bmatrix} \begin{bmatrix} \alpha \\ b_0^c \\ b_1^c \\ b_2^c \end{bmatrix} = \begin{bmatrix} d_1 - a_1^p + 1 \\ d_2 + a_1^p \\ d_3 \\ d_4 \end{bmatrix}$$

Example. Consider the system with the plant $G_p(z) = \frac{3z+0.5}{z-0.25}$. Assume we want a type one system so we assume a controller of the form

$$G_c(z) = \frac{b_0^c z^2 + b_1^c z + b_2^c}{(z-1)(z-\alpha)}$$

The closed loop characteristic equation will be a fourth order equation, so we will need to assign four pole locations. Let's assume we choose the pole locations to be 0.1, 0.2, 0.3+0.1j, 0.3-0.1j. Then the desired closed loop characteristic equation will be

$$\Delta(z) = z^4 - 0.9z^3 + 0.3z^2 - 0.042z + 0.002$$

Solving the above matrix equation we then have

$$G_c(z) = \frac{0.1527z^2 - 0.05389z + 0.004}{(z-1)(z-0.1082)}$$

This controller is asymptotically stable (we wanted the marginally stable pole at $z = 1$ to make a type one system). Note that we have also introduced two zeros into the system, which may affect the step response in a manner we do not like. The only way to know for sure is to simulate the system and modify the chosen closed loop pole locations if necessary.

We have presented two examples for placing the poles of a closed loop system. While the use of the root locus method for designing controllers is very useful, as you will see at the end of the next chapter, if we are having to estimate the transfer function for a time varying plant, then we need to be able to design an adaptive controller that allows us to design a controller on the fly, which is something we cannot do with a root locus based design.

3 Least Squares Problems

In this chapter, we will focus on solving least squares problems. These problems often show up when we are trying to estimate parameters for a system. We will initially review some basic vector calculus and then the use of Lagrange multipliers. Then we will go over setting up and solving some static least squares problems. Next, we will discuss the use of singular value decomposition (SVD) for solving least squares problems. Finally, we will end the chapter deriving a recursive least squares algorithm that can be used in adaptive control problems where the transfer function of the plant may be varying over time.

There are a few things you need to keep in mind when using least-squares methods. First of all, least squares methods have some very nice stochastic properties, which are beyond the scope of this course. Secondly, when doing a least squares fit we are generally squaring the error, so that we count negative errors the same as positive errors. This may not always be desirable. Finally, least squares methods are often used because they are mathematically tractable. That is, they lead to equations that can be solved in closed form. This last point cannot be over stressed. Just because someone uses a least squares method to solve a problem it does not mean it was the best way to solve the problem. Often, it was just more convenient and produced reasonable answers.

3.1 Mathematical Preliminaries

The following mathematical relationships will be useful in the remainder of this chapter and this course, and are provided for your reference. In what follows we will assume A, B, and C are matrices.

$$\begin{aligned}
AB &\neq BA \text{ unless } A \text{ and } B \textit{ commute} \\
(ABC)^T &= C^T B^T A^T \\
(ABC)^{-1} &= C^{-1}B^{-1}A^{-1} \text{ provided each of these matrices exist} \\
A &= A^T \text{ if } A \text{ is symmetric} \\
A^{-1}A = AA^{-1} &= I \text{ where } A^{-1} \text{ is the inverse of } A
\end{aligned}$$

It is also important to remember that $\frac{A}{B}$ is meaningless if B is a matrix.

3.2 Vector Calculus

The subject of vector calculus can be very involved. We will limit ourselves to some simple results we need for solving least squares problems. Let's assume we have a vector

$$y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_p \end{bmatrix}$$

and some function of the vector y, which we can write as $L(y) = L(y_1, y_2, \ldots, y_p)$. We will define the first and second derivatives of this vector as follows:

$$\begin{aligned}
\frac{\partial L}{\partial y} &= \begin{bmatrix} \frac{\partial L}{\partial y_1} & \frac{\partial L}{\partial y_2} & \cdots & \frac{\partial L}{\partial y_p} \end{bmatrix} \\
\frac{\partial^2 L}{\partial y^2} &= \begin{bmatrix} \frac{\partial^2 L}{\partial y_1^2} & \frac{\partial^2 L}{\partial y_1 \partial y_2} & \cdots & \frac{\partial^2 L}{\partial y_1 \partial y_p} \\ \vdots & \vdots & & \vdots \\ \frac{\partial^2 L}{\partial y_p \partial y_1} & \frac{\partial^2 L}{\partial y_p \partial y_2} & \cdots & \frac{\partial^2 L}{\partial y_p^2} \end{bmatrix}
\end{aligned}$$

Example. Consider the function

$$L(y) = L(y_1, y_2) = y_1^2 + y_1 y_2 + y_2$$

We compute the first and second derivatives as follows:

$$\begin{aligned}
\frac{\partial L}{\partial y} &= \begin{bmatrix} 2y_1 + y_2 & y_1 + 1 \end{bmatrix} \\
\frac{\partial^2 L}{\partial y^2} &= \begin{bmatrix} 2 & 1 \\ 1 & 0 \end{bmatrix}
\end{aligned}$$

We next need to make some general rules, so we can apply our derivatives in a more general setting. We will derive three simple rules in what follows:

Let's first assume

$$a = \begin{bmatrix} a_1 \\ a_2 \end{bmatrix}, \quad y = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

and that

$$L = a^T y = y^T a = a_1 y_1 + a_2 y_2$$

Then

$$\frac{\partial L}{\partial y} = \begin{bmatrix} a_1 & a_2 \end{bmatrix} = a^T$$

Let's next assume

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}, \quad y = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

and that

$$L = Ay = \begin{bmatrix} ay_1 + by_2 \\ cy_1 + dy_2 \end{bmatrix}$$

Then

$$\frac{\partial L}{\partial y} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} = A$$

Finally, using the previous definitions,

$$L = y^T A y = ay_1^2 + by_1 y_2 + cy_1 y_2 + dy_2^2$$

Then

$$\begin{aligned}
\frac{\partial L}{\partial y} &= \begin{bmatrix} 2ay_1 + (b+c)y_2 & (b+c)Y_1 + 2dy_2 \end{bmatrix} \\
&= \begin{bmatrix} y_1 & y_2 \end{bmatrix} \begin{bmatrix} 2a & b+c \\ b+c & 2d \end{bmatrix} \\
&= y^T \left(A + A^T\right)
\end{aligned}$$

We can then summarize our "rules" as follows:

$$\begin{aligned}
\frac{\partial}{\partial y}\left(a^T y\right) &= a^T \quad (a \text{ is a vector}) \\
\frac{\partial}{\partial y}(Ay) &= A \quad (A \text{ is a matrix}) \\
\frac{\partial}{\partial y}\left(y^T A y\right) &= y^T\left(A + A^T\right) = y^T(2A) \quad (\text{if } A \text{ is symmetric})
\end{aligned}$$

For single variable optimization problems, we can use second derivative information to determine if we have a maximum or a minimum,

$$\begin{aligned}
\frac{\partial^2 L}{\partial y^2} &> 0 \quad \text{minimum} \\
\frac{\partial^2 L}{\partial y^2} &< 0 \quad \text{maximum} \\
\frac{\partial^2 L}{\partial y^2} &= 0 \quad \text{inflection point}
\end{aligned}$$

In order to extend this to vector calculus, we need the following definitions

- A matrix Z is *positive definite* if $y^T Z y > 0$ for all nonzero vectors y, or, equivalently, if all of the eigenvalues of Z are positive.
- A matrix Z is *positive semidefinite* if $y^T Z y \geq 0$ for all nonzero vectors y, or, equivalently, if all of the eigenvalues of Z are positive or zero.
- A matrix Z is *negative definite* if $y^T Z y < 0$ for all nonzero vectors y, or, equivalently, if all of the eigenvalues of Z are negative
- A matrix Z is *negative semidefinite* if $y^T Z Y \leq 0$ for all nonzero vectors y, or, equivalently, if all of the eigenvalues of Z are negative or zero.

We can now extend our second derivative information to vectors,

$$\begin{aligned}
\frac{\partial^2 L}{\partial y^2} &> 0 \quad (\text{positive definite for a minimum}) \\
\frac{\partial^2 L}{\partial y^2} &< 0 \quad (\text{negative definite for a maximum})
\end{aligned}$$

The quantity $\frac{\partial^2 L}{\partial y^2}$ is often called the *Hessian* matrix, and the Hessian is often a function of y. When we take the required derivatives we may get many possible optimal points. We need to evaluate the Hessian at these points to determine if they are local minima or maxima.

3.3 Lagrange Multipliers

Lagrange multipliers are a way of enforcing conditions on a continuous optimization problem. The Lagrange multiplier is often written as λ and is either a vector or a scalar as necessary. The general idea is to take the expression you want to optimize, add the constraint multiplied by the Lagrange multiplier, and then perform the optimization treating the Lagrange multiplier as a variable to be determined. The constraint must be written in the form of an equation equalling zero. This is best illustrated by a few examples.

Example. Suppose we want to find the point in the plane $x+y=5$ nearest the origin. We first need to determine and expression for the distance to the origin, which we can write as $\sqrt{x^2+y^2}$. Since minimizing this function is the same as minimizing x^2+y^2, this is the function will will use. The function we then want to minimize can be written using Lagrange multipliers as

$$H = \underbrace{\left(x^2+y^2\right)}_{objective\ \ function} + \lambda \underbrace{(x+y-5)}_{costraint}$$

We next set

$$\begin{aligned} \frac{\partial H}{\partial x} &= 0 \\ \frac{\partial H}{\partial y} &= 0 \\ \frac{\partial H}{\partial \lambda} &= 0 \end{aligned}$$

and then solve. For this example

$$\begin{aligned} \frac{\partial H}{\partial x} &= 2x+\lambda = 0 \\ \frac{\partial H}{\partial y} &= 2y+\lambda = 0 \\ \frac{\partial H}{\partial \lambda} &= x+y-5 = 0 \quad \text{(you should always get the constraint!)} \end{aligned}$$

From the first two equations we get

$$x = y = -\frac{\lambda}{2}$$

Inserting this into the third equation we get

$$-\frac{\lambda}{2} - \frac{\lambda}{2} - 5 = 0, \quad \text{or } \lambda = -5$$

Hence we have $x = y = \frac{5}{2}$. To determine if this is indeed a minimum, we need to compute the Hessian matrix and evaluate it at these points:

$$\begin{bmatrix} \frac{\partial^2 H}{\partial x^2} & \frac{\partial^2 H}{\partial x \partial y} \\ \frac{\partial^2 H}{\partial y \partial x} & \frac{\partial^2 H}{\partial^2 y} \end{bmatrix} = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$$

The Hessian is positive definite, so we have found the minimum.

Example. Suppose we want to maximize the volume of a rectangular parallelepiped contained in an ellipsoid. The volume of the parallelepiped is $8xyz$ and the equation for an ellipsoid can be written as

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{d^2} = 1$$

Let's write the function we are trying to optimize as

$$H = -8xyz + \lambda \left(\frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2} - 1 \right)$$

where we have included the negative sign since we are trying to maximize the function. We will check at the end of the problem to see if this is the correct choice. Next we compute derivatives,

$$\begin{aligned} \frac{\partial H}{\partial x} &= -8yz + \lambda \frac{2x}{a^2} = 0 \quad \text{or} \quad \lambda = \frac{4a^2 yz}{x} \\ \frac{\partial H}{\partial y} &= -8xz + \lambda \frac{2y}{b^2} = 0 \quad \text{or} \quad \lambda = \frac{4b^2 xz}{y} \\ \frac{\partial H}{\partial z} &= -8xy + \lambda \frac{2z}{c^2} = 0 \quad \text{or} \quad \lambda = \frac{4c^2 xy}{z} \\ \frac{\partial H}{\partial \lambda} &= \frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2} - 1 = 0 \qquad \text{(the original constraint)} \end{aligned}$$

We then have

$$\lambda = \frac{4a^2 yz}{x} = \frac{4b^2 xz}{y} = \frac{4c^2 xy}{z}$$

Form which we can conclude

$$\frac{x^2}{a^2} = \frac{y^2}{b^2} = \frac{z^2}{c^2}$$

For convenience, we will denote these as d, so $d = \frac{x^2}{a^2} = \frac{y^2}{b^2} = \frac{z^2}{c^2}$. We next put these values into the constraint equation to get

$$d + d + d - 1 = 0$$

or $d = \frac{1}{3}$. This means the optimal value is

$$x = \frac{a}{\sqrt{3}}, \; y = \frac{b}{\sqrt{3}}, \; z = \frac{c}{\sqrt{3}}$$

Finally, we know we have found either a minimum or maximum (or infection/saddle) point, but we don't know which. We know that the point $(x, y, z) = (a, 0, 0)$ meets the constraint and produces a volume of zero. Hence we can conclude that we have found a local maximum.

Example. Assume M is a symmetric matrix, and find the vector x to maximize x^TMx subject to the constraint that x is a unit vector. We can write the constraint that x is a unit vector as $x^Tx = 1$. Hence the function we want to maximize is

$$H = -x^TMx + \lambda\left(x^Tx - 1\right)$$

We then compute

$$\frac{\partial H}{\partial x} = -x^T\left(M + M^T\right) + \lambda x^T\left(I + I^T\right) = -2x^TM + 2\lambda x^T = 0$$

and

$$\frac{\partial H}{\partial \lambda} = x^Tx - 1 = 0$$

From the first equation we have

$$x^TM = x^T\lambda$$

If we take the transpose of this expression we get

$$M^Tx = Mx = \lambda x$$

which indicates that λ is an eigenvalue of M. We can then post multiply both sides by x, and then use the constraint to get

$$x^TMx = x^Tx\lambda = \lambda$$

This result indicates the maximum (and minimum) value of x^TMx subject to $x^Tx = 1$ is the the largest (smallest) eigenvalue of the symmetric matrix M.

Example. Assume we have the system $Au = b$ where there are more columns than rows. This is an over determined system of equations, and it comes up alot in control systems. Let's find the solution to this system u that has the smallest norm. We can write this as

$$H = \underbrace{u^Tu}_{minimum\ \ norm} + \lambda^T(Au - b)$$

Note that in this case our Lagrange multiplier is a vector. Since x^Tx is a scalar, we need the second term to be a scalar, and since the constraint is a vector, that means the Lagrange multiplier must be a vector. We then compute

$$\frac{\partial H}{\partial u} = u^T\left(I + I^T\right) + \lambda^TA = 2u^T + \lambda^TA = 0$$

or

$$u = -\frac{1}{2}A^T\lambda$$

We also have

$$\frac{\partial H}{\partial \lambda} = Au - b = 0$$

Substituting in for u we have

$$-\frac{1}{2}AA^T\lambda = b, \quad \text{or } \lambda = -2\left(AA^T\right)^{-1}b$$

Finally,

$$u = -\frac{1}{2}A^T\lambda = -\frac{1}{2}A^T\left[-2\left(AA^T\right)^{-1}b\right] = A^T\left(AA^T\right)^{-1}b$$

which is our solution.

It is also possible to include integral and differential constraints on optimization problems, but we will not consider them here.

3.4 Least Squares Problems

Let's assume we have a model of a process

$$\hat{y} \;=\; H\hat{x}$$

where

$$\begin{aligned}\hat{y} &= \text{the estimated output}\\ \hat{x} &= \text{the estimated input or system parameters}\\ H &= \text{transfer matrix relating the input to the output}\end{aligned}$$

Let's also assume d is a vector of measured outputs of the system.

We want to choose $\hat{x}$ based on the measured data. We will define the error vector as

$$\begin{aligned}e &= d - \hat{y}\\ &= d - H\hat{x}\\ &= \begin{bmatrix} e_1\\ e_2\\ \vdots\\ e_n\end{bmatrix}\end{aligned}$$

We now want to choose $\hat{x}$ to minimize the squared error

$$\begin{aligned}e^Te &= e_1^2 + e_2^2 + \cdots + e_n^2\\ &= (d - H\hat{x})^T(d - H\hat{x})\\ &= (d^T - \hat{x}^TH^T)(d - H\hat{x})\\ &= d^Td - d^TH\hat{x} - \hat{x}^TH^Td + \hat{x}^TH^TH\hat{x}\end{aligned}$$

Before we go on, we need to make a few observations. First of all, all of the terms in the above sum are scalars and the transpose of a scalar is the same as the original scalar. Thus we have $d^TH\hat{x} = \hat{x}^TH^Td$. Thus we can rewrite the squared error as

$$e^Te \;=\; d^Td - 2d^TH\hat{x} + \hat{x}^TH^TH\hat{x}$$

We need to find the value of $\hat{x}$ to minimize this function. Taking the derivatives of each term with respect to $\hat{x}$ we get

$$\frac{\partial e^T e}{\partial \hat{x}} = 0 - 2d^T H + 2\hat{x}^T H^T H = 0$$

Taking the transpose of this expression and rearranging this we get

$$\begin{aligned} H^T d &= H^T H \hat{x} \\ \hat{x} &= \left[H^T H\right]^{-1} H^T d \end{aligned}$$

This is then our solution for the least squares problem. In order to be able to use this solution we need to be able to write our problem in the form

$$\underbrace{\hat{y}}_{model\ outputs} = \underbrace{H}_{known\ matrix} \underbrace{\hat{x}}_{unknowns}$$

Example. Consider a process with output at time t_i, $q(t_i)$. We have measured the following outputs the the specified times:

t_i	$q(t_i)$
1.0	0.50
1.5	0.15
1.6	0.15
2.3	-0.30
4.0	-2.40

We believe the process can be modeled as $q(t) = \alpha_0 + \alpha_1 t + \alpha_2 t^2$, and we want to use a least squares process to determine the unknown parameters α_0, α_1, and α_2. The first thing we need to do is to determine what are unknowns are, and they make up the $\hat{x}$ vector. So we have

$$\hat{x} = \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_3 \end{bmatrix}$$

If we look at the output at first time instant, t_0, we have

$$q(t_0) = \alpha_0 + \alpha_1 t_0 + \alpha_2 t_0^2 = \begin{bmatrix} 1 & t_0 & t_0^2 \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \end{bmatrix}$$

At the next time instant, t_1, we have

$$q(t_1) = \alpha_0 + \alpha_1 t_1 + \alpha_2 t_1^2 = \begin{bmatrix} 1 & t_1 & t_1^2 \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \end{bmatrix}$$

Continuing in this way we have, for all of out time instances,

$$\begin{bmatrix} q(t_0) \\ q(t_1) \\ q(t_2) \\ q(t_3) \\ q(t_4) \end{bmatrix} = \begin{bmatrix} 1 & t_0 & t_0^2 \\ 1 & t_1 & t_1^2 \\ 1 & t_2 & t_2^2 \\ 1 & t_3 & t_3^2 \\ 1 & t_4 & t_4^2 \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \end{bmatrix}$$

With numbers we have

$$d = \begin{bmatrix} 0.50 \\ 0.15 \\ 0.15 \\ -0.30 \\ -2.40 \end{bmatrix} \qquad H = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1.5 & 2.25 \\ 1 & 1.6 & 2.56 \\ 1 & 2.3 & 5.29 \\ 1 & 4.0 & 16.00 \end{bmatrix} \qquad \hat{x} = \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \end{bmatrix}$$

Solving this we get

$$\hat{x} = \left[H^T A\right]^{-1} H^T d = \begin{bmatrix} 0.632 \\ 0.018 \\ -0.193 \end{bmatrix}$$

It is usually a good idea to check some values to see if they are reasonable. At times $t = 1.6$ and $t = 3$ we have

$$\begin{aligned} q(1.6) &= 0.63 + 0.018(1.6) - 0.193(1.6^2) = 0.165 \\ q(3.0) &= 0.63 + 0.018(3.0) - 0.193(3.0^2) = -1.053 \end{aligned}$$

Both of these values seem reasonable given our measured data values, so we have probably not screwed anything up too badly. Note that we should not expect $q(1.6) = 0.15$. Although this was the measured data, we are not required to exactly match this.

Now, what if you have reason to believe that some of your data is more reliable than other data, and you want to 'count' the data you think is more reliable more than the other data. The you might want to use a *weighted least squares.* One way to formulate this is to minimizes the weighted squared error,

$$e^T Re = r_{11}e_1^2 + r_{22}e_2^2 + \cdots + r_{nn}e_n^2$$

where R is a diagonal matrix with values r_{ii} on the diagonal. In this case, the weighted least squares estimate would be

$$\hat{x} = \left[A^T RA\right]^{-1} A^T Rd$$

Example. From our previous example, if $r_{11} = r_{22} = r_{33} = 2$ and $r_{44} = r_{55} = 1$, then this indicates that we believe the first three data points more than the last two data points, and our corresponding estimate is

$$\hat{x} = \begin{bmatrix} 0.702 \\ -0.057 \\ -0.179 \end{bmatrix}$$

3.5 Singular Value Decomposition (SVD)

Let's consider the following matrix equation,

$$y = Hx$$

where x is the input, y is the output, and x and y are related through the system transfer matrix H, where

$$H = \begin{bmatrix} -0.0505 & 0.0683 & 0.1175 & 0.0538 & 0.1709 \\ 0.2215 & 0.1729 & 0.0005 & 0.2794 & 0.2963 \\ 0.4663 & 0.4675 & -0.0623 & 0.4237 & 0.3403 \\ -0.1871 & 0.0701 & 0.2650 & 0.0460 & 0.3136 \\ 0.2109 & 0.0203 & 0.0513 & -0.0603 & 0.0717 \end{bmatrix}$$

Let's assume we measure the output y and want to determine the input x. If we measure

$$y = \begin{bmatrix} 1.508300 \\ 3.167900 \\ 4.610700 \\ 2.500100 \\ 0.522700 \end{bmatrix}$$

then the corresponding input is

$$x = \begin{bmatrix} 1.000000 \\ 2.000000 \\ 3.000000 \\ 4.000000 \\ 5.000000 \end{bmatrix}$$

Now let's assume there is noise on our measured output signal, and instead measure

$$y = \begin{bmatrix} 1.312709 \\ 3.237598 \\ 4.196984 \\ 3.111464 \\ 0.333594 \end{bmatrix}$$

This noisy signal corresponds to a signal to noise ratio of approximately 20 dB. If this is the measured output, then the corresponding input is

$$x = \begin{bmatrix} 5273.048 \\ -5268.62 \\ 21083.12 \\ 15811.04 \\ -15801.36 \end{bmatrix}$$

Clearly this noisy output has dramatically changed our estimate of the input. Just to check our answer, let's assume we make another measurement of the output, again with a signal to noise ratio of approximately 20 dB. For our second measurement we get

$$y = \begin{bmatrix} 1.299894 \\ 3.397150 \\ 4.287858 \\ 2.217089 \\ 1.051207 \end{bmatrix}$$

and find the corresponding value of the input is

$$x = \begin{bmatrix} 629.59 \\ -627.80 \\ 2506.33 \\ 1881.00 \\ -1870.79 \end{bmatrix}$$

Now what we have are two noisy outputs with a 20 dB signal to noise ratio that yield estimates of the input signal that are quite different, and also quite different from the correct input if we could produce a *noise free* estimate of the output. While it might be nice to pretend all of our measurements were noise free, this is not likely to be true, so we need a different approach to this problem which will also allow us to gain some insight into what is going on in our system.

It turns out that for *any* matrix $H \in R^{m\times n}$ we can always compute its *singular value decomposition*

$$H = USV^T$$

where, $U \in R^{m\times m}$ and U is a *unitary* matrix ($U^T = U^{-1}$), $V \in R^{n\times n}$ and V is a unitary matrix ($V^T = V^{-1}$), and $S \in R^{m\times n}$. S is all zeros except for a submatrix, which will will denote Σ, of size $m \times m$ or $n \times n$ (depending on whether m or n is smaller) in the upper left corner. The entries in the submatrix Σ are called the *singular values* of the matrix and will be denoted σ_i for the i^{th} singular value. In general, $\sigma_i \geq \sigma_{i+1}$, that is, the singular values decrease as the mode number increases (this is important!) As you will see, these singular values contain a great deal of information about the matrix (and what may be going wrong).

Let's assume that

$$\begin{aligned} x &= V\alpha \\ y &= U\beta \end{aligned}$$

That is, x can be written as a linear combination of the columns of V

$$\begin{aligned} x &= V\alpha \\ &= [v_1\ v_2\ ...\ v_n]\ \alpha \\ &= v_1\alpha_1 + v_2\alpha_2 + ...\ + v_n\alpha_n \end{aligned}$$

Since $x \in R^n$ and the columns of V span R^n, we know such an expansion must exist. Similarly, we must be able to represent y as a linear combination of the columns of U.

Now assume we have our problem $y = Hx$. If we write $y = U\beta$, $x = V\alpha$, and $H = USV^T$, then we get

$$\begin{aligned} (U\beta) &= (USV^T)(V\alpha) \\ &= USV^TV\alpha \\ &= US\alpha \end{aligned}$$

Then since U has full rank we have $\beta = S\alpha$, or more precisely $\alpha = \Sigma^{-1}\beta$. (Note, if there are more α_i than β_i, the extra α_i are set to zero. This just means that $\mathbf{V}$ spans a larger space than $\mathbf{U}$. Similarly if there are more β_i than α_i, the extra β_i are set to zero. This just means $\mathbf{U}$ spans a larger space than $\mathbf{V}$.) This can be written in terms of components as

$$\alpha_i = \beta_i/\sigma_i$$

At this point we are almost done. To get β we just use the relationship

$$\begin{aligned} y &= U\beta \\ U^T y &= U^T U\beta = \beta \end{aligned}$$

we we just have $\beta = U^T y$.

Although it may seem like we just played with a bunch of math, we are now in a position to analyze what is going wrong with our matrix equation. If there was no noise in our measurement for y, then we would have

$$\begin{aligned} y &= \beta_1 u_1 + \beta_2 u_2 + \beta_3 u_3 + \beta_4 u_4 \\ x &= \frac{\beta_1}{\sigma_1} v_1 + \frac{\beta_2}{\sigma_2} v_2 + \frac{\beta_3}{\sigma_3} v_3 + \frac{\beta_4}{\sigma_4} v_4 \end{aligned}$$

However, since there is measurement error in our output, we will model the output as the 'true' output y plus an error term, δy, thus we have the measured output $y + \delta y$. Going through our analysis above we have instead of the above,

$$\begin{aligned} y + \delta y &= (\beta_1 + \delta\beta_1)u_1 + (\beta_2 + \delta\beta_2)u_2 + (\beta_3 + \delta\beta_3)u_3 + (\beta_4 + \delta\beta_4)u_4 \\ x + \delta x &= \frac{\beta_1 + \delta\beta_1}{\sigma_1} v_1 + \frac{\beta_2 + \delta\beta_2}{\sigma_2} v_2 + \frac{\beta_3 + \delta\beta_3}{\sigma_3} v_3 + \frac{\beta_4 + \delta\beta_4}{\sigma_4} v_4 \end{aligned}$$

or, the deviation from the true answer x is

$$\delta x = \frac{\delta\beta_1}{\sigma_1} v_1 + \frac{\delta\beta_2}{\sigma_2} v_2 + \frac{\delta\beta_3}{\sigma_3} v_3 + \frac{\delta\beta_4}{\sigma_4} v_4$$

Now, since by construction, the magnitude of the v_i are one, and the errors in the observation are assumed to be reasonably small, we need to look at the sizes of the singular values σ_i. As they get smaller, they will **amplify** the errors $\delta\beta_i$ more strongly.

If we go back to our original example, the singular values for the H matrix were 1.000017, 0.500001, 0.200018, 0.100003, and 0.000013. If we then compute the absolute value of $\delta\beta_i/\sigma_i$ for the first noisy vector we get

$$\begin{aligned} |\frac{\delta\beta_1}{\sigma_1}| &= 0.29 \\ |\frac{\delta\beta_2}{\sigma_2}| &= 1.16 \\ |\frac{\delta\beta_3}{\sigma_3}| &= 1.79 \\ |\frac{\delta\beta_4}{\sigma_4}| &= 31617.24 \end{aligned}$$

Clearly, the last (smallest) singular value is causing most of the problem. If we were to only use the first three vectors in the expansion for x, $x \approx \frac{\beta_1}{\sigma_1}v_1 + \frac{\beta_2}{\sigma_2}v_2 + \frac{\beta_3}{\sigma_3}v_3$, then our estimates for x for the two examples above would be

$$x = \begin{bmatrix} -0.121 \\ 0.703 \\ 2.162 \\ 4.115 \\ 6.723 \end{bmatrix} \qquad x = \begin{bmatrix} 3.130 \\ -1.793 \\ 1.868 \\ 3.107 \\ 7.239 \end{bmatrix}$$

While these are not the 'correct' answers, they are still orders of magnitude closer than if we had done nothing. Knowing how many terms to keep when using the singular value decomposition to solve a problem is a bit tricky. Usually you remove terms with singular values below a certain threshold. This is what the Matlab command *pinv* does.

Finally, let's assume we have an overdetermined least squares solution

$$\hat{x} = \left[H^T H\right]^{-1} H^T y$$

With the assumptions above, and writing H in terms of its singular value decomposition, we have

$$\begin{aligned}
\hat{x} = V\alpha &= \left[(USV^T)^T(USV^T)\right]^{-1}(USV^T)^T(U\beta) \\
&= \left[VS^TU^TUSV^T\right]^{-1}(VS^TU^T)(U\beta) \\
&= \left[VS^TSV^T\right]^{-1}(VS^T\beta) \\
&= \left[(V^T)^{-1}(S^TS)^{-1}V^{-1}\right](VS^T\beta) \\
&= V(S^TS)^{-1}S^T\beta
\end{aligned}$$

Where for simplicity we have assumed the S is invertible. Since V has full rank we have

$$\alpha = V(S^TS)^{-1}S^T\beta$$

Since S is a diagonal matrix with elements σ_i, we finally have

$$\alpha_i = \frac{\beta_i}{\sigma_i}$$

just as we did without the least squares.

One method that is often used to get a general idea of how errors in our measurements (the y values) may affect our estimates of the input (the x values) is the *condition number* of the transfer matrix (H). The condition number is generally defined as the ratio of the largest to smallest singular values,

$$\text{condition number} = \frac{\sigma_1}{\sigma_n}$$

We then have the general approximation

$$\text{error in } x \quad \approx \quad \text{error in } d \times \text{condtion number}$$

Keep in mind this is a very rough approximation. For our first example in this section, our condition number is $\frac{1.000017}{0.000013} \approx 77,000$. The largest error in our measured data is approximately 0.5. Hence our estimate of the expect error in the answer x is $(0.5) \times (77,000) \approx 39,000$. In this case our actual error was approximately 16,000, so the condition number estimate was fairly conservative.

3.6 Recursive Least Squares with a Forgetting Factor

Consider a single input-single output system. In general, you can modify the system behavior somewhat by employing an appropriate controller (lag, lead, lead-lag, PID, etc) once the correct transfer function is known. However, sometimes the transfer function may change over time and you will need to be able to adapt your controller (or the input to the plant) based on your current estimate of the system transfer function. As a specific example, consider a drilling process. When the drill is not touching the workpiece it might have one transfer function, when it first starts drilling it may have a different transfer function, and as the drill bit heats up, it may have a third transfer function. In order to try to "optimize" the drilling process, we need to determine estimates of the transfer function. We will assume the plant is represented by the transfer function

$$G_p(z) \quad = \quad \frac{Y(z)}{U(z)} = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + \cdots + b_m z^{-m}}{1 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_n z^{-n}}$$

where $n \geq m$. Converting this to the time domain, we assume the plant output $y(k)$ (which might be the speed of the drill bit) is

$$y(k) = -a_1 y(k-1) - a_2 y(k-2) + \cdots - a_n y(k-n) + b_0 u(k) + b_1 u(k-1) + ... + b_m u(k-m)$$

where $y(k-i)$ are the outputs at previous time steps, $u(k-i)$ are the inputs, and the coefficients $a_1, ...a_m, b_0, ...b_n$ are to be determined. However, these coefficients are changing over time, so we need to be able to estimate them based on current observations.

At this point, we could collect data over a time period and set up a least squares problem as we did before, but there are two general problems with that type of approach. The first is that in order to collect the data we may introduce some significant delays in the system. Secondly, we may end up with a fairly large matrix equation to solve which could also consume significant time. We would like to be able to implement this into a microcontroller with limited memory and computational ability. Hence we want to look at a method to perform the least squares calculation were we update our least squares estimate as each new data point is available. This will lead to a recursive least squares algorithm. However, we would also like to weigh our data so the most recent data is more important than past data. This is particularly important for a process that is changing over time. This leads to a recursive least squares with a *forgetting factor*, or a scaling that discounts past data. Our scaling factor λ will be a number between 0 and 1. If the scaling factor is a 1, then we do not forget any of the past data and our algorithm is a recursive implementation of a regular least squares algorithm. If the scaling factor is near zero, than we only use the most recent data. The recursive least squares is a precursor to the Kalman filter, which estimates the state of a system based on a model of the system and stochastic information, but that is beyond the scope of this course.

Assume, in general, the output at time k, $y(k)$, is given as

$$\begin{aligned} y(k) &= \phi_1(k)\theta_1 + \phi_2(k)\theta_2 + ... + \phi_n(k)\theta_n \\ &= \phi^T(k)\theta \end{aligned}$$

where

$$\begin{aligned} \phi^T(k) &= [\phi_1(k)\ \phi_2(k)\ ...\ \phi_n(k)] \\ \theta^T &= [\theta_1\ \theta_2\ ...\ \theta_n] \end{aligned}$$

For our difference equation describing the plant one way to assign variables is

$$\begin{aligned} \phi^T(k) &= [-y(k-1)\ \ -y(k-2)\ \cdots\ -y(k-n)\ u(k)\ u(k-1)\ \cdots u(k-m)\ ] \\ \theta^T &= [a_1\ a_2\ \cdots\ a_n\ b_0\ b_1\ \cdots\ b_m] \end{aligned}$$

It really doesn't matter whether we assume the coefficients of the y's are negative or the a's are negative.

Let's define the error at time step k as

$$e(k) = y(k) - \hat{y}(k) = y(k) - \phi^T(k)\hat{\theta}$$

where $\hat{\theta}$ is the current estimate of the parameters in θ and $\hat{y}(k) = \phi^T(k)\hat{\theta}$ is the estimate of the true output $y(k)$ at time k. We now want to determine $\hat{\theta}$ to minimize a simple function of the error that incorporates both current and past information. Lets look at the error measure

$$V(\hat{\theta}, k) = \frac{1}{2}\sum_{i=1}^{i=k} \lambda^{k-i} e^2(i)$$

Here λ is a forgetting or deweighting factor between 0 and 1. With this parameter, we can emphasize new data and "discount" or "deweight" old data. For $\lambda = 1$ we have the straight least squares algorithm.

To find the parameters $(\hat{\theta})$ that minimizes V, we need to find the derivative of

$$\begin{aligned} V &= \frac{1}{2}\sum_{i=1}^{i=k} \lambda^{k-i} e^2(i) = \frac{1}{2}\sum_{i=1}^{i=k} \lambda^{k-i} \left[y(i) - \phi^T(i)\hat{\theta}\right]\left[y(i) - \phi^T(i)\hat{\theta}\right] \\ &= \frac{1}{2}\sum_{i=1}^{i=k} \lambda^{k-i} \left[y^2(i) - 2\phi^T(i)\hat{\theta}y(i) + \phi^T(i)\hat{\theta}\phi^T(i)\hat{\theta}\right] \end{aligned}$$

Note that since all our terms are scalars, $\phi^T(i)\hat{\theta} = \hat{\theta}^T\phi(i)$. We want to make all of our terms look like something we know how to compute the derivative of, such as

$$\begin{aligned} \frac{\partial}{\partial\theta}\left(a^T\theta\right) &= a^T \\ \frac{\partial}{\partial\theta}\left(\theta^T A\theta\right) &= \theta^T(2A), \text{ if } A \text{ is symmetric} \end{aligned}$$

Rewriting V we have

$$V = \frac{1}{2}\sum_{i=1}^{i=k} \lambda^{k-i}\left[y^2(i) - 2y(i)\phi^T(i)\hat{\theta} + \hat{\theta}^T\phi(i)\phi^T(i)\hat{\theta}\right]$$

Taking the derivative and setting it equal to zero we have

$$\frac{\partial V}{\partial \hat{\theta}} = \frac{1}{2}\sum_{i=1}^{i=k} \lambda^{k-i}\left[0 - 2y(i)\phi^T(i) + 2\hat{\theta}^T\phi(i)\phi^T(i)\right] = 0$$

taking transposes

$$\sum_{i=1}^{i=k} \lambda^{k-i}\phi(i)y(i) = \sum_{i=1}^{i=k} \lambda^{k-i}\phi(i)\phi^T(i)\hat{\theta}$$

since $\hat{\theta}$ is not a function of the index variable (i) we can take it outside of the sum to get our optimal estimate of the parameters

$$\hat{\theta} = \left[\sum_{i=1}^{i=k} \lambda^{k-i}\phi(i)\phi^T(i)\right]^{-1} \sum_{i=1}^{i=k} \lambda^{k-i}y(i)\phi(i)$$

Since this is our estimate of the transfer function parameters for sample k, we will denote this $\hat{\theta}(k)$.

At this point it probably does not seem like we have made any progress, since our estimate for the parameters depends on keeping all of the data up to the current time. In order to get a recursive algorithm, we need to apply a few tricks. It won't look like we are making progress, but we really are, just be patient! Let's define

$$P(k) = \left\{\sum_{i=1}^{i=k} \lambda^{k-i}\phi(i)\phi^T(i)\right\}^{-1}$$

Then we have

$$P^{-1}(k) = \sum_{i=1}^{i=k} \lambda^{k-i}\phi(i)\phi^T(i)$$

Changing the index and then multiplying by λ we get

$$P^{-1}(k-1) = \sum_{i=1}^{i=k-1} \lambda^{k-1-i}\phi(i)\phi^T(i)$$

$$\lambda P^{-1}(k-1) = \sum_{i=1}^{i=k-1} \lambda^{k-i}\phi(i)\phi^T(i)$$

Finally, we have

$$\lambda P^{-1}(k-1) + \phi(k)\phi^T(k) = \sum_{i=1}^{i=k} \lambda^{k-i}\phi(i)\phi^T(i) = P^{-1}(k)$$

or

$$P^{-1}(k) = \lambda P^{-1}(k-1) + \phi(k)\phi^T(k)$$

At this point we have

$$\begin{aligned}
\hat{\theta}(k) &= P(k)\left[\sum_{i=1}^{i=k} \lambda^{k-i}\phi(i)y(i)\right] \\
&= P(k)\left[\sum_{i=1}^{i=k-1} \lambda^{k-i}\phi(i)y(i) + \phi(k)y(k)\right] \\
&= P(k)\left[\lambda \sum_{i=1}^{i=k-1} \lambda^{k-1-i}\phi(i)y(i) + \phi(k)y(k)\right]
\end{aligned}$$

By changing the index in the first of the previous three expressions we also have

$$\hat{\theta}(k-1) = P(k-1)\left[\sum_{i=1}^{i=k-1} \lambda^{k-1-i}\phi(i)y(i)\right]$$

Rearranging this expression we get

$$P^{-1}(k-1)\hat{\theta}(k-1) = \sum_{i=1}^{i=k-1} \lambda^{k-1-i}\phi(i)y(i)$$

Combining these expression we get

$$\begin{aligned}
\hat{\theta}(k) &= P(k)\left[\lambda P^{-1}(k-1)\hat{\theta}(k-1) + \phi(k)y(k)\right] \\
&= P(k)\left[\left(P^{-1}(k) - \phi(k)\phi^T(k)\right)\hat{\theta}(k-1) + \phi(k)y(k)\right] \\
&= \hat{\theta}(k-1) - P(k)\phi(k)\phi^T(k)\hat{\theta}(k-1) + P(k)\phi(k)y(k)
\end{aligned}$$

Rearranging this we get

$$\hat{\theta}(k) = \hat{\theta}(k-1) + P(k)\phi(k)\left[y(k) - \phi^T(k)\hat{\theta}(k-1)\right]$$

Let's define $K(k) = P(k)\phi(k)$. Then we will have

$$\hat{\theta}(k) = \hat{\theta}(k-1) + K(k)\left[y(k) - \phi(k)\hat{\theta}(k-1)\right]$$

At this point, we are getting closer to a recursive relationship. Our new estimate of the parameters ($\hat{\theta}(k)$) is the sum of our previous estimate ($\hat{\theta}(k-1)$) and a gain ($K(k)$) multiplying the difference between the measured output ($y(k)$) and our estimate of the current output based on the previous parameter estimate ($\phi^T(k)\hat{\theta}(k-1)$). This is very similar to the expression we will use for constructing an *observer*, which is really what we are doing now. All that remains is to figure out a recursive relationship for $K(k)$, or more precisely, $P(k)$.

From before, we have the relationship

$$P^{-1}(k) = \lambda P^{-1}(k-1) + \phi(k)\phi^T(k)$$

which we can rewrite as

$$P(k) \quad = \quad \left[\lambda P^{-1}(k-1) + \phi(k)\phi^T(k)\right]^{-1}$$

Next we use the *matrix inversion lemma*, which states the equality

$$(A + BCD)^{-1} \quad = \quad A^{-1} - A^{-1}B\left(C^{-1} + DA^{-1}B\right)^{-1} DA^{-1}$$

provided A, C, and $C^{-1} + DA^{-1}B$ are nonsingular square matrices. Let's let $A = \lambda P^{-1}(k-1)$ so $A^{-1} = \frac{1}{\lambda}P(k-1)$, $B = \phi(k)$, $C = I$ (the identity matrix), and $D = \phi^T(k)$. Then we have

$$\begin{aligned}
P(k) &= \left[\lambda P^{-1}(k-1) + \phi(k)\phi^T(k)\right]^{-1} \\
&= \left[\frac{1}{\lambda}P(k-1)\right] - \left[\frac{1}{\lambda}P(k-1)\right][\phi(k)]\left([I] + \left[\phi^T(k)\right]\left[\frac{1}{\lambda}P(k-1)\right][\phi(k)]\right)^{-1}\left[\phi^T(k)\right]\left[\frac{1}{\lambda}P(k-1)\right] \\
&= \frac{1}{\lambda}\left[P(k-1) - P(k-1)\phi(k)\left(\lambda + \phi^T(k)P(k-1)\phi(k)\right)^{-1}\phi^T(k)P(k-1)\right]
\end{aligned}$$

Using the previous definition that $K(k) = P(k)\phi(k)$, and the fact that $\phi^T(k)P(k-1)\phi(k)$ is a *scalar*, we have

$$\begin{aligned}
K(k) &= \frac{1}{\lambda}\left\{P(k-1) - \frac{P(k-1)\phi(k)\phi^T(k)P(k-1)}{\lambda + \phi^T(k)P(k-1)\phi(k)}\right\}\phi(k) \\
&= \frac{1}{\lambda}P(k-1)\phi(k)\left\{I - \frac{\phi^T(k)P(k-1)\phi(k)}{\lambda + \phi^T(k)P(k-1)\phi(k)}\right\} \\
&= \frac{1}{\lambda}\left\{\frac{\lambda + \phi^T(k)P(k-1)\phi(k) - \phi^T(k)P(k-1)\phi(k)}{\lambda + \phi^T(k)P(k-1)\phi(k)}\right\} \\
&= P(k-1)\phi(k)\left(\lambda + \phi^T(k)P(k-1)\phi(k)\right)^{-1}
\end{aligned}$$

Finally, using this relationship and the previous relationship

$$P(k) \quad = \quad \frac{1}{\lambda}\left[P(k-1) - P(k-1)\phi(k)\left(\lambda + \phi^T(k)P(k-1)\phi(k)\right)^{-1}\phi^T(k)P(k-1)\right]$$

we get

$$\begin{aligned}
P(k) &= \frac{1}{\lambda}\left[P(k-1) - K(k)\phi^T(k)P(k-1)\right] \\
&= \frac{1}{\lambda}\left[I - K(k)\phi^T(k)\right]P(k-1)
\end{aligned}$$

and this is the relationship we need.

In summary (and in computational order), we then have the following recursive relationships:

$$\begin{aligned}
K(k) &= P(k-1)\phi(k)\left(\lambda + \phi^T(k)P(k-1)\phi(k)\right)^{-1} \\
P(k) &= \frac{1}{\lambda}\left[I - K(k)\phi^T(k)\right]P(k-1) \\
\hat{\theta}(k) &= \hat{\theta}(k-1) + K(k)\left[y(k) - \phi^T(k)\hat{\theta}(k-1)\right]
\end{aligned}$$

Although the variables are indexed by the variable k, this is only for notational purposes. The computations for these equations will only depend on the current output of the system (y) and the current input-output parameters (ϕ), and will look something like the following:

$$\begin{aligned}
K &= P_{old}\, \phi \left(\lambda + \phi^T P_{old}\, \phi\right)^{-1} \\
P_{new} &= \frac{1}{\lambda}\left[I - K\phi^T\right] P_{old} \\
\hat{\theta}_{new} &= \hat{\theta}_{old} + K\left[y - \phi^T \hat{\theta}_{old}\right] \\
P_{old} &= P_{new} \\
\hat{\theta}_{old} &= \hat{\theta}_{new}
\end{aligned}$$

However, in order to start the recursion we need an initial value for ϕ and P. We usually assume $\phi = 0$ and $P = I$ (the identity matrix), though other starting conditions are obviously possible.

Before going though some examples, there are a few points that we need to make:

- To really do a recursive least squares like we are doing, we need a *persistently exciting* input of a sufficient order. That is, our input must have sufficient spectral content to allow us to determine the system. The easiest way to think about this is to try and imagine trying to experimentally determine the Bode plot for an unknown system. In order to determine the Bode plot we need to make measurements at a sufficient number of frequencies that we can accurately determine the parameters.

- The *covariance matrix* P is likely to become singular if our system does not change very much. There are a number of ways to take care of this, the simplest is to test to see if the matrix is becoming singular and resetting the P matrix to something like the identity matrix.

- The algorithm as presented is not very computationally efficient or robust. There are many other (and better) adaptive methods for estimating the parameters.

Now let's looks at an example. This example is fairly extreme, with a sudden change in the plant. For this example we assume the input is white noise. This is an unrealistic assumption, but will help us understand how the algorithms work a little better without having to worry about having a persistently exciting input signal.

Example. Consider the following two discrete-time transfer function,

$$\begin{aligned}
G_1(z) &= \frac{2z+1}{z^2 - z + 0.5} \\
G_2(z) &= \frac{3z-1}{z^2 - 0.5z - 0.25}
\end{aligned}$$

These are both of the general form

$$G(z) = \frac{Y(z)}{U(z)} = \frac{b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}}$$

or

$$y(k) \quad = \quad a_1 y(k-1) + a_2 y(k-2) + b_1 u(k-1) + b_2 u(k-2)$$

We will the assume our plant has the form $G(z) = G_1(z)$ for samples $k < 20$ and $G(z) = G_2(z)$ for $k \geq 20$. This means that we should find the following:

coefficient	$k < 20$	$k \geq 20$
a_1	-1.0	-0.5
a_2	0.5	-0.25
b_1	2	3
b_2	1	-1

Figures 10 - 12 show the results of using the recursive least squares algorithm to estimate the system parameters for $\lambda = 0.4$, $\lambda = 0.75$, and $\lambda = 0.95$. Note that for $\lambda = 0.4$ the coefficients are correctly determined once the transients have settled out, but there is not a very smooth transition from one set of parameters to the second set. For $\lambda = 0.75$ the transition between parameter values is smoother, but it takes longer for the system to settle to the new parameter values. For $\lambda = 0.95$ we are essentially using a recursive least squares with very little forgetting. When the system parameters change, it is very difficult for the system to converge to the correct parameter values since it is still remembering the data for the initial parameter values.

3.7 Adaptive Control

Now that we know how to estimate the plant transfer function using recursive least squares, and we know how to determine a controller to place the closed loop poles from the last chapter, we can combine these two ideas into an elementary example of an adaptive control scheme. Figure 13 shows a simple schematic for an adaptive control system, which is made up of three parts. First of all there is the plant, the system we are trying to control. However, the plant may be either unknown or varying over time. Secondly there is the system identification part, the takes the input to the plant and the output of the plant and constructs and estimate of the plant. Finally, there is the adaptive controller, which takes the estimate of the plant and error signal and produces a control signal.

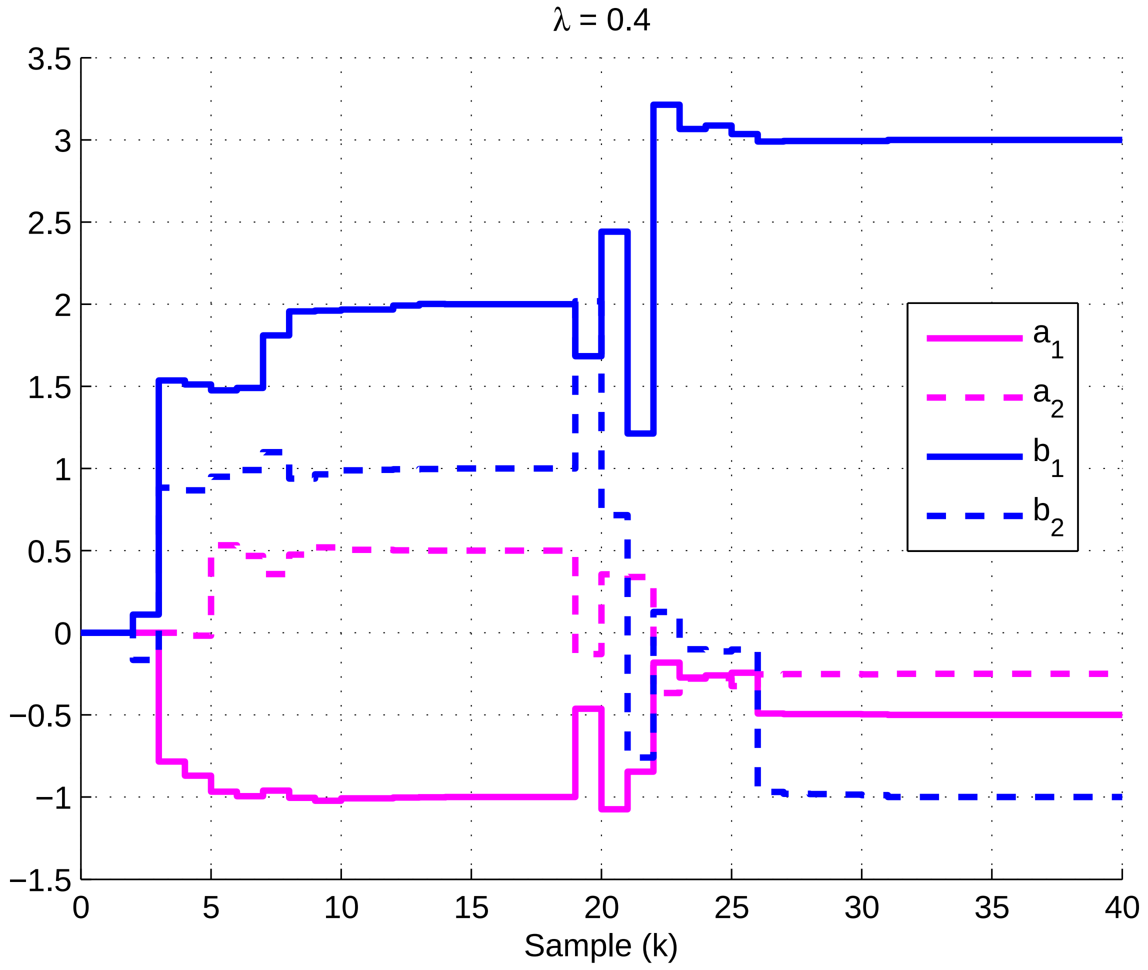

Figure 10: Results of using the recursive least squares algorithm to estimate the system parameters. $\lambda = 0.4$ in this case. Note that the parameter estimates settle at the correct values in the two regions, but there is some transient time when the parameter estimates are significantly in error. However, this is an extreme case of a transition.

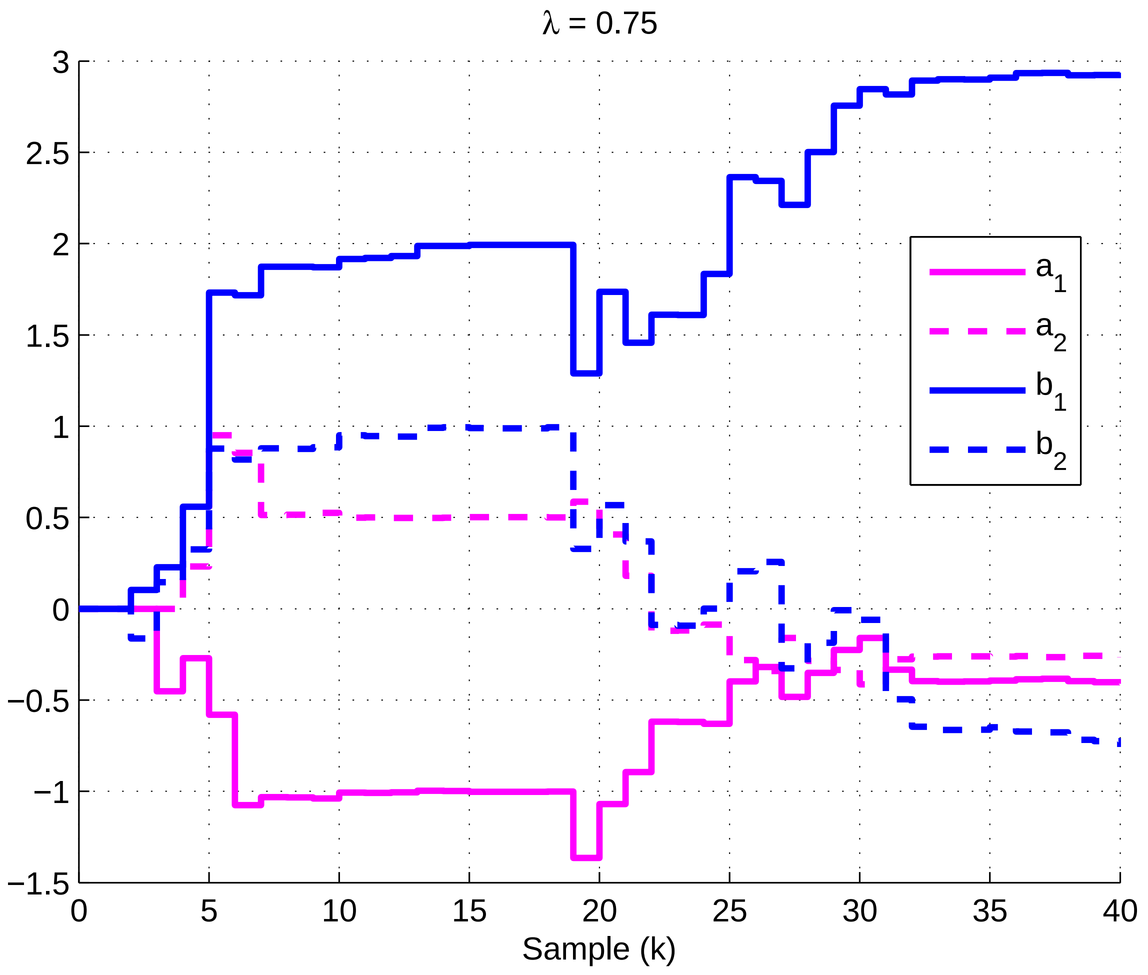

Figure 11: Results of using the recursive least squares algorithm to estimate the system parameters. $\lambda = 0.75$ in this case. Here the parameter estimates again settle at the correct values in the two regions, but the transient behavior is not as extreme as for the case when $\lambda = 0.4$. This smoother transition between regions is as the expense of a slower time to converge to the correct estimates.

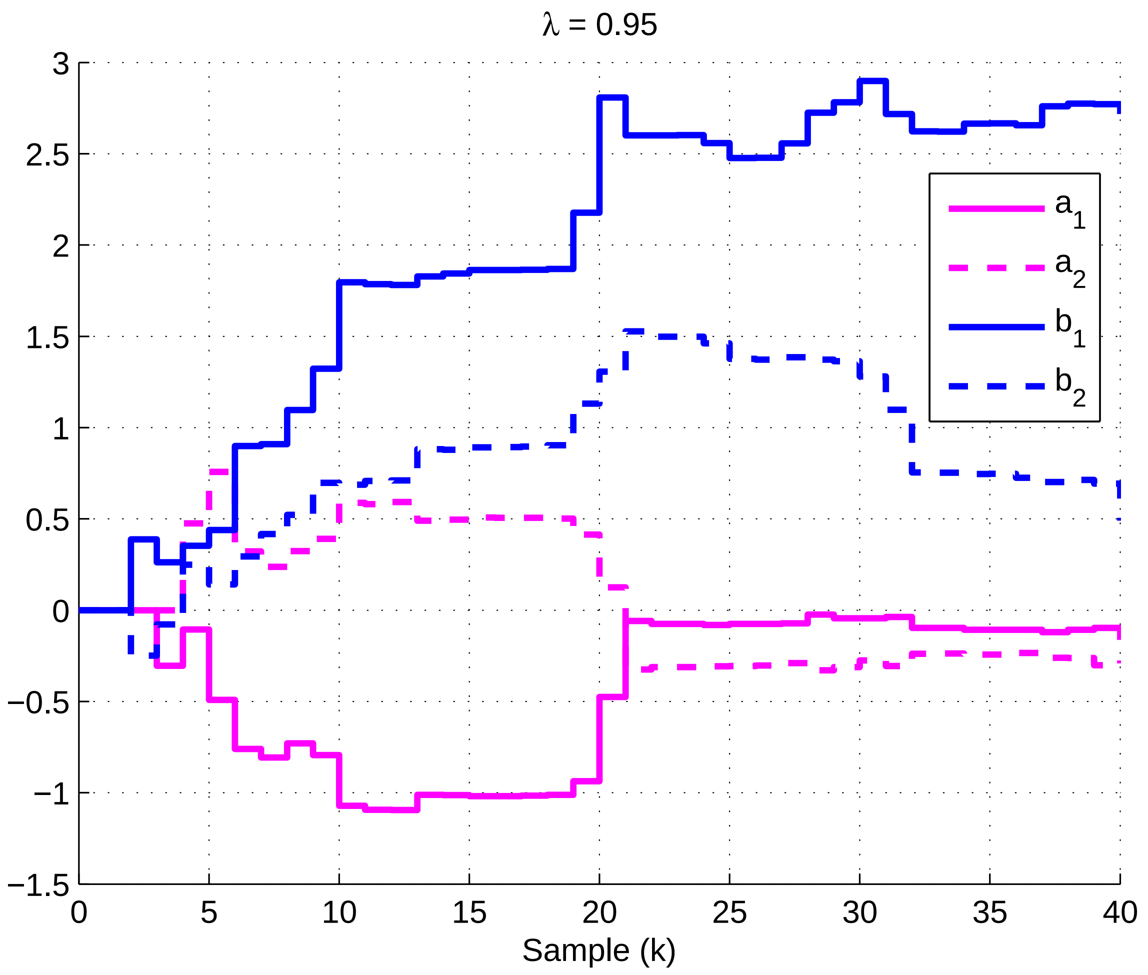

Figure 12: Results of using the recursive least squares algorithm to estimate the system parameters. $\lambda = 0.95$ in this case. Here the parameter estimates are fairly poor, especially after the system changes. This is because we are essentially not "forgetting" past data.

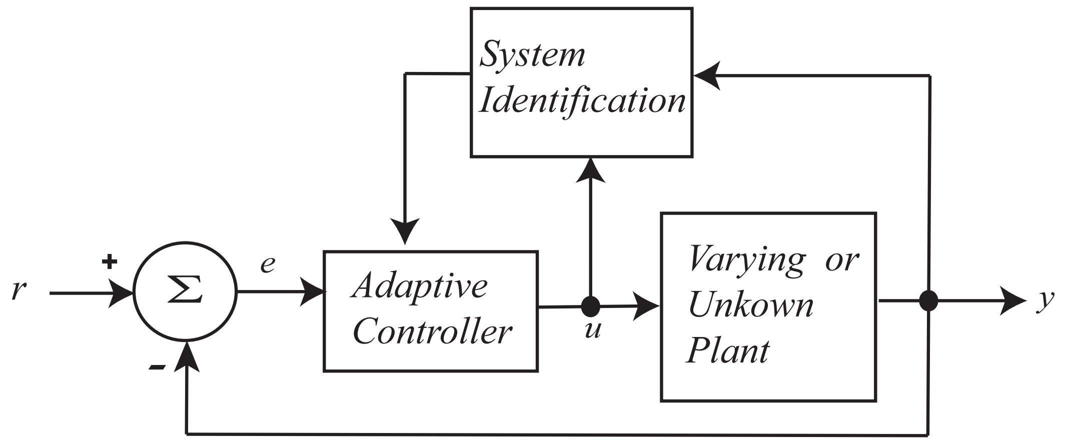

Figure 13: Schematic of an adaptive control system. The *plant* is the system we are trying to control, which may be unknown or varying over time. The *system identification* block takes the input to the plant and output of the plant and estimates the plant transfer function. Finally, the *adaptive controller* takes the transfer function estimate of the plant and the error signal, and produces a control signal.

4 Discrete-Time State Equations

In this chapter we develop discrete-time state equations assuming we have a time-invariant continuous-time state variable description of the system. The standard form for a time-invariant continuous time state variable model is

$$\begin{aligned}\dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) + Du(t)\end{aligned}$$

Here $x(t)$ is the state vector, $u(t)$ is the input vector to the plant[1], $y(t)$ is the output vector from the plant, and the A,B,C, and D matrices are constant. If these matrices were not constant this would not be a time-invariant system. A signal flow graph for this continuous-time system is shown in Figure 14. If we assume we sample this system, we will have a discrete-time state variable model of the system which we can write

$$\begin{aligned}x([k+1]T) &= Gx(kT) + Hu(kT) \\ y(kT) &= Cx(kT) + Du(kT)\end{aligned}$$

where we have explicitly shown the dependence on the sampling interval T. A signal flow graph of this discrete-time state variable model is shown in Figure 15. Note that the output equation contains the same C and D matrices as the continuous-time model. In this chapter we will show how to go from the continuous-time to the discrete-time state variable models assuming the sampling is done with a zero order hold (ZOH). We will also show how to derive the discrete-time equations when there may be delays between the input and output. We need to be able to solve the continuous-time state equations, and to do this we first need a bit of background mathematics.

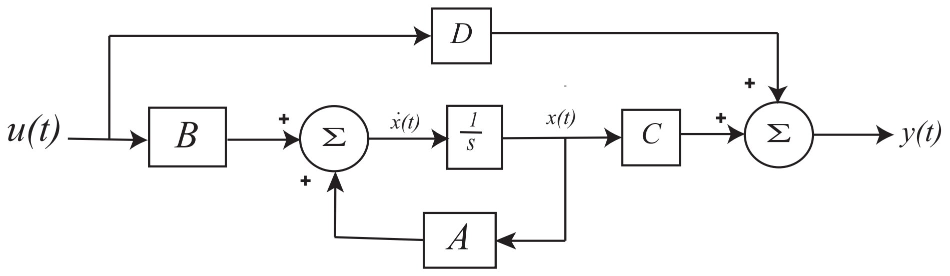

Figure 14: Continuous-time state variable model for a plant.

4.1 The Continuous-Time State Transition Matrix

The function e^{At}, where A is a *square* matrix, is a very important function in continuous-time state variable systems. If A is an $n \times n$ matrix, then as we will see, e^{At} will also be an $n \times n$ matrix. This matrix is called the *state transition matrix.* We will need to be able to compute this function and will need some of it's properties in order to derive discrete-time state variable

[1]This is not to be confused with a unit step function. The input may be a unit step function, but it is not necessarily a unit step function. This is common notation and you will need to look at the context to determine if $u(t)$ represents a unit step function or the input to the plant.

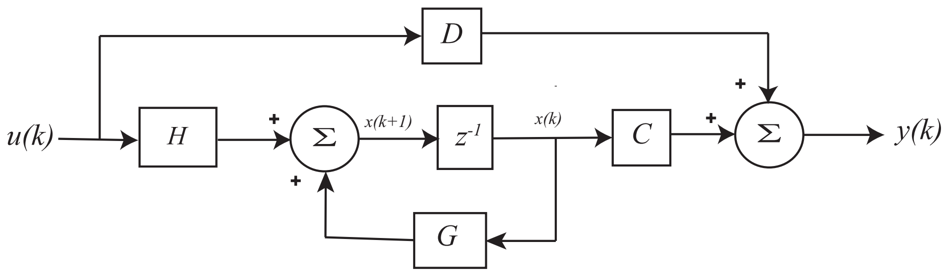

Figure 15: Discrete-time state variable model for a plant.

equations from continuous-time state variable equations.

Consider the function e^{at} for a scalar a. We know that this function has the Taylor series

$$e^{at} = \frac{1}{0!} + \frac{(at)}{1!} + \frac{(at)^2}{2!} + \ldots = \sum_{k=0}^{k=\infty} \frac{(at)^k}{k!}$$

Using this motivation, we will define the state transition matrix as

$$e^{At} = \frac{I}{0!} + \frac{At}{1!} + \frac{(At)^2}{2!} + \ldots = \sum_{k=0}^{k=\infty} \frac{(At)^k}{k!}$$

where $A^2 = A \times A$, $A^3 = A^2 \times A$, etc. Using this definition we can then compute

$$\begin{aligned}
\frac{d}{dt}\left\{e^{AT}\right\} &= \frac{d}{dt}\left[I + \frac{At}{1!} + \frac{A^2t^2}{2!} + \frac{A^3t^3}{3!} + \cdots\right] \\
&= \left[0 + A + \frac{A^2 2t}{2!} + \frac{A^3 3t^2}{3!} + \cdots\right] \\
&= A\left[I + At + \frac{A^2t^2}{2!} + \cdots\right] \\
&= Ae^{At} = e^{At}A
\end{aligned}$$

It can also be shown that $e^{At}e^{As} = e^{A(t+s)}$, for t and s *scalars*. A special case of this is when $s = -t$. The we have

$$e^{At}e^{-At} = e^{A0} = I$$

Hence

$$\left[e^{At}\right]^{-1} = e^{-At}$$

so we know what the inverse matrix of e^{At} is. We can use these results to perform integration. For example

$$\int_0^t e^{A\lambda} d\lambda = A^{-1} e^{A\lambda}\Big|_{\lambda=0}^{\lambda=t} = A^{-1}\left[e^{At} - I\right]$$

provided A^{-1} exists.

4.2 Solution of the Continuous-Time State Equations

Assume we have the continuous-time state variable model of a plant

$$\begin{aligned}\dot{x}(t) &= Ax(t) + Bu(t)\\ y(t) &= Cx(t) + Du(t)\end{aligned}$$

We can rewrite this system as

$$\dot{x}(t) - Ax(t) = Bu(t)$$

Next we *premultiply* both sides of the equation by e^{-At}, so we have

$$e^{-At}\left(\dot{x}(t) - Ax(t)\right) = e^{-At}Bu(t)$$

Now consider

$$\begin{aligned}\frac{d}{dt}\left(e^{-At}x(t)\right) &= \left(\frac{d}{dt}e^{-At}\right)x(t) + e^{-At}\left(\frac{d}{dt}x(t)\right)\\ &= -Ae^{-At}x(t) + e^{-At}\dot{x}(t)\\ &= e^{-At}\left(\dot{x}(t) - Ax(t)\right)\end{aligned}$$

since e^{-At} and A commute. Hence we have

$$\frac{d}{dt}\left(e^{-At}x(t)\right) = e^{-At}Bu(t)$$

Finally, we integrate both sides of this equation from an initial time t_0 to the current time t,

$$\begin{aligned}\int_{t_0}^{t}\frac{d}{d\lambda}\left(e^{-A\lambda}x(\lambda)\right)d\lambda &= \int_{t_0}^{t}e^{-A\lambda}Bu(\lambda)d\lambda\\ e^{-At}x(t) - e^{-At_0}x(t_0) &= \int_{t_0}^{t}e^{-A\lambda}Bu(\lambda)d\lambda\\ x(t) &= e^{A(t-t_0)}x(t_0) + \int_{t_0}^{t}e^{A(t-\lambda)}Bu(\lambda)d\lambda\end{aligned}$$

Now that we have solved the state equations, we can convert them to a discrete-time model. However, the model we end up with depends on how we are going to sample the system to get a discrete-time system. We will assume we are using a zero-order-hold (ZOH), which holds the input at a constant value over one sampling interval of length T. Specifically, we assume

$$u(t) = u(kT) \quad kT \le t < (k+1)T$$

Next we look at an interval from $t_0 = kT$ to $t = (k+1)T$. The solution to the state equations becomes

$$x([k+1]T) = e^{A([k+1]T-kT)}x(kT) + \left\{\int_{kT}^{[k+1]T}e^{A([k+1]T-\lambda)}d\lambda\right\}Bu(kT)$$

or

$$x([k+1]T) = e^{AT}x(kT) + \left\{ \int_{kT}^{[k+1]T} e^{A([k+1]T-\lambda)} d\lambda \right\} Bu(kT)$$

Notice that the integral is no longer a function of $u(t)$ so we can take both the B and $u(t)$ outside of the integral. Let's change variables, so

$$\sigma = [k+1]T - \lambda$$

With this variable change, the resulting equation is

$$x([k+1]T) = e^{AT}x(kT) + \left\{ \int_0^T e^{A\sigma} d\sigma \ B \right\} u(kT)$$

For convenience, the dependence on the sampling interval T is suppressed, and we have the resulting discrete-time state equations

$$\begin{aligned} x(k+1) &= Gx(k) + Hu(k) \\ y(k) &= Cx(k) + Du(k) \end{aligned}$$

where

$$\begin{aligned} G &= e^{AT} \\ H &= \int_0^T e^{A\lambda} d\lambda \ B \end{aligned}$$

Note that the output (second) equation does not really change when we go from a discrete-time to a continuous time state variable description of a system. The only thing we need to know now is how to compute e^{At}, which is discussed in the next section.

4.3 Computing the State Transition Matrix, e^{At}

In computing the discretized state equations, we need to be able to determine e^{At} (or e^{AT} for sampling interval T). There are three common approaches to this computation: (1) truncating the infinite sum, (2) using the Laplace transform, and (3) matching functions on the eigenvalues. The first two methods are fairly easy to understand, while the third method will just be used without justification.

4.3.1 Truncating the Infinite Sum

Since we know that by definition,

$$e^{At} = \sum_{k=0}^{k=\infty} \frac{At}{k!}$$

if the sample interval T is small enough ($t = T$) we will get a good approximation using only a few terms in the infinite series. Hence if were were to use only a few terms we would get the approximations

$$\begin{aligned} e^{At} &\approx I + At \ \ \text{(two terms)} \\ &\approx I + At + \frac{1}{2}A^2t^2 \ \ \text{(three terms)} \\ &\approx I + At + \frac{1}{2}A^2t^2 + \frac{1}{6}A^3t^3 \ \ \text{(four terms)} \end{aligned}$$

This technique generally becomes difficult if more than a few terms are needed.

4.3.2 Laplace Transform Method

In our solution to the continuous-time state variable equations, if we assume $t_0 = 0$ (the initial time is zero), $x(0) = 0$ (the initial conditions are zero), and $D = 0$, then we have

$$y(t) = C \int_0^t e^{A(t-\lambda)} Bu(\lambda) d\lambda$$

For LTI systems, if we know the input to a system, $u(t)$, and the impulse response of the system, $f(t)$, then the output of the system can be determined by using convolution,

$$y(t) = f(t) \star u(t) = \int_{-\infty}^{\infty} f(t-\lambda)u(\lambda)d\lambda$$

If we know that both the input and the system are causal (i.e., both $u(t)$ and $f(t)$ are zero for $t < 0$), then this simplifies to

$$y(t) = f(t) \star u(t) = \int_0^t f(t-\lambda)u(\lambda)d\lambda$$

If we compare this to our solution to the state equations above, we have

$$f(t) = Ce^{At}B$$

Now let's look at solving the state equations using Laplace transforms. Again we assume the initial time is zero, the initial state is zero, and the D matrix is zero. We then have initially

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) \end{aligned}$$

Taking the Laplace transform we have

$$\begin{aligned} sX(s) &= AX(s) + BU(s) \\ Y(s) &= CX(s) \end{aligned}$$

Solving for $X(s)$ and putting this into the $Y(s)$ equation we have

$$\begin{aligned} X(s) &= (sI - A)^{-1}BU(s) \\ Y(s) &= C(sI - A)^{-1}BU(s) \end{aligned}$$

Comparing time-domain and s-domain solutions, we have

$$\begin{aligned} y(t) &= f(t) \star u(t) \\ Y(s) &= F(s)U(s) \end{aligned}$$

which means

$$\begin{aligned} F(s) &= C(sI-A)^{-1}B \\ f(t) &= Ce^{At}B \end{aligned}$$

or, finally

$$e^{At} = \mathcal{L}^{-1}\left\{(sI-A)^{-1}\right\}$$

In what follows, we will be computing the inverse of a 2×2 matrix, which you are expected to be able to do. If

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

then if A^{-1} exists, it can be computed as

$$A^{-1} = \frac{1}{ad-bc}\begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

A simple way to remember this is (1) divide by the determinant, (2) switch the elements on the major diagonal, and (3) negate the off diagonal terms.

Finally, it should be noted that, as a very simple check, $e^{A0} = e^0 = I$, so if we set $t = 0$ we should get and identity matrix in our computation of e^{At}.

Example. Compute e^{At} for

$$A = \begin{bmatrix} 1 & 2 \\ 0 & 1 \end{bmatrix}$$

First we compute $sI - A$,

$$\begin{aligned} sI - A &= \begin{bmatrix} s & 0 \\ 0 & s \end{bmatrix} - \begin{bmatrix} 1 & 2 \\ 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} s-1 & -2 \\ 0 & s-1 \end{bmatrix} \end{aligned}$$

and then $(sI-A)^{-1}$,

$$(sI-A)^{-1} = \frac{1}{(s-1)^2}\begin{bmatrix} s-1 & 2 \\ 0 & s-1 \end{bmatrix} = \begin{bmatrix} \frac{1}{s-1} & \frac{2}{(s-1)^2} \\ 0 & \frac{1}{s-1} \end{bmatrix}$$

and then find the inverse Laplace transform,

$$e^{At} = \begin{bmatrix} e^t & 2te^t \\ 0 & e^t \end{bmatrix} = \mathcal{L}^{-1}\left\{(sI-A)^{-1}\right\}$$

Example. Compute e^{At} for

$$A = \begin{bmatrix} 2 & -2 \\ 2 & 6 \end{bmatrix}$$

First we compute $sI - A$,

$$\begin{aligned} sI - A &= \begin{bmatrix} s & 0 \\ 0 & s \end{bmatrix} - \begin{bmatrix} 2 & -2 \\ 2 & 6 \end{bmatrix} \\ &= \begin{bmatrix} s-2 & 2 \\ -2 & s-6 \end{bmatrix} \end{aligned}$$

and then $(sI - A)^{-1}$,

$$(sI-A)^{-1} = \frac{1}{(s-4)^2}\begin{bmatrix} s-6 & -2 \\ 2 & s-2 \end{bmatrix} = \begin{bmatrix} \frac{s-6}{(s-4)^2} & \frac{-2}{(s-4)^2} \\ \frac{2}{(s-4)^2} & \frac{s-2}{(s-4)^2} \end{bmatrix}$$

and then find the inverse Laplace transform,

$$e^{At} = \begin{bmatrix} e^{4t} - 2te^{4t} & -2te^{4t} \\ 2te^{4t} & e^{4t} + 2te^{4t} \end{bmatrix} = \mathcal{L}^{-1}\left\{(sI-A)^{-1}\right\}$$

4.3.3 Matching on Eigenvalues

This method of determining e^{At} can be proved using the Cayley-Hamilton Theorem from linear algebra. Let the characteristic polynomial of the matrix A be

$$\Delta(\lambda) = \det(\lambda I - A) = \prod_{i=1}^{i=m} (\lambda - \lambda_i)^{n_i}$$

where there are m distinct eigenvalues and n_i is the number of times eigenvalue λ_i is repeated. We assume A is an $n \times n$ square matrix. Let $f(x)$ be arbitrary functions of x and $g(x)$ be a polynomial of degree n. Then if

$$\left.\frac{d^l f(x)}{dx^l}\right|_{x=\lambda_i} = \left.\frac{d^l g(x)}{dx^l}\right|_{x=\lambda_i} \quad \text{for } l = 0, ..., n_i - 1, \quad i = 1, ..., m$$

then

$$f(A) = g(A)$$

Example. Compute A^4 for the matrix

$$A = \begin{bmatrix} 1 & 2 \\ 0 & 3 \end{bmatrix}$$

We first need to determine the characteristic equation and find the eigenvalues,

$$\begin{aligned} \lambda I - A &= \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix} - \begin{bmatrix} 1 & 2 \\ 0 & 3 \end{bmatrix} = \begin{bmatrix} \lambda - 1 & -2 \\ 0 & \lambda - 3 \end{bmatrix} \\ \Delta(\lambda) &= (\lambda - 1)(\lambda - 3) \end{aligned}$$

so the eigenvalues are $\lambda_1 = 1$ and $\lambda_2 = 3$. We next need to identify our functions,

$$\begin{aligned} f(x) &= x^4 \\ g(x) &= \alpha_0 + \alpha_1 x \end{aligned}$$

Note that $g(x)$ is a polynomial with n unknowns, where A is an $n \times n$ matrix. Next we match on the eigenvalues of A to determine the equations we need to solve for α_0 and α_1,

$$\begin{aligned} f(\lambda_1) &= 1^4 = g(\lambda_1) = \alpha_0 + \alpha_1 \\ f(\lambda_2) &= 3^4 = g(\lambda_2) = \alpha_0 + 3\alpha_1 \end{aligned}$$

We can solve these simple equations to get $\alpha_0 = -39$ and $\alpha_1 = 40$, so $g(x) = -39 + 40x$. Finally, we evaluate with the matrix A,

$$f(A) = A^4 = g(A) = -39I + 40A = \begin{bmatrix} -39 & 0 \\ 0 & -39 \end{bmatrix} + \begin{bmatrix} 40 & 80 \\ 0 & 120 \end{bmatrix} = \begin{bmatrix} 1 & 80 \\ 0 & 81 \end{bmatrix}$$

Example. Compute A^k for the matrix

$$A = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$$

We first need to determine the characteristic equation and find the eigenvalues,

$$\begin{aligned} \lambda I - A &= \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix} - \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} \lambda - 1 & 0 \\ -1 & \lambda - 1 \end{bmatrix} \\ \Delta(\lambda) &= (\lambda - 1)^2 \end{aligned}$$

so the eigenvalue is repeated, $\lambda = 1$. We next need to identify our functions,

$$\begin{aligned} f(x) &= x^k \\ g(x) &= \alpha_0 + \alpha_1 x \end{aligned}$$

Note that $g(x)$ is a polynomial with n unknowns, where A is an $n \times n$ matrix. Next we match on the eigenvalues of A to determine the equations we need to solve for α_0 and α_1,

$$\begin{aligned} f(\lambda) &= 1^k = g(\lambda) = \alpha_0 + \alpha_1 \\ \left.\frac{df(x)}{dx}\right|_{x=\lambda} &= \left. kx^{k-1}\right|_{x=\lambda} = k = \left.\frac{dg(x)}{dx}\right|_{x=\lambda} = \alpha_1 \end{aligned}$$

We can solve these simple equations to get $\alpha_0 = 1 - k$ and $\alpha_1 = k$, so $g(x) = (1 - k) + kx$. Note that in this case the coefficients are functions of k, they are not constants. Finally, we evaluate with the matrix A,

$$f(A) = A^k = g(A) = (1 - k)I + kA = \begin{bmatrix} 1 - k & 0 \\ 0 & 1 - k \end{bmatrix} + \begin{bmatrix} k & 0 \\ k & k \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ k & 1 \end{bmatrix}$$

Example. Compute e^{At} for the matrix

$$A = \begin{bmatrix} 1 & 2 \\ 0 & 1 \end{bmatrix}$$

We first need to determine the characteristic equation and find the eigenvalues,

$$\begin{aligned} \lambda I - A &= \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix} - \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} \lambda - 1 & 0 \\ -1 & \lambda - 1 \end{bmatrix} \\ \Delta(\lambda) &= (\lambda - 1)^2 \end{aligned}$$

so the eigenvalue is repeated, $\lambda = 1$. We next need to identify our functions,

$$\begin{aligned} f(x) &= e^{xt} \\ g(x) &= \alpha_0 + \alpha_1 x \end{aligned}$$

Next we match on the eigenvalues of A to determine the equations we need to solve for α_0 and α_1,

$$\begin{aligned} f(\lambda) &= e^t = g(\lambda) = \alpha_0 + \alpha_1 \\ \left.\frac{df(x)}{dx}\right|_{x=\lambda} &= \left. te^{xt}\right|_{x=\lambda} = te^t = \left.\frac{dg(x)}{dx}\right|_{x=\lambda} = \alpha_1 \end{aligned}$$

We can solve these simple equations to get $\alpha_0 = e^t - te^t$ and $\alpha_1 = te^t$, so $g(x) = (e^t - te^t) + te^t x$. Note that in this case the coefficients are functions of t, they are not constants. Finally, we evaluate with the matrix A,

$$f(A) = e^{At} = g(A) = (e^t - te^t)I + te^t A = \begin{bmatrix} e^t & 2te^t \\ 0 & e^t \end{bmatrix}$$

Example Compute e^{At} for the matrix

$$A = \begin{bmatrix} 0 & 1 \\ -4 & -5 \end{bmatrix}$$

We first need to determine the characteristic equation and find the eigenvalues,

$$\begin{aligned} \lambda I - A &= \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix} - \begin{bmatrix} 0 & 1 \\ -4 & -5 \end{bmatrix} = \begin{bmatrix} \lambda & -1 \\ 4 & \lambda + 5 \end{bmatrix} \\ \Delta(\lambda) &= (\lambda + 1)(\lambda + 4) \end{aligned}$$

so the eigenvalues are $\lambda_1 = -4$ and $\lambda_2 = -1$. We next need to identify our functions,

$$\begin{aligned} f(x) &= e^{xt} \\ g(x) &= \alpha_0 + \alpha_1 x \end{aligned}$$

Next we match on the eigenvalues of A to determine the equations we need to solve for α_0 and α_1,

$$\begin{aligned} f(\lambda_1) &= e^{-4t} = g(\lambda_1) = \alpha_0 - 4\alpha_1 \\ f(\lambda_2) &= e^{-t} = g(\lambda_2) = \alpha_0 - \alpha_1 \end{aligned}$$

We can solve these equations to get $\alpha_0 = \frac{4}{3}e^{-t} - \frac{1}{3}e^{-4t}$ and $\alpha_1 = \frac{1}{3}e^{-t} - \frac{1}{3}e^{-4t}$, so $g(x) = (\frac{4}{3}e^{-t} - \frac{1}{3}e^{-4t}) + (\frac{1}{3}e^{-t} - \frac{1}{3}e^{-4t})x$. Note that in this case the coefficients are functions of t, they are not constants. Finally, we evaluate with the matrix A,

$$f(A) \;=\; e^{At} = g(A) = (\frac{4}{3}e^{-t} - \frac{1}{3}e^{-4t}) + (\frac{1}{3}e^{-t} - \frac{1}{3}e^{-4t})A = \begin{bmatrix} \frac{4}{3}e^{-t} - \frac{1}{3}e^{-4t} & \frac{1}{3}e^{-t} - \frac{1}{3}e^{-4t} \\ -\frac{4}{3}e^{-t} + \frac{4}{3}e^{-4t} & -\frac{1}{3}e^{-t} + \frac{4}{3}e^{-4t} \end{bmatrix}$$

Example. Compute e^{At} for the matrix

$$A \;=\; \begin{bmatrix} 0 & 0 & -2 \\ 0 & 1 & 0 \\ 1 & 0 & 3 \end{bmatrix}$$

The characteristic equation for this system is

$$\Delta(\lambda) \;=\; (\lambda - 1)^2(\lambda + 2)$$

so the eigenvalues are $\lambda_1 = 1$ (repeated) and $\lambda_2 = 2$. We next need to identify our functions,

$$\begin{aligned} f(x) &= e^{xt} \\ g(x) &= \alpha_0 + \alpha_1 x + \alpha_2 x^2 \end{aligned}$$

Next we match on the eigenvalues of A to determine the equations we need to solve for α_0, α_1, and α_2

$$\begin{aligned} f(\lambda_1) &= e^t = g(\lambda_1) = \alpha_0 + \alpha_1 + \alpha_2 \\ \left.\frac{df(x)}{dx}\right|_{x=\lambda_1} &= te^t = \left.\frac{dg(x)}{dx}\right|_{x=\lambda_1} = \alpha_1 + 2\alpha_2 \\ f(\lambda_2) &= e^{2t} = g(\lambda_2) = \alpha_0 + 2\alpha_1 + 4\alpha_2 \end{aligned}$$

We then need so solve the system of equations,

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 0 \\ 1 & 2 & 4 \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \end{bmatrix} = \begin{bmatrix} e^t \\ te^t \\ e^{2t} \end{bmatrix}$$

It should be pointed out that the most important part of this method is the knowledge that we can always write

$$e^{At} \;=\; \alpha_0(t)I + \alpha_1(t)A + \alpha_2(t)A^2 + \ldots + \alpha_{n-1}(t)A^{n-1}$$

This fact will be very important later when we talk about *controllability*.

4.4 Discretization with Delays in the Input

We would now like to extend our development to the case where there are delays in the input. We can model such a system as

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t - \tau) \\ y(t) &= Cx(t) + Du(t) \end{aligned}$$

where τ is the delay. Following the same procedure as before, we will have

$$x(t) \quad = \quad e^{A(t-t_0)}x(t_0) + \int_{t_0}^{t} e^{A(t-\lambda}Bu(\lambda-\tau)d\tau$$

Again assume $t_0 = kT$ and $t = [k+1]T$, so we have

$$x([k+1]T) \quad = \quad e^{AT}x(kT) + \int_{kT}^{[k+1]T} e^{A([k+1]T-\lambda)}Bu(\lambda-\tau)d\lambda$$

Let's assume $0 \leq \tau \leq T$ and look at $u(t)$ and the delayed signal $u(t-\tau)$, shown in Figure 16. In order to evaluate the integral in our expression above we will need to break the integral up into two pieces as follows:

$$\begin{aligned}\int_{kT}^{[k+1]T} e^{A([k+1]T-\lambda)}Bu(\lambda-\tau)d\lambda &= \int_{kT}^{kT+\tau} e^{A([k+1]T-\lambda)}Bu([k-1]T)d\lambda + \int_{kT+\tau}^{[k+1]T} e^{A([k+1]T-\lambda)}Bu(kT)d\lambda \\ &= H_1u([k-1]T) + H_0u(kT)\end{aligned}$$

where

$$\begin{aligned}H_0 &= \int_{kT+\tau}^{[k+1]T} e^{A([k+1]T-\lambda)}d\lambda\ B = \int_0^{T-\tau} e^{A\sigma}d\sigma\ B \\ H_1 &= \int_{kT}^{kT+\tau} e^{A([k+1]T-\lambda)}d\lambda\ B = e^{A(T-\tau)}\int_0^{\tau} e^{A\sigma}d\sigma\ B\end{aligned}$$

so we have

$$x([k+1]T) \quad = \quad Gx(kT) + H_0u(kT) + H_1u([k-1]T)$$

Suppressing the dependence on the sampling interval T and writing as a state variable model we have

$$\left[\begin{array}{c} x(k+1) \\ u(k) \end{array}\right] \quad = \quad \left[\begin{array}{cc} G & H_1 \\ 0 & 0 \end{array}\right]\left[\begin{array}{c} u(k) \\ u(k-1) \end{array}\right] + \left[\begin{array}{c} H_0 \\ I \end{array}\right]u(k)$$

where

$$\begin{aligned}G &= e^{AT} \\ H_0 &= \int_0^{T-\tau} e^{A\sigma}d\sigma\ B \\ H_1 &= \int_0^{T} e^{A\sigma}d\sigma\ B\end{aligned}$$

For the special case where $\tau = T$ (a 1 sample delay) we have

$$\begin{aligned}G &= e^{AT} \\ H_0 &= \int_0^{0} e^{A\sigma}d\sigma = 0 \\ H_1 &= \int_0^{T} e^{a\sigma}d\sigma\ B = H\end{aligned}$$

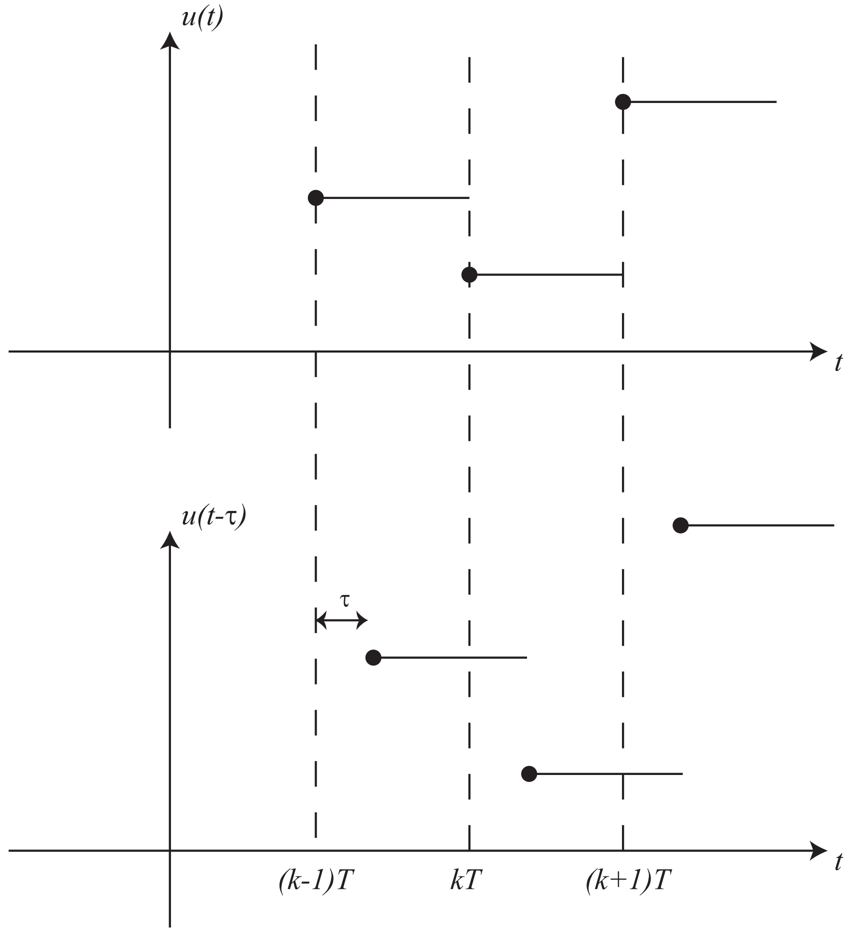

Figure 16: Original and delayed (by an amount τ) input.

The state equations are then

$$\begin{bmatrix} x(k+1) \\ u(k) \end{bmatrix} = \begin{bmatrix} G & H \\ 0 & 0 \end{bmatrix} \begin{bmatrix} u(k) \\ u(k-1) \end{bmatrix} + \begin{bmatrix} 0 \\ I \end{bmatrix} u(k)$$

Example. Consider the continuous-time double integrator system

$$\begin{aligned} \dot{x}(t) &= \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t) \\ y(t) &= \begin{bmatrix} 1 & 0 \end{bmatrix} x(t) \end{aligned}$$

A double integrator system with a delay of τ in the input can be described as

$$\begin{aligned} \dot{x}(t) &= \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t-\tau) \\ y(t) &= \begin{bmatrix} 1 & 0 \end{bmatrix} x(t) \end{aligned}$$

We need to compute e^{At}, so we'll use the Laplace transform method,

$$\begin{aligned}
sI - A &= \begin{bmatrix} s & 0 \\ 0 & s \end{bmatrix} - \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} s & -1 \\ 0 & s \end{bmatrix} \\
(sI - A)^{-1} &= \frac{1}{s^2}\begin{bmatrix} s & 1 \\ 0 & s \end{bmatrix} = \begin{bmatrix} \frac{1}{s} & \frac{1}{s^2} \\ 0 & \frac{1}{s} \end{bmatrix}
\end{aligned}$$

so

$$e^{At} = \begin{bmatrix} 1 & t \\ 0 & 1 \end{bmatrix}$$

Then

$$\begin{aligned}
G &= e^{AT} = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} \\
H_0 &= \int_0^{T-\tau} e^{A\sigma} d\sigma B = \int_0^{T-\tau} \begin{bmatrix} 1 & \sigma \\ 0 & 1 \end{bmatrix} d\sigma \begin{bmatrix} 0 \\ 1 \end{bmatrix} \\
&= \begin{bmatrix} (T-\tau) & \frac{(T-\tau)^2}{2} \\ 0 & (T-\tau) \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} \\
&= \begin{bmatrix} \frac{(T-\tau)^2}{2} \\ T-\tau \end{bmatrix} \\
H_1 &= e^{A(T-\tau)} \int_0^{\tau} e^{A\sigma} d\sigma B = \begin{bmatrix} 1 & T-\tau \\ 0 & 1 \end{bmatrix} \int_0^{\tau} \begin{bmatrix} 1 & \sigma \\ 0 & 1 \end{bmatrix} d\sigma \begin{bmatrix} 0 \\ 1 \end{bmatrix} \\
&= \begin{bmatrix} 1 & T-\tau \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \tau & \tau^2 \\ 0 & \tau \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & T-\tau \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{\tau^2}{2} \\ \tau \end{bmatrix} \\
&= \begin{bmatrix} \frac{\tau^2}{2} + T\tau - \tau^2 \\ \tau \end{bmatrix} = \begin{bmatrix} \tau(T - \frac{\tau}{2}) \\ \tau \end{bmatrix}
\end{aligned}$$

so our new state model of the discretized system is

$$\begin{aligned}
\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \\ u(k) \end{bmatrix} &= \begin{bmatrix} 1 & T & \tau(T - \frac{\tau}{2}) \\ 0 & 1 & \tau \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \\ u(k-1) \end{bmatrix} + \begin{bmatrix} \frac{(T-\tau)^2}{2} \\ T-\tau \\ 1 \end{bmatrix} u(k) \\
y(k) &= \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \\ u(k-1) \end{bmatrix}
\end{aligned}$$

4.5 State Variable to Transfer Function

Just as in continuous-time it is possible to go from a state variable description to a transfer function description of a system. Doing this sometimes helps with different concepts, such as pole placement, steady state error, and the speed of response. Consider the usual state-variable description of a discrete-time system

$$\begin{aligned}
x(k+1) &= Gx(k) + hu(k) \\
y(k) &= Cx(k) + Du(k)
\end{aligned}$$

Taking the z-transform of the equation and assuming the system is initially at rest, we get

$$\begin{aligned} zX(z) &= GX(z) + HU(z) \\ Y(z) &= CX(z) + DU(z) \end{aligned}$$

Solving for $X(z)$ and putting this into the equation for $Y(z)$ we get

$$\begin{aligned} X(z) &= (zI - G)^{-1} HU(z) \\ Y(z) &= C\,(zI - G)^{-1} HU(z) \end{aligned}$$

Hence the transfer function between the input and output (the plant) is

$$G_p(z) = C\,(zI - G)^{-1} H$$

<u>**Example.**</u> Find the equivalent transfer function for the following state variable model

$$\begin{aligned} x(k+1) &= \begin{bmatrix} 0.1 & 0 \\ 0.2 & 0.3 \end{bmatrix} x(k) + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u(k) \\ y(k) &= \begin{bmatrix} 0 & 1 \end{bmatrix} \end{aligned}$$

First we compute $zI - G$,

$$\begin{bmatrix} z & 0 \\ 0 & z \end{bmatrix} - \begin{bmatrix} 0.1 & 0 \\ 0.2 & 0.3 \end{bmatrix} = \begin{bmatrix} z-0.1 & 0 \\ -0.2 & z-0.3 \end{bmatrix}$$

then $(zI - G)^{-1}$,

$$(zI - G)^{-1} = \frac{1}{(z-0.1)(z-0.3)} \begin{bmatrix} z-0.3 & 0 \\ 0.2 & z-0.1 \end{bmatrix}$$

Finally, $G_p(z) = C\,(zI - G)^{-1} H$, so

$$\begin{aligned} G_p(z) &= \frac{1}{(z-0.1)(z-0.3)} \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} z-0.3 & 0 \\ 0.2 & z-0.1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \\ &= \frac{0.2}{(z-0.1)(z-0.3)} \end{aligned}$$

4.6 Poles and Eignevalues

For continuous-time systems we know the poles of the system are given by the determinant of $sI - A$, or the roots of the characteristic equation $\Delta(s) = |sI - A| = 0$. These poles are the eigenvalues of the A matrix. Similarly, for a discrete-time system, the poles of the system are given by the determinant of $zI - G$, or the roots of the characteristic equation $\Delta(z) = |zI - G| = 0$. The poles of the discrete-time system are also the eigenvalues of the G matrix. If all of the poles of a discrete-time system is at the origin, this is referred to as *deadbeat response*. The deadbeat response is the fastest possible response of a discrete-time system. However, the control effort required for deadbeat response may be too large for actual implementation.

5 Controllability and Observability

In this chapter we introduce the ideas of controllability and observability. These are very important concepts in this course, and in control systems in general. If a system is controllable, then we can find an input to take the system from any initial state to any final state in at most n time steps, where $G \in R^{n \times n}$. More importantly, controllability allows us to use state variable feedback to place the closed loop poles where ever we want to, as we will see in the next chapter. Observability means that if we know the input and measure the output of a system for n time steps, we can determine the initial state of the system, $x(0)$. It also means that we can construct *observers* to estimate the state of the system. Since for state variable feedback we need knowledge of all of the state this is a very useful property. Sometimes it is much cheaper and easier to estimate the states than to try and measure the states directly. Of course, the accuracy with which we can measure the states depends to some extend on how accurate our model of the system is. However, in order to understand where the tests for controllability and observability come from, we need to first review some concepts from linear algebra.

5.1 Linear Algebra Review

In this section we will review a number of concepts from linear algebra that are of particular importance in justifying our tests for controllability and observability. These concepts will also be used in subsequent chapters, so ignore learning them now at your own peril!

5.1.1 Linear Independence

We will denote the set of complex numbers as $\mathcal{C}$. A set of vectors $x_1, x_2, x3, \ldots, x_n$ is said to be linearly dependent over $\mathcal{C}$ if there are constants $\alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_n \in \mathcal{C}$ not all zero so that

$$\alpha_1 x_1 + \alpha_2 x_2 + \ldots + \alpha_n x_n \quad = \quad 0$$

If the only set of $\alpha_i \in \mathcal{C}$ for which this expression holds is $\alpha_1 = \alpha_2 = \ldots = \alpha_n = 0$, then the vectors $x_1, x_2, \ldots, x_n$ are said to be linearly independent over the field $\mathcal{C}$. Note that, technically, linear independence is a function of the *field* the constants α_i are members of. However, in this course we will not emphasize this.

Example. Are the vectors

$$x_1 = \begin{bmatrix} 1 \\ 0 \\ 2 \end{bmatrix}, \; x_2 = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}, \; x_3 = \begin{bmatrix} 3 \\ 1 \\ 4 \end{bmatrix}$$

linearly independent over $\mathcal{C}$? Clearly $2x_1 + x_2 - x_3 = 0$ ($\alpha_1 = 2, \alpha_2 = 1, \alpha_3 = -1$), so these vectors are linearly dependent.

Example. Are the vectors

$$x_1 = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}, \; x_2 = \begin{bmatrix} 2 \\ 0 \\ 1 \end{bmatrix}, \; x_3 = \begin{bmatrix} 0 \\ 1 \\ 2 \end{bmatrix}$$

linearly independent over $\mathcal{C}$? In this case, the solution is not as obvious, so let's write out what we need. We want to find constants $\alpha_i \in \mathcal{C}$ so that

$$\alpha_1 x_1 + \alpha_2 x_2 + \alpha_3 x_3 \;=\; 0$$

We can write this in matrix form as

$$\begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix} \;=\; \begin{bmatrix} 1 & 2 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 2 \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix} = 0$$

Then by elementary row operations we have

$$\begin{bmatrix} 1 & 2 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 2 \end{bmatrix} \;\rightarrow\; \begin{bmatrix} 1 & 2 & 0 \\ 0 & -2 & 1 \\ 0 & 1 & 2 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 2 & 0 \\ 0 & 1 & -0.5 \\ 0 & 0 & 2.5 \end{bmatrix}$$

We then solve from the bottom row to the top row. From the bottom row we have $2.5\alpha_3 = 0$, which means $\alpha_3 = 0$. From the middle row we then have $\alpha_2 - 0.5\alpha_3 = 0$, which means $\alpha_2 = 0$. Finally, from the top row we have $\alpha_1 + 2\alpha_2 = 0$ from which we get $\alpha_1 = 0$. Since the only solution is for all of the $\alpha_i = 0$ the vectors are linearly independent.

Example. Assume the vectors u, v, and w are linearly independent vectors. Are the vectors $u+v$, $u+v+w$ and $v-w$ linearly independent? We need to find α_1, α_2, and α_3 so that

$$\alpha_1(u+v) + \alpha_2(u+v+w) + \alpha_3(v-w) \;=\; 0$$

We can rewrite these equations as

$$(\alpha_1+\alpha_2)u + (\alpha_1+\alpha_2+\alpha_3)v + (\alpha_2-\alpha_3)w \;=\; 0$$

Since we know u, v, and w are linearly independent, we know the coefficients in the above equation must all be zero,

$$\begin{aligned} \alpha_1+\alpha_2 &= 0 \\ \alpha_1+\alpha_2+\alpha_3 &= \;= 0 \\ \alpha_2-\alpha_3 &= 0 \end{aligned}$$

We could write this as a matrix equation, but in this case it may be easiest to just solve the first and third equations in terms of α_2 and then put it into the second equation,

$$\begin{aligned} \alpha_1 &= -\alpha_2 \\ \alpha_3 &= \alpha_2 \end{aligned}$$

inserting these results into the middle equation we get

$$(-\alpha_2) + \alpha_2 + (\alpha_2) \;=\; 0$$

which indicates $\alpha_2 = 0$, so both α_1 and α_3 must also be zero. Hence the new vectors are linearly independent.

5.1.2 Rank of a Matrix

We can consider both the rows of a matrix and the columns of a matrix as vectors. Then the *rank* of a matrix is the number of linearly independent rows or columns. It doesn't matter if we look at the rows or columns of a matrix, we will get the same answer answer. It should be obvious that if A is an $m \times n$ matrix, the rank of A, which we will denote as $r(A)$ can be at most the smaller of n or m,

$$r(A) \leq \min(m, n)$$

If $r(A) = \min(n, m)$ then we say that A has *full rank*.

Example. Consider the matrix

$$A = \begin{bmatrix} 1 & 2 & 3 & 0 \\ 0 & 2 & 3 & 1 \end{bmatrix}$$

We know that the maximum value of $r(A)$ is 2, and this is in fact the rank of this matrix. It is probably easiest to look at the number of linearly independent columns in this case, and once we find two linearly independent columns we know that is the most we can find. This matrix has full rank.

Example. Consider the matrix

$$A = \begin{bmatrix} 1 & 1 & 0 \\ 2 & 1 & 0 \\ 3 & 2 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

We know the maximum value of $r(A)$ is 3, and this is the rank of this matrix. Clearly the first, second, and fourth rows are linearly independent, and the third row is the sum of these rows. This matrix has full rank.

Example. Consider the matrix

$$A = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 2 & 2 & 0 & 0 \\ 0 & 0 & 0 & 3 \end{bmatrix}$$

We know the maximum value of $r(A)$ is 3. In this case the first row is a linear combination of the second and third rows, so there are only two linearly independent rows, so the rank of this matrix is two.

Example. Consider the matrix

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 2 & 0 \\ 3 & 4 & 0 \end{bmatrix}$$

We know that maximum value of $r(A)$ is 3. Clearly the first and second columns are linearly independent, so the rank of this matrix is 2.

5.1.3 Unique Representation of a Vector

Let's assume we have linearly independent vectors $x_1, x_1, \ldots, x_p$. If for any vector y we can write

$$y = \alpha_1 x_1 + \alpha - 2x_2 + \ldots + \alpha_p x_p$$

then the expansion coefficients $\alpha_1, \alpha_2, \ldots, \alpha_p$ are *unique*.

To show that this is true, we'll use contradiction. We will assume there is another (different) set of expansion coefficients $\beta_1, \beta_2, \ldots, \beta_p$ and then show that this leads to a contradiction. By hypothesis we have that

$$y = \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_p x_p$$

Subtracting this expression from the original expression we get

$$0 = (\alpha_1 - \beta_1)x_1 + (\alpha_2 - \beta_2)x_2 + \ldots + (\alpha_p - \beta_p)x_p$$

Since we know the vectors x_i are linearly independent, we know the only way the above expression can be true is if each coefficient is zero, which means $\alpha_i = \beta_i$, and we have a contradiction. Hence the expansion coefficients are unique if the vectors are linearly independent.

5.1.4 Dimension of a Space

The *dimension* of a space is defined to be the number of linearly independent vectors in that space. For most of this course we will be dealing with either $\mathcal{R}^n$ or $\mathcal{C}^n$, which contain vectors with n real or complex components. Both of these are n dimensional spaces.

5.1.5 The Cayley-Hamilton Theorem

In state variable control systems, such used in this course, one of the most important and useful theorems from linear algebra is the Cayley-Hamilton Theorem. In its most basic form we can state this theorem as:

Cayley-Hamilton Theorem: *A matrix satisfies its own characteristic equation.*

This seemingly simple theorem (and its corollaries) lead to some very important results. The most important result for us at this point is that if we have a matrix A that is $n \times n$, we can always write A^n in terms of lower powers of A, i.e. in terms of $A^{n-1}, A^{n-2}, \ldots, A, I$. We will next illustrate this will a few examples.

Example. Consider the matrix

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

We need to find the characteristic equation

$$\Delta(\lambda) = \det[\lambda I - A]$$

First we compute $\lambda I - A$,

$$\begin{aligned}\lambda I - A &= \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix} - \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \\ &= \begin{bmatrix} \lambda - 1 & -2 \\ -3 & \lambda - 4 \end{bmatrix}\end{aligned}$$

Then

$$\begin{aligned}\Delta(\lambda) &= \det \begin{bmatrix} \lambda - 1 & -2 \\ -3 & \lambda - 4 \end{bmatrix} \\ &= (\lambda - 1)(\lambda - 4) - (-3)(-2) = \lambda^2 - 5\lambda - 2\end{aligned}$$

So by the Cayley-Hamilton Theorem we have

$$\Delta(A) = A^2 - 5A - 2I = 0$$

or

$$A^2 = 5A + 2I$$

Example. Consider the matrix

$$A = \begin{bmatrix} -5 & 2 \\ 0 & 1 \end{bmatrix}$$

We need to find the characteristic equation

$$\Delta(\lambda) = \det[\lambda I - A]$$

First we compute $\lambda I - A$,

$$\begin{aligned}\lambda I - A &= \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix} - \begin{bmatrix} -5 & 2 \\ 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} \lambda + 5 & -2 \\ 0 & \lambda - 1 \end{bmatrix}\end{aligned}$$

Then

$$\begin{aligned}\Delta(\lambda) &= \det \begin{bmatrix} \lambda + 5 & -2 \\ 0 & \lambda - 1 \end{bmatrix} \\ &= (\lambda + 5)(\lambda - 1) - (0)(-2) = \lambda^2 + 4\lambda - 5\end{aligned}$$

So by the Cayley-Hamilton Theorem we have

$$\Delta(A) = A^2 + 4A - 5I = 0$$

or

$$A^2 = -4A + 5I$$

5.2 Controllability

Controllability is one of the most important concepts in control systems. There are a number of equivalent definitions of controllability. In this course we will concentrate on the following three equivalent definitions of controllability:

- A system is *controllable* if there exists a control signal $u(k)$ that takes the system from the origin to any possible state in n steps, where G is an $n \times n$ matrix.
- A system is *controllable* if there exists a control signal $u(k)$ that takes the system from any possible initial state to any possible final state in n steps, where G is an $n \times n$ matrix.
- A system is *controllable* if we can use state variable feedback to place all of the system poles where ever we want

Note that if we meet any one of the conditions of controllability we meet all of the conditions. In this chapter we concentrate on the first two definitions, in the following chapter when we try and use state variable feedback to place the poles we will concentrate on the third definition. In what follows we will show that the discrete-time system

$$\begin{aligned} x(k+1) &= Gx(k) + Hu(k) \\ y(k) &= Cx(k) + Du(k) \end{aligned}$$

where $G \in \mathcal{R}^{n \times n}$ is controllable if the *controllability matrix*

$$\left[\begin{array}{cccccc} G^{n-1}H & G^{n-2}H & \cdots & GH & H \end{array} \right]$$

has rank n. Note that controllability is only a function of G and H.

Example. Let's consider the following discrete-time state variable system,

$$\begin{aligned} x(k+1) &= Gx(k) + Hu(k) \\ y(k) &= Cx(k) + Du(k) \end{aligned}$$

where

$$G = \begin{bmatrix} 0 & 1 \\ 2 & 3 \end{bmatrix}, \quad H = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \quad x(0) = \begin{bmatrix} 1 \\ -2 \end{bmatrix}$$

Now let's start marching our system through time. Our goal will be to determine if we can go from our initial state $x(0)$ to any final state $x(2)$. If this is possible, then our system is controllable. We have

$$\begin{aligned} x(1) &= Gx(0) + Hu(0) \\ x(2) &= Gx(1) + Hu(1) = G\left[Gx(0) + Hu(0)\right] + Hu(1) = G^2x(0) + GHu(0) + Hu(1) \end{aligned}$$

We can rewrite this equation as

$$x(2) - G^2x(0) = \Delta x = \left[\begin{array}{cc} GH & H \end{array} \right] \left[\begin{array}{c} u(0) \\ u(1) \end{array} \right]$$

Now the vector Δx can be *any* vector in $\mathcal{R}^2$, so if we want to be able to solve this system of equations, the matrix

$$\left[\begin{array}{cc} GH & H \end{array}\right] = \left[\begin{array}{cc} 2 & 1 \\ 8 & 2 \end{array}\right]$$

must have rank 2, or there must be two linearly independent row or columns in this matrix. This matrix has rank 2, so the system is controllable.

Example. Let's consider the following discrete-time state variable system,

$$\begin{aligned} x(k+1) &= Gx(k) + Hu(k) \\ y(k) &= Cx(k) + Du(k) \end{aligned}$$

where

$$G = \left[\begin{array}{cc} 1 & 0 \\ 0 & 2 \end{array}\right], \quad H = \left[\begin{array}{c} 1 \\ 0 \end{array}\right], \quad x(0) = \left[\begin{array}{c} 1 \\ 0 \end{array}\right]$$

Now let's start marching our system through time. Our goal will be to determine if we can go from our initial state $x(0)$ to any final state $x(2)$. If this is possible, then our system is controllable. We have

$$\begin{aligned} x(1) &= Gx(0) + Hu(0) \\ x(2) &= Gx(1) + Hu(1) = G\left[Gx(0) + Hu(0)\right] + Hu(1) = G^2x(0) + GHu(0) + Hu(1) \end{aligned}$$

We can rewrite this equation as

$$x(2) - G^2x(0) = \Delta x = \left[\begin{array}{cc} GH & H \end{array}\right]\left[\begin{array}{c} u(0) \\ u(1) \end{array}\right]$$

Now the vector Δx can be *any* vector in $\mathcal{R}^2$, so if we want to be able to solve this system of equations, the matrix

$$\left[\begin{array}{cc} GH & H \end{array}\right] = \left[\begin{array}{cc} 1 & 1 \\ 0 & 0 \end{array}\right]$$

must have rank 2, or there must be two linearly independent row or columns in this matrix. This matrix has rank 1, so the system is not controllable.

At this point, you might wonder why we stop at $n = 2$ for this system. Why not allow more inputs? This is a good question, and to answer it we need to use the Cayely-Hamilton Theorem. Let's find the characteristic equation for G

$$\begin{aligned} \lambda I - G &= \left[\begin{array}{cc} \lambda & 0 \\ 0 & \lambda \end{array}\right] - \left[\begin{array}{cc} 1 & 0 \\ 0 & 2 \end{array}\right] = \left[\begin{array}{cc} \lambda - 1 & 0 \\ 0 & \lambda - 2 \end{array}\right] \\ \Delta(\lambda) &= (\lambda - 1)(\lambda - 2) = \lambda^2 - 3\lambda + 2 \end{aligned}$$

so we know $G^2 = 3G - 2I$. Now let's look at the next time step,

$$\begin{aligned} x(3) &= Gx(2) + Hu(2) = G[G^2x(0) + GHu(0) + Hu(1)] + Hu(2) \\ &= G^3x(0) + G^2Hu(0) + GHu(1) + Hu(2) \\ &= G^3x(0) + [3G - 2I]Hu(0) + GHu(1) + Hu(2) \\ &= G^3x(0) + GH[3u(0) + u(1)] + H[-2u(0) + u(2)] \end{aligned}$$

which we can write as

$$x(3) - G^3x(0) = \Delta x \;=\; \begin{bmatrix} GH & H \end{bmatrix} \begin{bmatrix} 3u(0) + u(1) \\ -2u(0) + u(2) \end{bmatrix}$$

or

$$\Delta x \;=\; \begin{bmatrix} GH & H \end{bmatrix} \tilde{u}$$

Again, Δx can be *any* vector in $\mathcal{R}^2$, so in order to be able to solve this, the controllability matrix must have rank equal to n, which is 2.

Let's next look at $x(4)$,

$$\begin{aligned} x(4) &= Gx(3) + Hu(3) \\ &= G\Big\{G^3x(0) + GH[3u(0) + u(1)] + H[-2u(0) + u(2)]\Big\} + Hu(3) \\ &= G^4x(0) + G^2H[3u(0) + u(1)] + GH[-2u(0) + u(2)] + Hu(3) \\ &= G^4x(0) + [3G - 2I]H[3u(0) + u(1)] + GH[-2u(0) + u(2)] + Hu(3) \\ &= G^4x(0) + GH[9u(0) + 3u(1)] + H[-6u(0) - 2u(1)] + GH[-2u(0) + u(2)] + Hu(3) \\ &= G^4x(0) + GH[7u(0) + 3u(1) + u(2)] + H[-6u(0) - 2u(1) + u(3)] \end{aligned}$$

which we can write as

$$x(3) - G^4x(0) = \Delta x \;=\; \begin{bmatrix} GH & H \end{bmatrix} \begin{bmatrix} 7u(0) + 3u(1) + u(2) \\ -6u(0) - 2u(1) + u(3) \end{bmatrix}$$

or

$$\Delta x \;=\; \begin{bmatrix} GH & H \end{bmatrix} \tilde{u}$$

Again, Δx can be *any* vector in $\mathcal{R}^2$, so in order to be able to solve this, the controllability matrix must have rank equal to n, which is 2.

It should be pretty clear at this point that we could use as many time steps as we want, but once we get to n time steps, where $G \in \mathcal{R}^{\,n\times n}$, there is no point in continuing. We will always get a system of the same form, and the controllability matrix must have rank equal to n. Although we have only examined single-input systems, it should be clear how this is extended to multi-input systems.

5.3 Output Controllability

Sometimes we are more interested in the ability to reach any possible output, rather than reaching any possible state. Even though a system is completely controllable, this does not mean we can force the *output* of the system to reach any possible output. We will define *output controllability* as follows:

A system is *output controllable* if it is possible to find a sequence of inputs $u(k)$ to take the output from any initial output $y(0)$ to any possible other output $y(n)$ in n steps, where $G \in \mathcal{R}^{n\times n}$. For the discrete-time system

$$\begin{aligned} x(k+1) &= Gx(k) + Hu(k), \quad G \in \mathcal{R}^{n\times n}, \quad H \in \mathcal{R}^{n\times r} \\ y(k) &= Cx(k) + Du(k), \quad C \in \mathcal{R}^{m\times n}. \quad D \in \mathcal{R}^{m\times r} \end{aligned}$$

the system is output controllable if the matrix

$$\left[\begin{array}{cccccc} CG^{n-1}H & CG^{n-2}H & \cdots & CGH & CH & D \end{array}\right]$$

has rank m. Note that the presence of the D matrix can only help with output controllability.

5.4 Observability

If a system is observable, then we will be able to determine the initial state based on knowledge of the input and measurement of the outputs. While this does not seem all that interesting, it also means that we will be able to construct devices, call *observers*, that allow us to estimate the states of a system based on knowledge of the inputs and measurements of the output. This is a very important idea, since it is sometimes very difficult and/or expensive to measure all of the states of a system, and state variable feedback assumes we have knowledge of all of the states. For our purposes, we will use the following definition of observability:

A system is *observable* if given:

- knowledge of the state variable system model
- a finite number of input signals $u(k)$
- measurement of the resulting outputs $y(k)$

the initial state $x(0)$ can be determined uniquely.

Let's initially look at a system with $G \in \mathcal{R}^{3\times 3}$ and a scalar input $u(k)$. Once we see this result, it is pretty easy to see how we would generalize the result. We have

$$\begin{aligned} y(0) &= Cx(0) \\ y(1) &= Cx(1) + Du(1) = C[Gx(0) + Hu(0)] + Du(1) = CGx(0) + [CHu(0) + Du(1)] \\ y(2) &= Cx(2) + Du(2) = C[Gx(1) + Hu(1)] + Du(2) = C[G\{Gx(0) + Hu(0)\} + Hu(1)] + Du(2) \\ &= CG^2x(0) + [CGHu(0) + CHu(1) + Du(2)] \end{aligned}$$

Since everything is known except $x(0)$, we can rewrite this system of equations as

$$\begin{bmatrix} y(0) \\ y(1) - [CHu(0) + Du(1)] \\ y(2) - [CGHu(0) + CHu(1) + Du(2)] \end{bmatrix} = \tilde{y} \;\; = \;\; \begin{bmatrix} C \\ CG \\ CG^2 \end{bmatrix} x(0)$$

For convenience we can rewrite this as

$$\begin{bmatrix} C \\ CG \\ CG^2 \end{bmatrix} x(0) \;\; = \;\; \begin{bmatrix} o_1 & o_2 & o_3 \end{bmatrix} \begin{bmatrix} x_1(0) \\ x_2(0) \\ x_3(0) \end{bmatrix} = \tilde{y}$$

where we have explicitly written the column vectors of the *observability matrix* as o_1, o_2 and o_3 and the components of the initial state, $x_1(0)$, $x_2(0)$, and $x_3(0)$. We then have

$$o_1 x_1(0) + o_2 x_2(0) + o_3 x_3(0) \;\; = \;\; \tilde{y}$$

The o_1, o_2, and o_3 are vectors, and the components of $x(0)$, $x_1(0)$, $x_2(0)$, and $x_3(0)$ are the expansion coefficients. We know that $\tilde{y}$ will have a *unique* representation if the vectors are linearly independent. Hence, in order to be able to determine $x(0)$ uniquely, we need n linearly independent columns in the controllability matrix.

This results can be generalized to the following test for observability:

If the observability matrix

$$\begin{bmatrix} C \\ CG \\ CG^2 \\ \vdots \\ CG^{n-1} \end{bmatrix}$$

has rank n, where $G \in \mathcal{R}^{\,n \times n}$, then the system is observable.

Note that

- $y(k)$ need not be a scalar
- $u(k)$ need not be a scalar
- we have used the Cayley-Hamilton Theorem again so we only need to look at the first n entries in the observability matrix

It should be pointed out that sometimes the observability matrix is written

$$\begin{bmatrix} C^T & G^T C^T & (G^2)^T C^T & \cdots & (G^{n-1})^T C^T \end{bmatrix}$$

6 State Variable Feedback

State variable feedback is a commonly used method in modern control systems. If the system is controllable, then we will be able to use state variable feedback to place the poles of the closed loop system anywhere we want. This is generally not the case for classical transfer function based methods. While we can use state-variable feedback to place the poles of a controllable system where ever we want, we cannot change the location of the system zeros. We can place the poles to try a pole/zero cancelation, but this is usually not desirable since the zeros of the system are just approximations based on our model.

We begin this chapter deriving the equations for state variable feedback, then utilize three different algorithms for determining the state variable feedback gain vector/matrix.

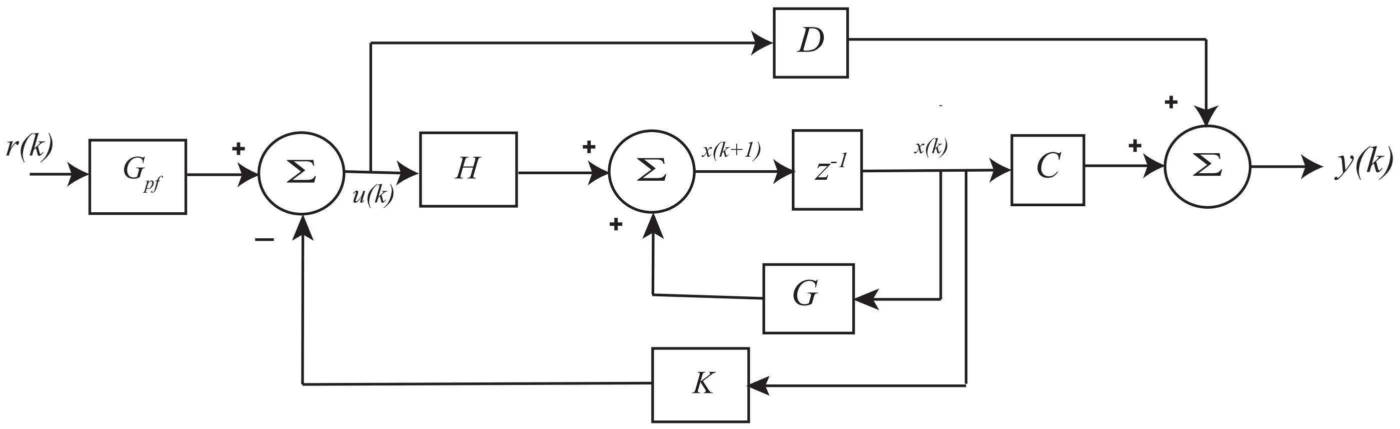

Figure 17: Basic discrete-time state variable feedback system. The feedback gain matrix is K, which is used to place the closed loop poles, and the prefilter gain G_{pf}, which is used to adjust the steady state error for a step input.

Consider the open loop system

$$\begin{aligned} x(k+1) &= Gx(k) + Hu(k) \\ y(k) &= Cx(k) + Du(k) \end{aligned}$$

Assume we have a state variable system as shown in Figure 17, where K is the feedback gain vector/matrix, G_{pf} is the prefilter gain, and $r(k)$ is the new reference input. Hence we have $u(k) = G_{pf}r(k) - Kx(k)$. Using this in our model of the plant we have

$$\begin{aligned} x(k+1) &= Gx(k) + H[G_{pf}r(k) - Kx(k)] = [G - HK]x(k) + [HG_{pf}]r(k) \\ y(k) &= Cx(k) + D[G_{pf}r(k) - Kx(k)] = [C - DK]x(k) + [DG_{pf}]r(k) \end{aligned}$$

The transfer function between the new input $r(k)$ and the output $y(k)$ can then be computed using z transforms as follows:

$$zX(z) = [G - HK]X(z) + [HG_{pf}]R(z)$$

$$\begin{aligned}
[zI - G + HK]\,X(z) &= [HG_{pf}]R(z) \\
X(z) &= [zI - G + HK]^{-1}[HG_{pf}]R(z) \\
Y(z) &= [C - DK]X(z) + [DG_{pf}]R(z) \\
Y(z) &= \Big\{[C - DK][zI - G + HK]^{-1}[HG_{pf}] + [DG_{pf}]\Big\}\,R(z)
\end{aligned}$$

so the transfer function/transfer matrix (if there are vector inputs) between the input $R(z)$ and the output $Y(z)$ is

$$F(z) \;=\; [C - DK][zI - G + HK]^{-1}[HG_{pf}] + [DG_{pf}]$$

Normally we will assume $D = 0$, and the transfer function reduces to

$$F(z) \;=\; C[zI - G + HK]^{-1}[HG_{pf}]$$

Let's assume we have a scalar input and $D = 0$. We know that $Y(z) = F(z)R(z)$, and if the input is a unit step $r(k) = u(k)$, then for a zero steady state error we use the *Final Value Theorem* to make the final final of $y(k) = 1$,

$$\begin{aligned}
\lim_{k\to\infty} y(k) = 1 &= \lim_{z\to 1} \frac{z-1}{z} Y(z) = \lim_{z\to 1} \frac{z-1}{z} F(z)R(z) \\
1 &= \lim_{z\to 1} \frac{z-1}{z} F(z) \frac{z}{z-1} = \lim_{z\to 1} F(z) = F(1) = C[I - G + HK]^{-1}[HG_{pf}]
\end{aligned}$$

or

$$G_{pf} \;=\; \frac{1}{C[I - G + HK]^{-1}H}$$

This prefilter will produce a steady state error of zero for a unit step input *provided* our system is accurately enough modeled. Note that we need to determine the feedback gain vector K *before* we try to determine the prefilter gain G_{pf}. Since the prefilter is outside of the feedback loop this is not a very robust method of trying to produce a steady state error of zero. A better method would be to make the system a type one system, which will be discussed in the next chapter.

6.1 Pole Placement by Transfer Functions

To determine the state feedback gain vector using the transfer function, we need to realize that the denominator of the transfer function will be determined by

$$\Delta(z) \;=\; \det[zI - G + HK]$$

We want the closed loop poles at $\mu_1, \mu_2, \ldots, \mu_n$, so the *desired* characteristic equations is

$$\begin{aligned}
\Delta(z) &= (z - \mu_1)(z - \mu_2)\ldots(z - \mu_n) \\
&= z^n + \alpha_1 z^{n-1} + \alpha_2 z^{n-2} + \ldots + \alpha_{n-1} z + \alpha_n
\end{aligned}$$

We then determine K by setting these equal,

$$\Delta(z) \;=\; \det[zI - G + HK] = z^n + \alpha_1 z^{n-1} + \alpha_2 z^{n-2} + \ldots + \alpha_{n-1} z + \alpha_n$$

This is a difficult computation for system with more than a few poles, so it is seldom used except for low order systems.

Example. Assume we have the system

$$G = \begin{bmatrix} 0.1 & 0.2 \\ 0 & 0.2 \end{bmatrix}, \quad H = \begin{bmatrix} 0 \\ 0.1 \end{bmatrix}$$

We first check to be sure the system is controllable,

$$\begin{bmatrix} GH & H \end{bmatrix} = \begin{bmatrix} 0.02 & 0 \\ 0.02 & 0.1 \end{bmatrix}$$

Clearly this matrix has rank two, so the system is controllable. Let's find the gain vector K to place the closed loop poles at -0.1 and -0.2. First we compute the desired characteristic polynomial

$$\Delta(z) = [z-(-0.1)][z-(-0.2)] = (z+0.1)(z+0.2) = z^2+0.3z+0.02$$

and then compute $\det[zI-G+HK]$,

$$\begin{aligned} zI-G+HK &= \begin{bmatrix} z & 0 \\ 0 & z \end{bmatrix} - \begin{bmatrix} 0.1 & 0.2 \\ 0 & 0.2 \end{bmatrix} + \begin{bmatrix} 0 \\ 0.1 \end{bmatrix} \begin{bmatrix} K_1 & K_2 \end{bmatrix} \\ &= \begin{bmatrix} z-0.1 & -0.2 \\ 0.1K_1 & z-0.2+0.1K_2 \end{bmatrix} \\ \det[zI-G+HK] &= (z-0.1)(z-0.2+0.1K_2)-(-0.2)(0.1K_1) \\ &= z^2+(-0.1-0.2+0.1K_2)z+(0.02-0.01K_2+0.02K_1) \end{aligned}$$

Equating terms with our desired characteristic polynomial, we get $K_2=6$ and then $K_1=3$, so

$$K = \begin{bmatrix} 3 & 6 \end{bmatrix}$$

Example. Assume we have the system

$$G = \begin{bmatrix} 0.1 & 0 \\ 0.2 & 0 \end{bmatrix}, \quad H = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

We first check to be sure the system is controllable,

$$\begin{bmatrix} GH & H \end{bmatrix} = \begin{bmatrix} 0.1 & 1.0 \\ 0.2 & 0 \end{bmatrix}$$

Clearly this matrix has rank two, so the system is controllable. We know from our previous examples that this system is controllable. Let's find the gain vector K to place the closed loop poles at 0.1 and -0.1. First we compute the desired characteristic polynomial

$$\Delta(z) = [z-(0.1)][z-(-0.1)] = (z-0.1)(z+0.1) = z^2-0.01$$

and then compute $\det[zI - G + HK]$,

$$\begin{aligned}
zI - G + HK &= \begin{bmatrix} z & 0 \\ 0 & z \end{bmatrix} - \begin{bmatrix} 0.1 & 0 \\ 0.1 & 0 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} \begin{bmatrix} K_1 & K_2 \end{bmatrix} \\
&= \begin{bmatrix} z - 0.1 + K_1 & K_2 \\ -0.2 & z \end{bmatrix} \\
\det[zI - G + HK] &= (z - 0.1 + K_1)(z) - (K_2)(-0.2) \\
&= z^2 + (K_1 - 0.1)z + K_2 0.2
\end{aligned}$$

Equating terms with our desired characteristic polynomial, we get $K_1 = 0.1$ and then $K_1 = -0.05$, so

$$K = \begin{bmatrix} 0.1 & -0.05 \end{bmatrix}$$

Example. Assume we again have the system

$$G = \begin{bmatrix} 0.1 & 0.2 \\ 0 & 0.2 \end{bmatrix}, \quad H = \begin{bmatrix} 0 \\ 0.1 \end{bmatrix}$$

Let's find the gain vector K to place both closed loop poles at 0. This clearly results in *deadbeat control.* First we compute the desired characteristic polynomial

$$\Delta(z) = (z - 0)(z - 0) = z^2$$

and then compute $\det[zI - G + HK]$,

$$\begin{aligned}
zI - G + HK &= \begin{bmatrix} z & 0 \\ 0 & z \end{bmatrix} - \begin{bmatrix} 0.1 & 0.2 \\ 0 & 0.2 \end{bmatrix} + \begin{bmatrix} 0 \\ 0.1 \end{bmatrix} \begin{bmatrix} K_1 & K_2 \end{bmatrix} \\
&= \begin{bmatrix} z - 0.1 & -0.2 \\ 0.1K_1 & z - 0.2 + 0.1K_2 \end{bmatrix} \\
\det[zI - G + HK] &= (z - 0.1)(z - 0.2 + 0.1K_2) - (-0.2)(0.1K_1) \\
&= z^2 + (-0.1 - 0.2 + 0.1K_2)z + (0.02 - 0.01K_2 + 0.02K_1)
\end{aligned}$$

Equating terms with our desired characteristic polynomial, we get $K_2 = 3$ and then $K_1 = 0.5$, so

$$K = \begin{bmatrix} 0.5 & 3 \end{bmatrix}$$

Our next algorithm, Ackermann's formula, is a bit easier to implement on a computer.

6.2 Pole Placement by Ackermann's Formula

Let's start with the open loop (plant) discrete-time state variable model

$$x(k+1) = Gx(k) + Hu(k)$$

where $u(k)$ is a *scalar* input. If we have state variable feedback of the form $u(k) = -Kx(k)$, then

$$\begin{aligned} x(k+1) &= Gx(k) + H\left(-Kx(k)\right) \\ &= \left(G - HK\right)x(k) \\ &= \tilde{G}x(k) \end{aligned}$$

where $\tilde{G} = G - HK$. Assume we want the closed loop poles at $\mu_1, \mu_2, \ldots, \mu_n$, so the *desired* characteristic equations is

$$\begin{aligned} \Delta(z) &= (z-\mu_1)(z-\mu_2)\ldots(z-\mu_n) \\ &= z^n + \alpha_1 z^{n-1} + \alpha_2 z^{n-2} + \ldots + \alpha_{n-1}z + \alpha_n \end{aligned}$$

Now we want the eigenvalues of $\tilde{G}$ to be the roots of this characteristic equation. Since the Cayley-Hamilton Theorem states that a matrix must satisfy its own characteristic polynomial, we know that once we have found the K to place the poles were we want them we will have

$$\Delta(\tilde{G}) = \tilde{G}^n + \alpha_1\tilde{G}^{n-1} + \alpha_2\tilde{G}^{n-2} + \ldots + \alpha_{n-1}\tilde{G} + \alpha_n I = 0$$

Let's assume for the sake of development that $n = 3$. We then have

$$\begin{aligned} \tilde{G} &= G - HK \\ \tilde{G}^2 &= \left(G-HK\right)^2 = \left(G-HK\right)\tilde{G} = G\tilde{G} - HK\tilde{G} = G(G-HK) - HK\tilde{G} = G^2 - GHK - HK\tilde{G} \\ \tilde{G}^3 &= (G-HK)\tilde{G}^2 = G\tilde{G}^2 - HK\tilde{G}^2 = G(G^2 - GHK - HK\tilde{G}) - HK\tilde{G}^2 \end{aligned}$$

We then have

$$\begin{aligned} 0 &= \Delta(\tilde{G}) \\ &= \tilde{G}^3 + \alpha_1\tilde{G}^2 + \alpha_2\tilde{G} + \alpha_3 I \\ &= \left[G^3 - G^2HK - GHK\tilde{G} - HK\tilde{G}^2\right] + \alpha_1\left[G^2 - GHK - HK\tilde{G}\right] + \alpha_2\left[G - HK\right] + \alpha_3 I \end{aligned}$$

Rearranging the terms we get

$$\Delta(\tilde{G}) = \left[G^3 + \alpha_1 G^2 + \alpha_2 G + \alpha_3 I\right] - H(\alpha_2 K + \alpha_1 K\tilde{G} + K\tilde{G}^2) - GH(\alpha_1 K + K\tilde{G}) - G^2H(K)$$

or

$$\Delta(\tilde{G}) = \Delta(G) - \left[\begin{array}{ccc} H & GH & G^2H \end{array}\right]\left[\begin{array}{c} \alpha_2 K + \alpha_1 K\tilde{G} + K\tilde{G}^2 \\ \alpha_1 K + K\tilde{G} \\ K \end{array}\right]$$

We clearly recognize the matrix

$$\left[\begin{array}{ccc} H & GH & G^2H \end{array}\right]$$

as the *controllability* matrix. Since we are assuming the system is controllable and the input is a scalar, we know this matrix will have an inverse. We also know that $\Delta(\tilde{G}) = 0$, so we can rearrange our above equations to be

$$\left[\begin{array}{ccc} H & GH & G^2H \end{array}\right]^{-1}\Delta(G) = \left[\begin{array}{c} \alpha_2 K + \alpha_1 K\tilde{G} + K\tilde{G}^2 \\ \alpha_1 K + K\tilde{G} \\ K \end{array}\right]$$

Now remember our goal is to find K, the state variable feedback vector. We can compute the controllability matrix since G and H are known, and we can compute $\Delta(G)$ since we know G and the desired characteristic polynomial. In order to find K, left's premultiply both sides of the above equation by the vector $\begin{bmatrix} 0 & 0 & 1 \end{bmatrix}$,

$$\begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} H & GH & G^2H \end{bmatrix}^{-1} \Delta(G) \quad = \quad \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \alpha_2 K + \alpha_1 K\tilde{G} + K\tilde{G}^2 \\ \alpha_1 K + K\tilde{G} \\ K \end{bmatrix} = K$$

Although we have only derived this formula for the case of $n = 3$, we can generalize this to the case of and arbitrary degree n,

$$K \quad = \quad \begin{bmatrix} 0 & \ldots & 0 & 1 \end{bmatrix} \begin{bmatrix} H & GH & \ldots & G^{n-1}H \end{bmatrix}^{-1} \Delta(G)$$

This form of Ackermann's formula is only valid for a system with a scalar input.

Example. Assume we have the system

$$G \quad = \quad \begin{bmatrix} 0.1 & 0.2 \\ 0 & 0.2 \end{bmatrix}, \quad H = \begin{bmatrix} 0 \\ 0.1 \end{bmatrix}$$

We know from our previous examples that this system is controllable. Let's again find the gain vector K to place the closed loop poles at -0.1 and -0.2 using Ackermann's formula. First we compute the desired characteristic polynomial

$$\Delta(z) \quad = \quad [z - (-0.1)][z - (-0.2)] = (z + 0.1)(z + 0.2) = z^2 + 0.3z + 0.02$$

and then $\Delta(G)$,

$$\Delta(G) \quad = \quad G^2 + 0.3G + 0.02I = \begin{bmatrix} 0.06 & 0.12 \\ 0 & 0.12 \end{bmatrix}$$

Next we need

$$\begin{bmatrix} H & GH \end{bmatrix} \quad = \quad \begin{bmatrix} 0 & 0.02 \\ 0.1 & 0.02 \end{bmatrix}$$

and then

$$\begin{bmatrix} H & GH \end{bmatrix}^{-1} \quad = \quad \begin{bmatrix} -10 & 10 \\ 50 & 0 \end{bmatrix}$$

Finally we have

$$\begin{aligned} K \quad &= \quad \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} -10 & 10 \\ 50 & 0 \end{bmatrix} \begin{bmatrix} 0.06 & 0.12 \\ 0 & 0.12 \end{bmatrix} \\ &= \quad \begin{bmatrix} 3 & 6 \end{bmatrix} \end{aligned}$$

Example. Assume we have the system

$$G = \begin{bmatrix} 0.1 & 0 \\ 0.2 & 0 \end{bmatrix}, \quad H = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

We know from our previous examples that this system is controllable. Let's again find the gain vector K to place the closed loop poles at 0.1 and -0.1 using Ackermann's formula. First we compute the desired characteristic polynomial

$$\Delta(z) = [z-(0.1)][z-(-0.1)] = (z-0.1)(z+0.1) = z^2 - 0.01$$

and then $\Delta(G)$,

$$\Delta(G) = G^2 - 0.01I = \begin{bmatrix} 0 & 0 \\ 0.02 & -0.01 \end{bmatrix}$$

Next we need

$$\begin{bmatrix} H & GH \end{bmatrix} = \begin{bmatrix} 1 & 0.1 \\ 0 & 0.2 \end{bmatrix}$$

and then

$$\begin{bmatrix} H & GH \end{bmatrix}^{-1} = \begin{bmatrix} 1 & -0.5 \\ 0 & 5 \end{bmatrix}$$

Finally we have

$$\begin{aligned} K &= \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & -0.5 \\ 0 & 5 \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 0.02 & -0.01 \end{bmatrix} \\ &= \begin{bmatrix} 0.1 & -0.05 \end{bmatrix} \end{aligned}$$

Example. Assume we again have the system

$$G = \begin{bmatrix} 0.1 & 0.2 \\ 0 & 0.2 \end{bmatrix}, \quad H = \begin{bmatrix} 0 \\ 0.1 \end{bmatrix}$$

We know from our previous examples that this system is controllable. Let's again find the gain vector K to place both closed loop poles at 0. This clearly results in *deadbeat control.* First we compute the desired characteristic polynomial

$$\Delta(z) = (z-0)(z+0) = z^2$$

and then $\Delta(G)$,

$$\Delta(G) = G^2 = \begin{bmatrix} 0.01 & 0.06 \\ 0 & 0.04 \end{bmatrix}$$

Next we need

$$\begin{bmatrix} H & GH \end{bmatrix} = \begin{bmatrix} 0 & 0.02 \\ 0.1 & 0.02 \end{bmatrix}$$

and then

$$\begin{bmatrix} H & GH \end{bmatrix}^{-1} = \begin{bmatrix} -10.0 & 10.0 \\ 50.0 & 0 \end{bmatrix}$$

Finally we have

$$\begin{aligned} K &= \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} -10 & 10 \\ 50 & 0 \end{bmatrix} \begin{bmatrix} 0.01 & 0.06 \\ 0 & 0.04 \end{bmatrix} \\ &= \begin{bmatrix} 0.5 & 3 \end{bmatrix} \end{aligned}$$

Ackermann's formula can be modified for a system with multiple inputs, however we will use an alternate method for systems with multiple inputs. Before we develop that method, we need to understand the idea of the null space of a matrix.

6.3 Null Space of a Matrix

The *null space* of a matrix A consists of all vectors x such that $Ax = 0$. If $A \in \mathcal{R}^{m \times n}$, and we assume that $n \geq m$, then the dimension of the null space, $q(A)$ is given by

$$q(A) = n - r(A)$$

Since the dimension of a space consists of the number of linear independent vectors in the space, this formula tell us how many linearly independent vectors we need to find in the null space of a matrix. If we think of the rows of the matrix A as vectors, then the dot product of each of these rows with any vector x in the null space must be zero. It is a common mistake to make the dot product of one or two rows equal to zero and believe that is good enough!

Example. Consider the matrix

$$A = \begin{bmatrix} 1 & 0 & 2 \\ 0 & 3 & 0 \end{bmatrix}$$

The rank of this matrix is clearly 2, so $r(A) = 2$. We also have $n = 3$. Hence $q(A) = n - r(A) = 3 - 2 = 1$, and there is only one linearly independent vector in the null space of the matrix. One such vector is

$$x = \begin{bmatrix} 2 \\ 0 \\ -1 \end{bmatrix}$$

Example. Consider the matrix

$$A = \begin{bmatrix} 1 & 0 & 1 & 2 \\ 0 & 1 & 1 & 2 \end{bmatrix}$$

For this matrix $n = 4$ and $r(A) = 2$, so we need to find two linearly independent vectors. Two linearly independent vectors in the null space of this matrix are

$$x_1 = \begin{bmatrix} 0 \\ 0 \\ -2 \\ 1 \end{bmatrix}, \quad x_2 = \begin{bmatrix} 1 \\ 1 \\ -1 \\ 0 \end{bmatrix}$$

One good way to be sure the vectors are linearly independent is to be sure one of the vectors has a zero in a component where the other vector is not zero as we have done here.

Example. Consider the matrix

$$A = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 2 & 1 & 1 \end{bmatrix}$$

For this matrix $n = 3$ and $r(A) = 2$, so $q(A) = n - r(A) = 1$. One linearly independent vector in the null space is

$$x = \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix}$$

6.4 Pole Placement by Direct Eigenvector/Eigenvalue Assignment

The poles of the open loop system are determined by the characteristic equation,

$$\Delta(\lambda) = \det[\lambda I - G] = 0$$

If we want to choose the state feedback matrix K to place a closed loop pole at λ_i, then we want

$$\Delta(\lambda_i) = \det[\lambda_i I - (G - HK)] = \det[\lambda_i I - G + HK] = 0$$

However, if $\Delta(\lambda_i) = 0$, then $[\lambda_i I - G + HK]$ does not have full rank. This means $q(A) \neq 0$ and there must be a vector ψ_i in the null space of this matrix. Hence

$$[\lambda_i I - G + HK]\psi_i = 0$$

Rearranging this equation we get

$$[G - HK]\psi_i = \lambda_i \psi_i$$

which means λ_i is an eigenvector of $G - HK$ and ψ_i is the associated eigenvector. We can also rearrange this equation as

$$(\lambda_i I - G)\psi_i + H(K\psi_i) = 0$$

or

$$\begin{bmatrix} \lambda_i I - G & \vdots & H \end{bmatrix} \begin{bmatrix} \psi_i \\ \cdots \\ K\psi_i \end{bmatrix} = 0$$

If we then define

$$\xi_i = \begin{bmatrix} \psi_i \\ \cdots \\ K\psi_i \end{bmatrix}$$

then we have

$$\begin{bmatrix} \lambda_i I - G & \vdots & H \end{bmatrix} \xi_i = 0$$

so that ξ_i is in the null space of $\begin{bmatrix} \lambda_i I - G & \vdots & H \end{bmatrix}$.

6.4.1 Pole Placement Algorithm (Scalar Input)

To determine the state feedback gain vector, go through the following steps:

1. For each closed loop pole (λ_i = desired closed loop pole location) solve

$$\left[\begin{array}{ccc} \lambda_i I - G & \vdots & H \end{array}\right] \xi_i = 0$$

2. After solving for the ξ_i, use the relationship

$$\xi_i = \left[\begin{array}{c} \psi_i \\ \cdots \\ K\psi_i \end{array}\right]$$

to determine the necessary relationships for determining K. It is important to remember that if $G \in \mathcal{R}^{\,n\times n}$ then $\psi_i \in \mathcal{R}^n$ (or $\psi_i \in \mathcal{C}^n$ if λ_i is complex).

3. For $u(k) \in \mathcal{R}^r$ there will generally be r linearly independent solutions ξ_i for each λ_i. Sometimes when there are repeated poles we can find more than one ξ_i for each λ_i, but this is not guaranteed.

4. If the input is a scalar ($r = 1$) then the gain vector K is unique.

5. If the system is controllable, we are guaranteed that this will work.

Example. Assume we have the system

$$G = \left[\begin{array}{cc} 0.1 & 0.2 \\ 0 & 0.2 \end{array}\right], \quad H = \left[\begin{array}{c} 0 \\ 0.1 \end{array}\right]$$

We know from our previous examples that this system is controllable. Since $u(k) \in \mathcal{R}^1$ we know there will be only one ξ_i for each λ_i. Let's again find the gain vector K to place the closed loop poles at -0.1 and -0.2.

For $\lambda_1 = -0.1$. We need to compute

$$\left[\begin{array}{ccc} \lambda_1 I - G & \vdots & H \end{array}\right] = \left[\begin{array}{ccc} -0.2 & -0.2 & 0 \\ 0 & -0.3 & 0.1 \end{array}\right]$$

Next we need to find a ξ_1 in the null space of this matrix. Clearly $\xi_1^T = \left[\begin{array}{ccc} 1 & -1 & -3 \end{array}\right]^T$ is one such vector.

For $\lambda_2 = -0.2$. we need to compute

$$\left[\begin{array}{ccc} \lambda_2 I - G & \vdots & H \end{array}\right] = \left[\begin{array}{ccc} -0.3 & -0.2 & 0 \\ 0 & -0.4 & 0.1 \end{array}\right]$$

Next we need to find a ξ_2 in the null space of this matrix. Clearly $\xi_1^T = \begin{bmatrix} -2 & 3 & 12 \end{bmatrix}^T$ is one such vector.

Now we need to use the relationship

$$\xi_i = \begin{bmatrix} \psi_i \\ K\psi_i \end{bmatrix}$$

Since in this case we know $n = 2$, we know that $\psi_i \in \mathcal{R}^2$. From ξ_1 we have

$$K \begin{bmatrix} 1 \\ -1 \end{bmatrix} = -3$$

and from ξ_2 we have

$$K \begin{bmatrix} -2 \\ 3 \end{bmatrix} = 12$$

Combining these we get

$$K \begin{bmatrix} 1 & -2 \\ -1 & 3 \end{bmatrix} = \begin{bmatrix} -3 & 12 \end{bmatrix}$$

which we can solve to get $K = \begin{bmatrix} 3 & 6 \end{bmatrix}$

Example. Assume we have the system

$$G = \begin{bmatrix} 0.1 & 0 \\ 0.2 & 0 \end{bmatrix}, \quad H = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

We know from our previous examples that this system is controllable. Since $u(k) \in \mathcal{R}^1$ we know there will be only one ξ_i for each λ_i. Let's again find the gain vector K to place the closed loop poles at 0.1 and -0.1.

For $\lambda_1 = 0.1$. We need to compute

$$\begin{bmatrix} \lambda_1 I - G & \vdots & H \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ -0.2 & 0.1 & 0 \end{bmatrix}$$

Next we need to find a ξ_1 in the null space of this matrix. Clearly $\xi_1^T = \begin{bmatrix} 1 & 2 & 0 \end{bmatrix}^T$ is one such vector.

For $\lambda_2 = -0.1$. we need to compute

$$\begin{bmatrix} \lambda_2 I - G & \vdots & H \end{bmatrix} = \begin{bmatrix} -0.2 & 0 & 1 \\ -0.2 & -0.1 & 0 \end{bmatrix}$$

Next we need to find a ξ_2 in the null space of this matrix. Clearly $\xi_1^T = \begin{bmatrix} 1 & -2 & 0.2 \end{bmatrix}^T$ is one such vector.

Now we need to use the relationship

$$\xi_i \;=\; \begin{bmatrix} \psi_i \\ K\psi_i \end{bmatrix}$$

Since in this case we know $n = 2$, we know that $\psi_i \in \mathcal{R}^2$. From ξ_1 we have

$$K\begin{bmatrix} 1 \\ 2 \end{bmatrix} \;=\; 0$$

and from ξ_2 we have

$$K\begin{bmatrix} 1 \\ -2 \end{bmatrix} \;=\; 0.2$$

Combining these we get

$$K\begin{bmatrix} 1 & 1 \\ 2 & -2 \end{bmatrix} \;=\; \begin{bmatrix} 0 & 0.2 \end{bmatrix}$$

which we can solve to get $K = \begin{bmatrix} 0.1 & -0.05 \end{bmatrix}$

Example. Assume we have the system

$$G \;=\; \begin{bmatrix} 0.1 & 0.2 \\ 0 & 0.2 \end{bmatrix}, \quad H = \begin{bmatrix} 0 \\ 0.1 \end{bmatrix}$$

We know from our previous examples that this system is controllable. Since $u(k) \in \mathcal{R}^1$ we know there will be only one ξ_i for each λ_i. Let's again find the gain vector K to place both closed loop poles at 0. This clearly results in *deadbeat control.*

For $\lambda = 0$*.* We need to compute

$$\begin{bmatrix} \lambda_I - G & \vdots & H \end{bmatrix} \;=\; \begin{bmatrix} -0.1 & -0.2 & 0 \\ 0 & -0.2 & 0.1 \end{bmatrix}$$

Next we need to find a ξ_1 in the null space of this matrix. Clearly $\xi_1^T = \begin{bmatrix} -2 & 1 & 2 \end{bmatrix}^T$ is one such vector.

There is only one vector in the null space. What do we do? We use what is called a *generalized eigenvector*[2]. We know that

$$\xi \;=\; \begin{bmatrix} \psi \\ K\psi \end{bmatrix}$$

and that

$$(G - HK)\psi \;=\; \lambda\psi$$

[2]This is only a brief introduction to the concept of generalized eigenvectors.

For a generalize eigenvector ψ_g we have

$$(G - HK)\psi_g \;=\; \lambda\psi_g + \psi$$

where ψ is a regular eigenvector (such as we have found already in this example). We can rewrite this as

$$[\lambda_i I - G + HK]\,\psi_g \;=\; -\psi$$

Since we don't really care about scaling on the eigenvector, we can also write this as

$$[\lambda_i I - G + HK]\,\psi_g \;=\; \psi$$

or

$$\left[\begin{array}{ccc} (\lambda I - G) & \vdots & H \end{array}\right]\left[\begin{array}{c} \psi_g \\ K\psi_g \end{array}\right] \;=\; \psi$$

or

$$\left[\begin{array}{ccc} (\lambda I - G) & \vdots & H \end{array}\right]\xi_g \;=\; \psi$$

In our case this becomes

$$\left[\begin{array}{ccc} -0.1 & -0.2 & 0 \\ 0 & -0.2 & 0.1 \end{array}\right]\xi_g \;=\; \left[\begin{array}{c} -2 \\ 1 \end{array}\right]$$

One solution is $\xi_g = \left[\begin{array}{ccc} 20 & 0 & 10 \end{array}\right]^T$. Next we use both ξ and ξ_g to get K. From ξ we have

$$K\left[\begin{array}{c} -2 \\ 1 \end{array}\right] \;=\; 2$$

and from ξ_g we have

$$K\left[\begin{array}{c} 20 \\ 0 \end{array}\right] \;=\; 10$$

Combining these we get

$$K\left[\begin{array}{cc} 20 & -2 \\ 0 & 1 \end{array}\right] \;=\; \left[\begin{array}{cc} 10 & 2 \end{array}\right]$$

from which we get $K = \left[\begin{array}{cc} 0.5 & 3 \end{array}\right]$.

6.4.2 Pole Placement Algorithm (Vector Input)

If we have a vector input, we need to modify our algorithm slightly, since as we will see, the feedback gain matrix is generally not unique. In what follows we will assume $u(k) \in \mathcal{R}^r$. To determine the state feedback gain matrix, go through the following steps:

1. For each closed loop pole (λ_i = desired closed loop pole location) solve

$$\left[\begin{array}{ccc} \lambda_i I - G & \vdots & H \end{array}\right] \xi = 0$$

. There will be r linearly independent vectors ξ. We can write these as

$$\left[\begin{array}{cccc} \xi_1 & \xi_2 & \ldots & \xi_r \end{array}\right] = \left[\begin{array}{cccc} \psi_1 & \psi_2 & \ldots & \psi_r \\ K\psi_1 & K\psi_2 & \ldots & K\psi_r \end{array}\right] = \left[\begin{array}{c} \Psi(\lambda_i) \\ \mathcal{F}(\lambda_i) \end{array}\right]$$

2. After solving for all the λ_i, we will have

$$K\left[\begin{array}{cccc} \Psi(\lambda_1) & \Psi(\lambda_2) & \ldots & \Psi(\lambda_n) \end{array}\right] = \left[\begin{array}{cccc} \mathcal{F}(\lambda_1) & \mathcal{F}(\lambda_2) & \ldots & \mathcal{F}(\lambda_n) \end{array}\right]$$

This is an *overdetermined* system of equations.

3. Determine n *linearly independent* columns from

$$\left[\begin{array}{cccc} \Psi(\lambda_1) & \Psi(\lambda_2) & \ldots & \Psi(\lambda_n) \end{array}\right]$$

Each column must be associated with a different eigenvalue, i.e., $g_1 \in \Psi(\lambda_1)$, $g_2 \in \Psi(\lambda_2)$, ..., $g_n \in \Psi(\lambda_n)$. Construct

$$G = \left[\begin{array}{cccc} g_1 & g_2 & \ldots & g_n \end{array}\right]$$

4. Find the corresponding columns from

$$\left[\begin{array}{cccc} \mathcal{F}(\lambda_1) & \mathcal{F}(\lambda_2) & \ldots & \mathcal{F}(\lambda_n) \end{array}\right]$$

$f_1 \in \mathcal{F}(\lambda_1)$, $f_2 \in \mathcal{F}(\lambda_2)$, etc. so $Kg_i = f_i$. Construct

$$F = \left[\begin{array}{cccc} f_1 & f_2 & \ldots & f_n \end{array}\right]$$

5. Solve $KG = F$. Note that there are multiple solutions since we can arbitrarily choose the columns g_i (as long as they are linearly independent). K is not unique!

Example. Determine the state feedback matrix K so the closed loop poles are at -0.2 and -0.3 for the system with

$$G = \left[\begin{array}{cc} 0.1 & 0.2 \\ 0.2 & 0.1 \end{array}\right] \qquad H = \left[\begin{array}{cc} 0.1 & 0.1 \\ 0 & 0.1 \end{array}\right]$$

For $\lambda_1 = -0.2$ we have

$$\left[\begin{array}{cc} \lambda_1 I - G & H \end{array}\right] = \left[\begin{array}{cccc} -0.3 & -0.2 & 0.1 & 0.1 \\ -0.2 & -0.3 & 0 & 0.1 \end{array}\right]$$

Two linearly independent vectors in the null space of this matrix are

$$\xi_1^a = \left[\begin{array}{c} 1 \\ 1 \\ 0 \\ 5 \end{array}\right] \qquad \xi_1^b = \left[\begin{array}{c} 0 \\ 1 \\ -1 \\ 3 \end{array}\right]$$

For $\lambda_1 = -0.3$ we have

$$\left[\begin{array}{cc} \lambda_1 I - G & H \end{array}\right] = \left[\begin{array}{cccc} -0.4 & -0.2 & 0.1 & 0.1 \\ -0.2 & -0.4 & 0 & 0.1 \end{array}\right]$$

Two linearly independent vectors in the null space of this matrix are

$$\xi_2^a = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 6 \end{bmatrix} \qquad \xi_2^b = \begin{bmatrix} 0 \\ 1 \\ -2 \\ 4 \end{bmatrix}$$

So we have

$$\Psi(\lambda_1) = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \qquad \mathcal{F}(\lambda_1) = \begin{bmatrix} 0 & -1 \\ 5 & 3 \end{bmatrix}$$

$$\Psi(\lambda_2) = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \qquad \mathcal{F}(\lambda_2) = \begin{bmatrix} 0 & -2 \\ 6 & 4 \end{bmatrix}$$

Now we choose $g_1 \in \Psi(\lambda_1)$ and the corresponding $f_1 \in \mathcal{F}(\lambda_1)$, $g_2 \in \Psi(\lambda_2)$ and the corresponding $f_2 \in \mathcal{F}(\lambda_2)$. In addition, g_1 and g_2 must be linearly independent.

Let's first assume we choose the first columns of $\Psi(\lambda_1)$ and the second column of $\Psi(\lambda_2)$. Then we have

$$g_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \; f_1 = \begin{bmatrix} 0 \\ 5 \end{bmatrix}, \; K\begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 5 \end{bmatrix}$$

$$g_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \; f_2 = \begin{bmatrix} -2 \\ 4 \end{bmatrix}, \; K\begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} -2 \\ 4 \end{bmatrix}$$

Combining we have

$$K\begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & -2 \\ 5 & 4 \end{bmatrix}$$

which yields

$$K = \begin{bmatrix} 2 & -2 \\ 1 & 4 \end{bmatrix}$$

Let's next assume we choose the second columns of $\Psi(\lambda_1)$ and the first column of $\Psi(\lambda_2)$. Then we have

$$g_1 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \; f_1 = \begin{bmatrix} -1 \\ 3 \end{bmatrix}, \; K\begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} -1 \\ 3 \end{bmatrix}$$

$$g_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \; f_2 = \begin{bmatrix} 0 \\ 6 \end{bmatrix}, \; K\begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 6 \end{bmatrix}$$

Combining we have

$$K\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 3 & 6 \end{bmatrix}$$

which yields

$$K = \begin{bmatrix} 1 & -1 \\ 3 & 3 \end{bmatrix}$$

We cannot choose any other combinations since we will not get linearly independent g_1 and g_2.

Example. Determine the state feedback matrix K so the both closed loop poles are at 0 for the system with

$$G = \begin{bmatrix} 0.1 & 0.2 \\ 0.2 & 0.1 \end{bmatrix} \qquad H = \begin{bmatrix} 0.1 & 0.1 \\ 0 & 0.1 \end{bmatrix}$$

For $\lambda = 0$ we have

$$\begin{bmatrix} \lambda I - G & H \end{bmatrix} = \begin{bmatrix} -0.1 & -0.2 & 0.1 & 0.1 \\ -0.2 & -0.1 & 0 & 0.1 \end{bmatrix}$$

Two linearly independent vectors in the null space of this matrix are

$$\xi_1^a = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 3 \end{bmatrix} \qquad \xi_1^b = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

Since we only have one pole location, these are the only vectors we need. Combining we have

$$K \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 3 & 1 \end{bmatrix}$$

which yields

$$K = \begin{bmatrix} -1 & 1 \\ 2 & 1 \end{bmatrix}$$

6.5 State Feedback Examples

Assume we have the discrete-time state variable model given by

$$\begin{bmatrix} x_1(k) \\ v_1(k) \\ x_2(k) \\ v_2(k) \end{bmatrix} = \begin{bmatrix} 0.4964 & 0.0391 & 0.5153 & 0.0095 \\ -15.5298 & 0.4361 & 14.7498 & 0.5022 \\ 0.4452 & 0.0082 & 0.3068 & 0.0358 \\ 12.7738 & 0.4326 & -21.0039 & 0.2572 \end{bmatrix} \begin{bmatrix} x_1(k) \\ v_1(k) \\ x_2(k) \\ v_2(k) \end{bmatrix} + \begin{bmatrix} 4.0181 \\ 142.8037 \\ 0.3955 \\ 29.8022 \end{bmatrix} u(k)$$

$$y(k) = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1(k) \\ v_1(k) \\ x_2(k) \\ v_2(k) \end{bmatrix}$$

The sample interval is $T_s = 0.05$ seconds. In this model our output is the first state (x_1). For this plant, the transfer function is

$$G_p(z) = \frac{4.018z^3 + 2.048z^2 + 1.939z + 3.65}{z^4 - 1.496z^3 + 1.492z^2 - 1.392z + 0.8639}$$

with the four poles at $-0.1073 \pm 0.9583j$ and $0.8555 \pm 0.4437j$. This system also has zeros at -0.9732 and $0.2317 \pm 0.9380j$.

The open loop unit step response of this system is shown in Figure 18. We can estimate the settling time using our previous formula as $T_s \approx \frac{-4T}{\ln(|p|)} = \frac{-4(0.05)}{\ln(|-0.1073+0.9583j|)} \approx 5.5$ seconds. This estimated settling time agrees with the results in the figure.

Figure 19 shows the unit step response using state variable feedback to place the closed loop poles at $0.1, 0.2, 0.3$ and 0.8. This produces a state feedback gain vector

$$K = \left[\begin{array}{cccc} -0.1165 & 0.0044 & 0.1280 & -0.0039 \end{array} \right]$$

and prefilter gain $G_{pf} = 0.0086$. We can estimate the settling time of this system as $T_s \approx \frac{-4(0.05)}{\ln(0.8)} \approx 0.9$ seconds, which agrees fairly well with the results shown in the figure.

Figure 20 shows the unit step response when the closed loop poles are located at $0.1, 0.2, 0.3$ and $-0.3 \pm 0.1j$. This produces a state feedback gain vector

$$K = \left[\begin{array}{cccc} 0.1179 & 0.0089 & -0.0796 & 0.0030 \end{array} \right]$$

and prefilter gain $G_{pf} = 0.1050$. We can estimate the settling time of this system as $T_s \approx \frac{-4(0.05)}{\ln(|-0.3+0.1j)} \approx 0.17$ seconds, which agrees fairly well with the results shown in the figure.

Figure 21 shows the unit step response when the closed loop poles are located at $0.1, 0.2, 0.3$ and $0.3 \pm 0.1j$. This produces a state feedback gain vector

$$K = \left[\begin{array}{cccc} -0.0468 & 0.0059 & 0.0566 & -0.0025 \end{array} \right]$$

and prefilter gain $G_{pf} = 0.0309$. We can estimate the settling time of this system as $T_s \approx \frac{-4(0.05)}{\ln(|0.3+0.1j)} \approx 0.17$ seconds. This settling time does not agree very will the the response of the system shown in the Figure. This is because our estimate of the settling time is based on a model having only poles and no zeros. In the previous examples the poles of the system were "far" from the zeros of the system, an had little effect on the step response of the system. In this example, the poles of the system are "near" the zeros of the system, and the zeros are affecting the step response of the system.

Figure 21 shows the unit step response when all the closed loop poles are located at 0. This is *deadbeat response* and produces a state feedback gain vector

$$K = \left[\begin{array}{cccc} 0.0765 & 0.0082 & -0.0463 & 0.0014 \end{array} \right]$$

and prefilter gain $G_{pf} = 0.0858$. Since this is a fourth order system, the settling time is four time steps. Since this is deadbeat response and the zeros of the system do not change, we can determine the system difference equation as follows:

$$\begin{aligned} \frac{Y(z)}{U(z)} &= (0.0858)\frac{4.018z^3 + 2.048z^2 + 1.939z + 3.65}{z^4} \\ &= 0.3447z^{-2} + 0.1758z^{-3} + 0.1663z^{-3} + 0.3132z^{-4} \end{aligned}$$

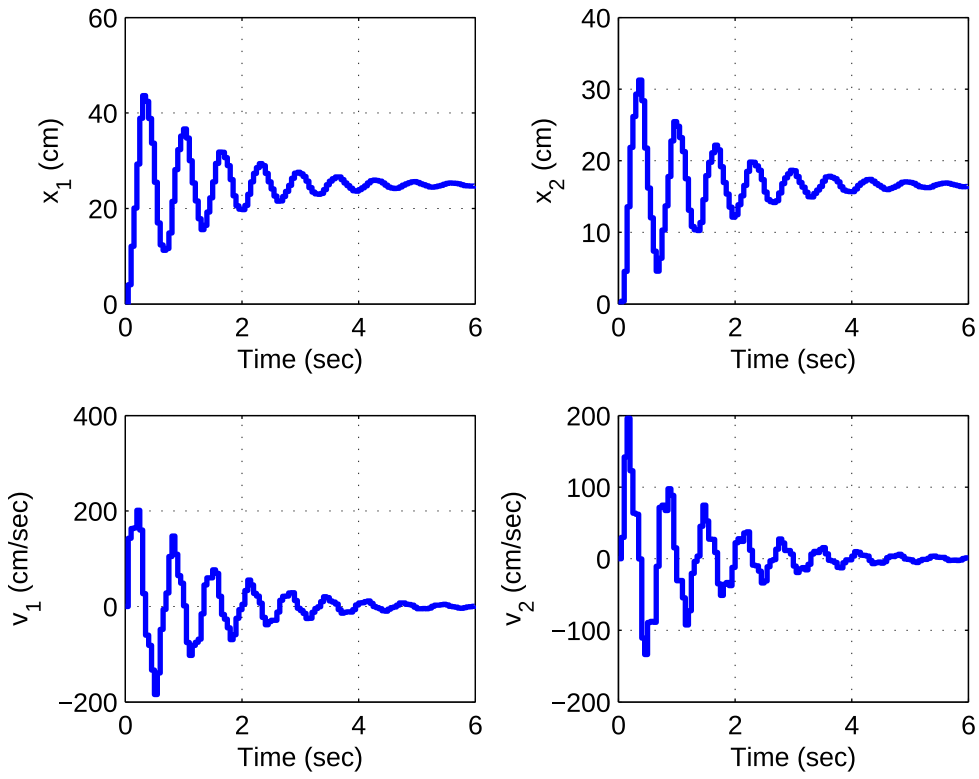

Figure 18: Open loop step response of fourth order system. The poles of the plant are at $-0.1073 \pm 0.9583j$ and $0.8555 \pm 0.4437j$. This system also has zeros at -0.9732 and $0.2317 \pm 0.9380j$.

or

$$y(k) = 0.3447u(k-1) + 0.1758u(k-2) + 0.1663u(k-3) + 03132u(k-4)$$

Since we assume the system is initially at rest with no initial conditions (all initial conditions are zero), we have

$$\begin{aligned}
y(0) &= 0 \\
y(1) &= 0.3447 \\
y(2) &= 0.3447 + 0.1758 = 0.5205 \\
y(3) &= 0.3447 + 0.1758 + 0.1663 = 0.6868 \\
y(4) &= 0.3447 + 0.1758 + 0.1663 + 0.3132 = 1.0000
\end{aligned}$$

All $y(k) = 1$ for $k \geq 4$. These results match those in the Figure.

6.6 General Guidelines for State Feedback Pole Locations

As a general rule the closer in magnitude the system poles are to the origin, the faster the response of the system. Just as with continuous time systems, if there are multiple poles, the

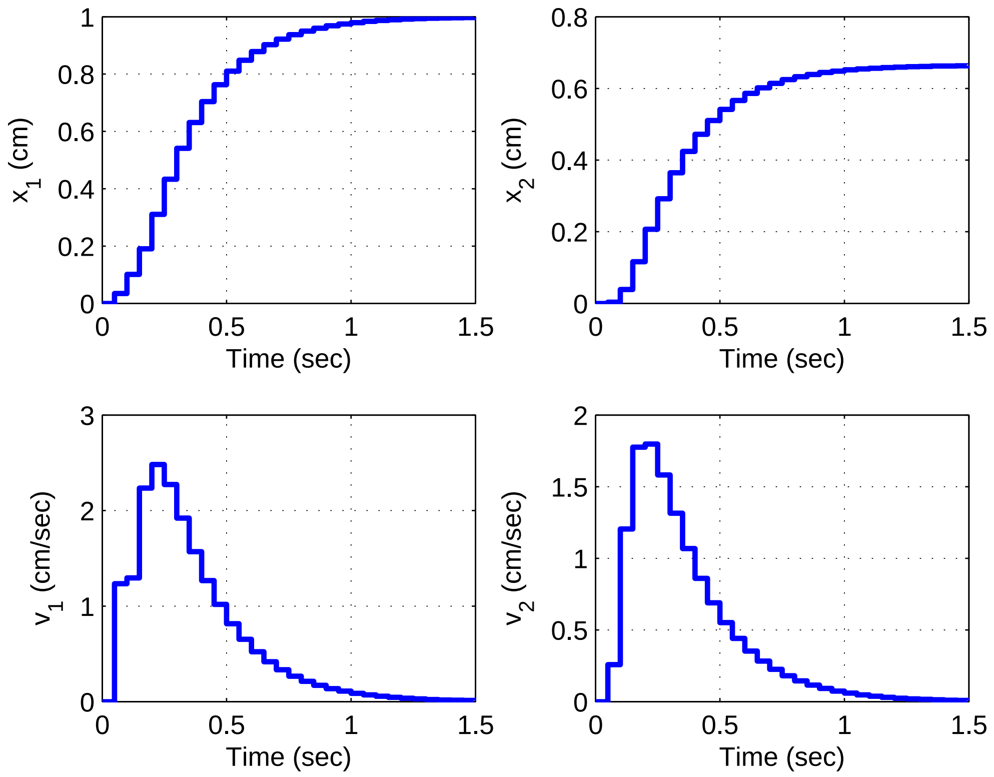

Figure 19: Closed loop step response of fourth order system with state variable feedback. The poles of the closed loop system are at $0.1, 0.20.3$ and 0.8. This system zeros remain at -0.9732 and $0.2317 \pm 0.9380j$.

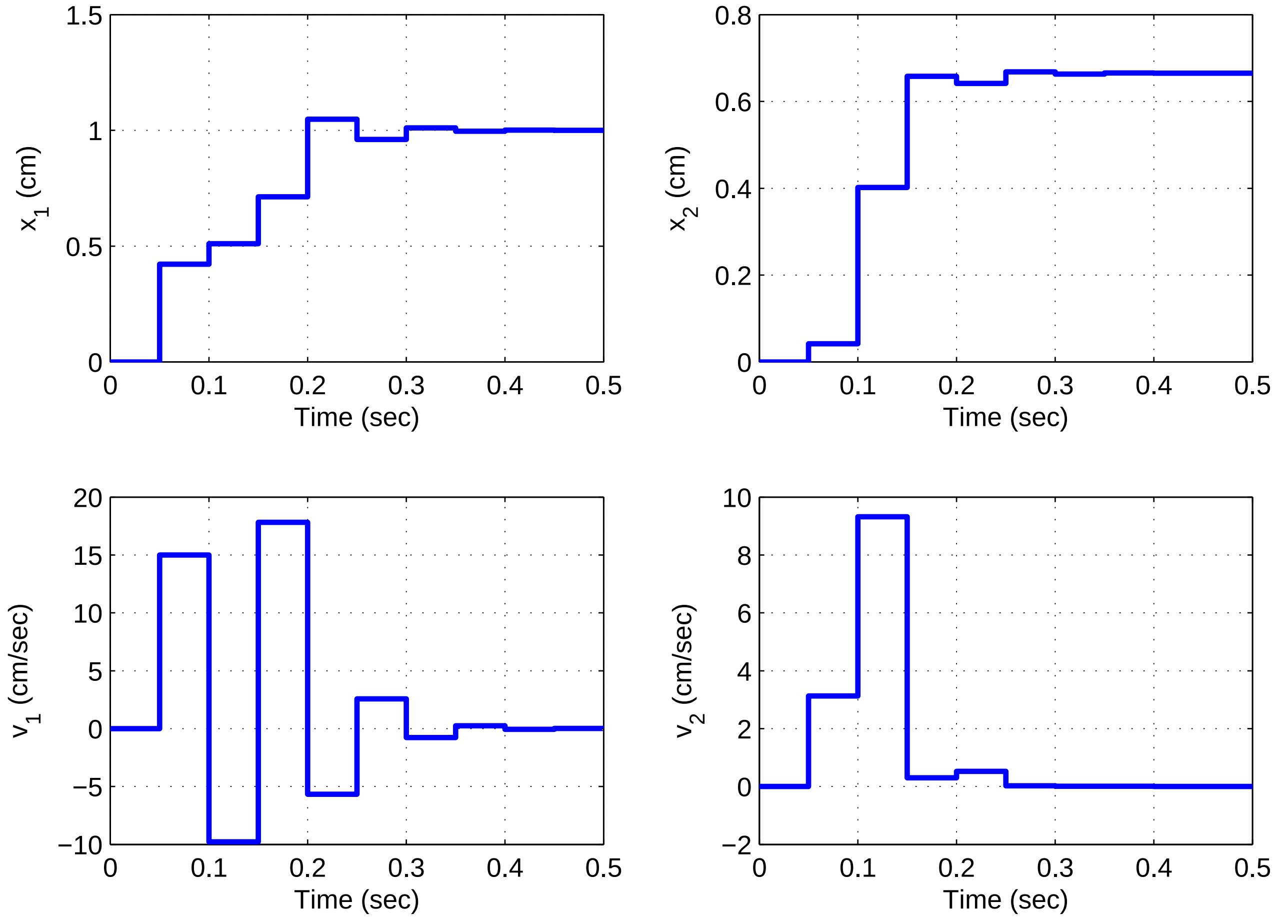

Figure 20: Closed loop step response of fourth order system with state variable feedback. The poles of the closed loop system are at $0.1, 0.2$ and $-0.3 \pm 0.1j$. This system zeros remain at -0.9732 and $0.2317 \pm 0.9380j$.

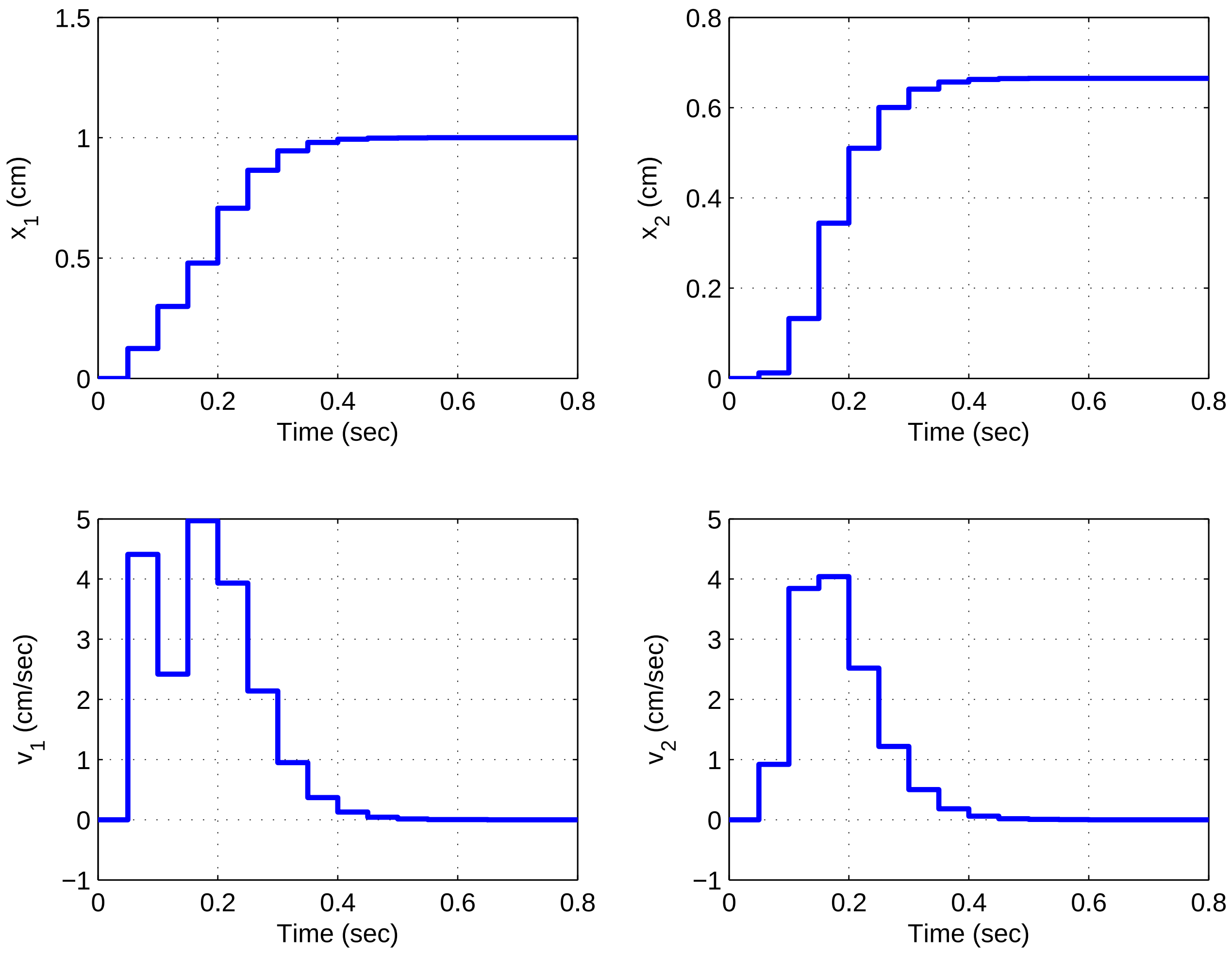

Figure 21: Closed loop step response of fourth order system with state variable feedback. The poles of the closed loop system are at $0.1, 0.2$ and $0.3 \pm 0.1j$. This system zeros remain at -0.9732 and $0.2317 \pm 0.9380j$.

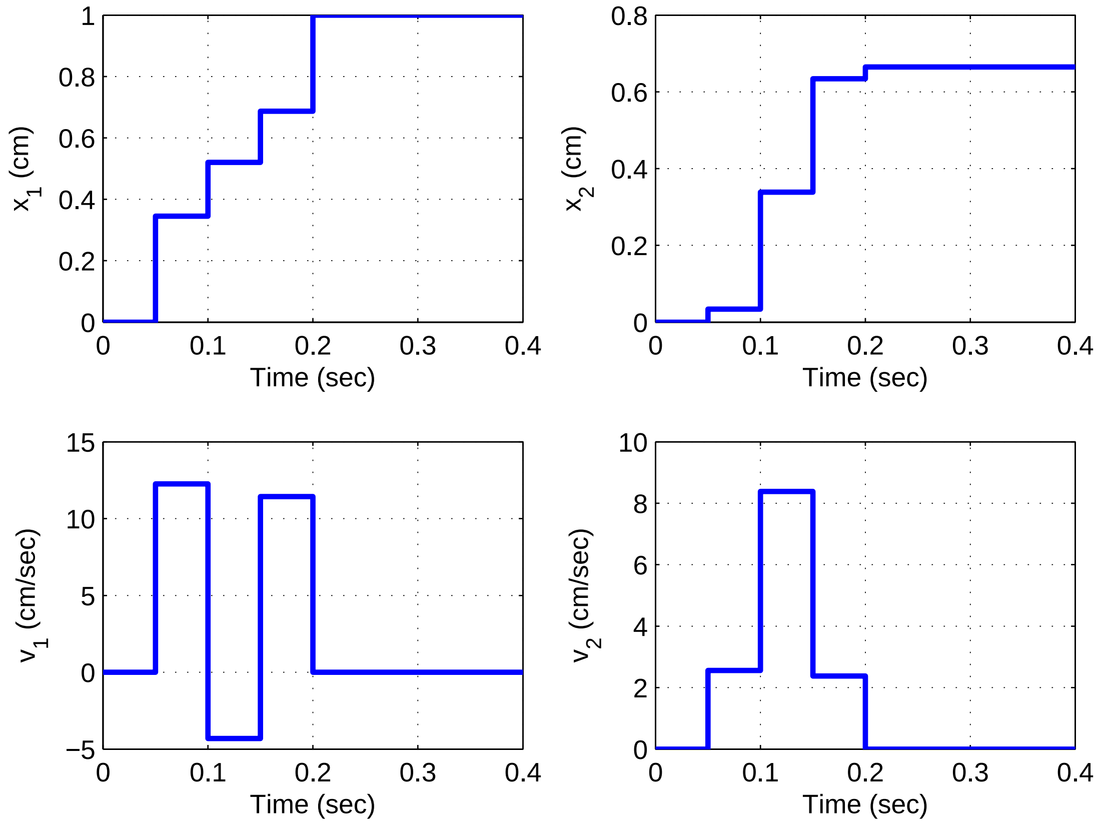

Figure 22: Closed loop step response of fourth order system with state variable feedback. The poles of the closed loop system are at 0. This is *deadbeat response*. This system zeros remain at -0.9732 and $0.2317 \pm 0.9380j$.

response of the *dominant* poles will dominate the response of the overall system. In this case, the poles with the largest magnitude will dominate the response and system will not reach steady state until the response of these poles have subsided. However, it should also be noted that placing the poles closer to the origin increases the system bandwidth and make the system less robust and more sensitive to noise, external disturbances, and errors in the mathematical model of the system. In addition, poles closer to the origin increases the control effort. Often, the designer needs to use some iterative "guess and check" to see if the system preforms adequately.

7 Integral Control

In the previous chapter we introduced state variable feedback for changing the poles of the closed loop system. With the use of a prefilter, we can often achieve a step response with zero steady state error. However, this assumes we have a very accurate model of the plant, which is often not the case. Consider the state variable model of a system we used in the last chapter,

$$\begin{aligned}
\begin{bmatrix} x_1(k) \\ v_1(k) \\ x_2(k) \\ v_2(k) \end{bmatrix} &= \begin{bmatrix} 0.4964 & 0.0391 & 0.5153 & 0.0095 \\ -15.5298 & 0.4361 & 14.7498 & 0.5022 \\ 0.4452 & 0.0082 & 0.3068 & 0.0358 \\ 12.7738 & 0.4326 & -21.0039 & 0.2572 \end{bmatrix} \begin{bmatrix} x_1(k) \\ v_1(k) \\ x_2(k) \\ v_2(k) \end{bmatrix} + \begin{bmatrix} 4.0181 \\ 142.8037 \\ 0.3955 \\ 29.8022 \end{bmatrix} u(k) \\
y(k) &= \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1(k) \\ v_1(k) \\ x_2(k) \\ v_2(k) \end{bmatrix}
\end{aligned}$$

The sample interval is $T_s = 0.05$ seconds. In this model our output is the third state (x_2). For this plant, the transfer function is

$$G_p(z) = \frac{4.018z^3 + 2.048z^2 + 1.939z + 3.65}{z^4 - 1.496z^3 + 1.492z^2 - 1.392z + 0.8639}$$

with the four poles at $-0.1073 \pm 0.9583j$ and $0.8555 \pm 0.4437j$. This system also has zeros at -0.9732 and $0.2317 \pm 0.9380j$. Figure 23 shows the unit step response using state variable feedback to place the closed loop poles at $0.1, 0.2, 0.3$ and 0.4, assuming the model accurately represents the plant. The state feedback gain vector is

$$K = \begin{bmatrix} -0.0607 & 0.0056 & 0.0701 & -0.0028 \end{bmatrix}$$

and prefilter gain $G_{pf} = 0.0390$. The figure shows that if our model is indeed accurate, we will pretty much get the response we expect.

Now let's assume that there are small errors in our model, and the correct model is given by

$$\begin{aligned}
\begin{bmatrix} x_1(k) \\ v_1(k) \\ x_2(k) \\ v_2(k) \end{bmatrix} &= \begin{bmatrix} 0.5460 & 0.0391 & 0.5153 & 0.0095 \\ -15.5298 & 0.3925 & 14.7498 & 0.5022 \\ 0.4452 & 0.0082 & 0.3374 & 0.0358 \\ 12.7738 & 0.4326 & -21.0039 & 0.2572 \end{bmatrix} \begin{bmatrix} x_1(k) \\ v_1(k) \\ x_2(k) \\ v_2(k) \end{bmatrix} + \begin{bmatrix} 4.0181 \\ 142.8037 \\ 0.3955 \\ 29.8022 \end{bmatrix} u(k) \\
y(k) &= \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1(k) \\ v_1(k) \\ x_2(k) \\ v_2(k) \end{bmatrix}
\end{aligned}$$

In this simplified example, the only modeling errors are on the diagonal. We, of course, do not know the correct system, and if we use our model to determine the state feedback gains and the prefilter gain G_{pf}, we will get the unit step response shown in Figure 24. Clearly the steady state error is no longer correct, since our prefilter gain is not correct. However, since we have an inaccurate model, there is really nothing we can do except try to "guess and check" the prefilter gain until we get something that works better. However, this will only work if our plant does not change over time.

An alternative approach is to include *integral control* so we can obtain zero steady state error in one state for a unit step input. This is very similar to what is done using transfer functions. The idea is to remove the prefilter gain (make it unity), which is outside the feedback loop, and include an integrator, which is inside the feedback loop. We will assume a configuration like that shown in Figure 25. Note that we have introduced C_y as the 'C' vector we use to choose the state for our output. This notation will be more important in the next chapter when we introduce observers. Using this approach, with $K_1 = 0.0195$ and

$$K_2 = \left[\begin{array}{cccc} 0.0053 & 0.0068 & 0.0282 & -0.0003 \end{array} \right]$$

the closed loop poles will be at $0.1, 0.2, 0.30.4$ and 0.5. The step response of this new system with the integrator is shown in Figure 26. We will spend the remainder of this chapter determining how to find the gains K_1 and K_2 for this new structure. Our derivation will be based on **Ogata**. It should be pointed out if the error in modeling the plant is too large, our controller may produce an unstable system.

Consider the closed loop system in Figure 25, which includes an integrator as well as state variable feedback. This configuration allows for zero steady state error for a single state. In this configuration we introduce C_y as the vector that multiplies the output y to get the single state we want to have a zero steady state error. We have the state equations

$$\begin{aligned} x(k+1) &= Gx(k) + Hu(k) \;\; x(k) \in \mathcal{R}^n, \; G \in \mathcal{R}^{n\times n}, \; H \in \mathcal{R}^{n\times m}, \; u(k) \in \mathcal{R}^m \\ y(k) &= C_y x(k), \; C_y \in \mathcal{R}^{1\times n} \end{aligned}$$

We also have

$$v(k) = r(k) - y(k) + v(k-1)$$

or

$$v(k) - v(k-1) = r(k) - y(k)$$

Taking z-transforms we have

$$V(z)\left(1 - z^{-1}\right) = R(z) - Y(z)$$

or

$$\frac{V(z)}{R(z) - Y(z)} = \frac{1}{1 - z^{-1}}$$

Recall that the transfer function corresponding to an integrator in continuous time, $\frac{1}{s}$ corresponds to the transfer function $\frac{1}{1-z^{-1}}$ in discrete-time. Hence this configuration functions as an integrator. As $k \rightarrow \infty$ we have

$$v(\infty) = r - y(\infty) + v(\infty)$$

or

$$lim_{k\rightarrow\infty} y(k) = r$$

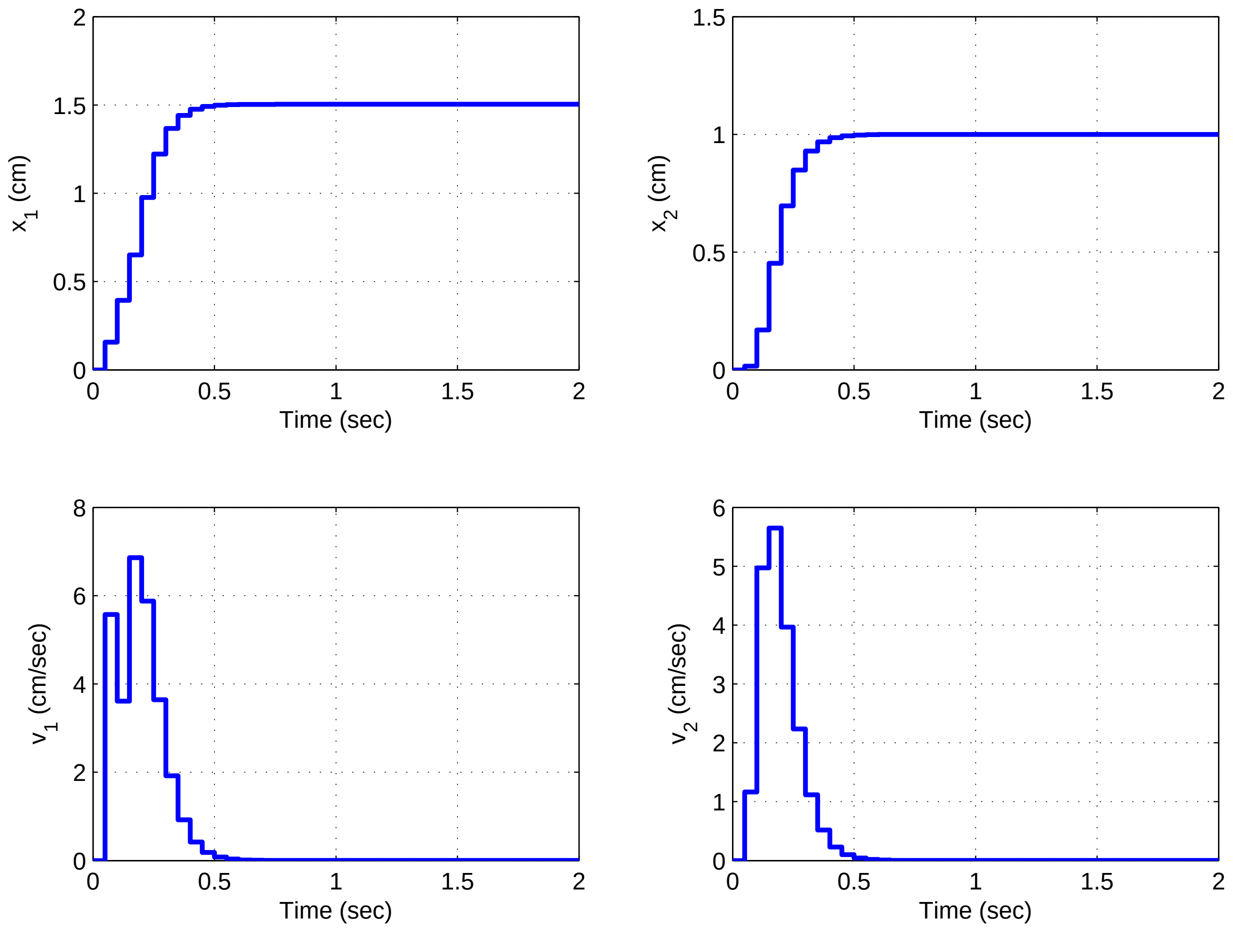

Figure 23: State variable feedback with poles at 0.1, 0.2, 0.3 and 0.4. We are trying to control the position of the third state (x_2). This example assumes our model accurately represents the plant. We clearly have a zero steady state error for a unit step input.

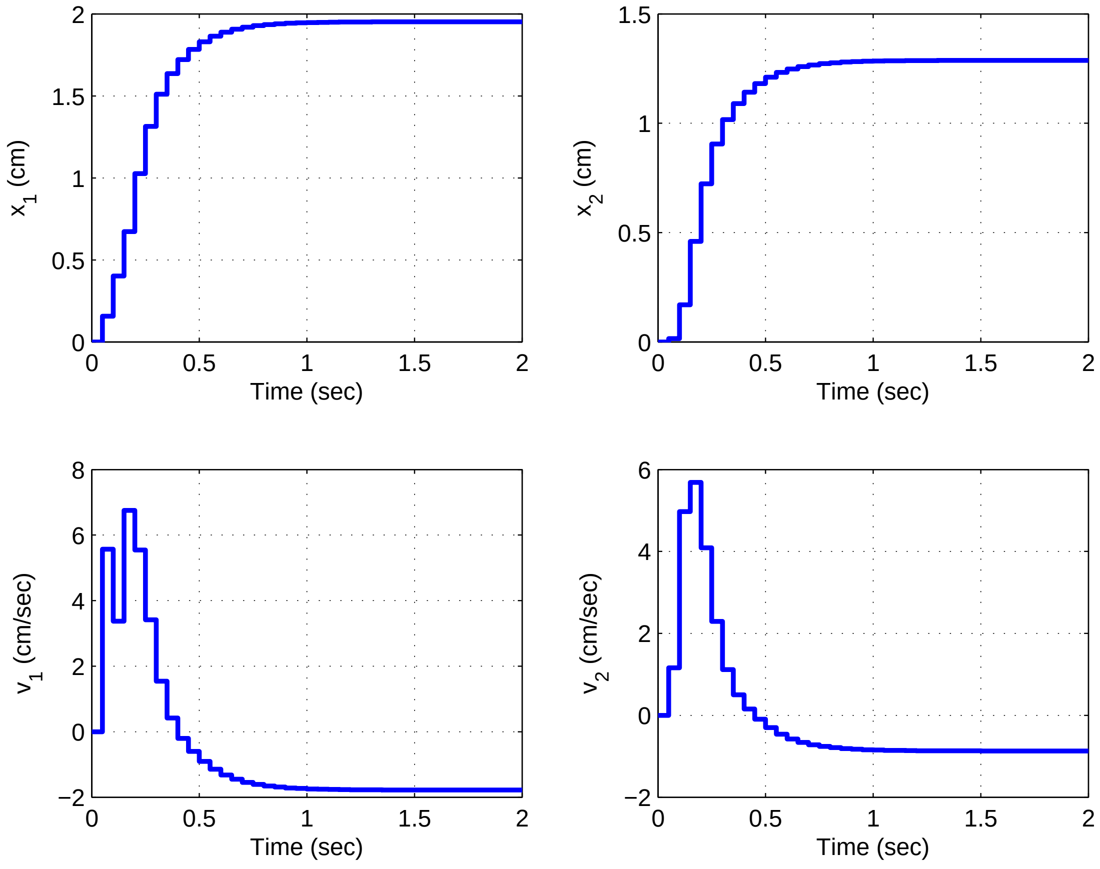

Figure 24: State variable feedback with poles at 0.1, 0.2, 0.3 and 0.4. We are trying to control the position of the third state (x_2). This example assumes our model does not accurately represents the plant. We clearly do not have a zero steady state error for a unit step.

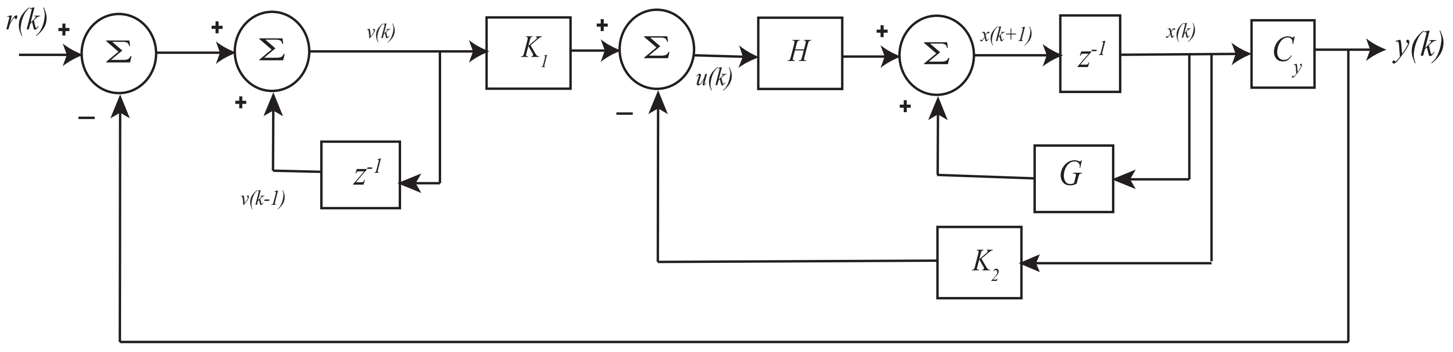

Figure 25: Basic configuration for including integral control in a state variable feedback system.

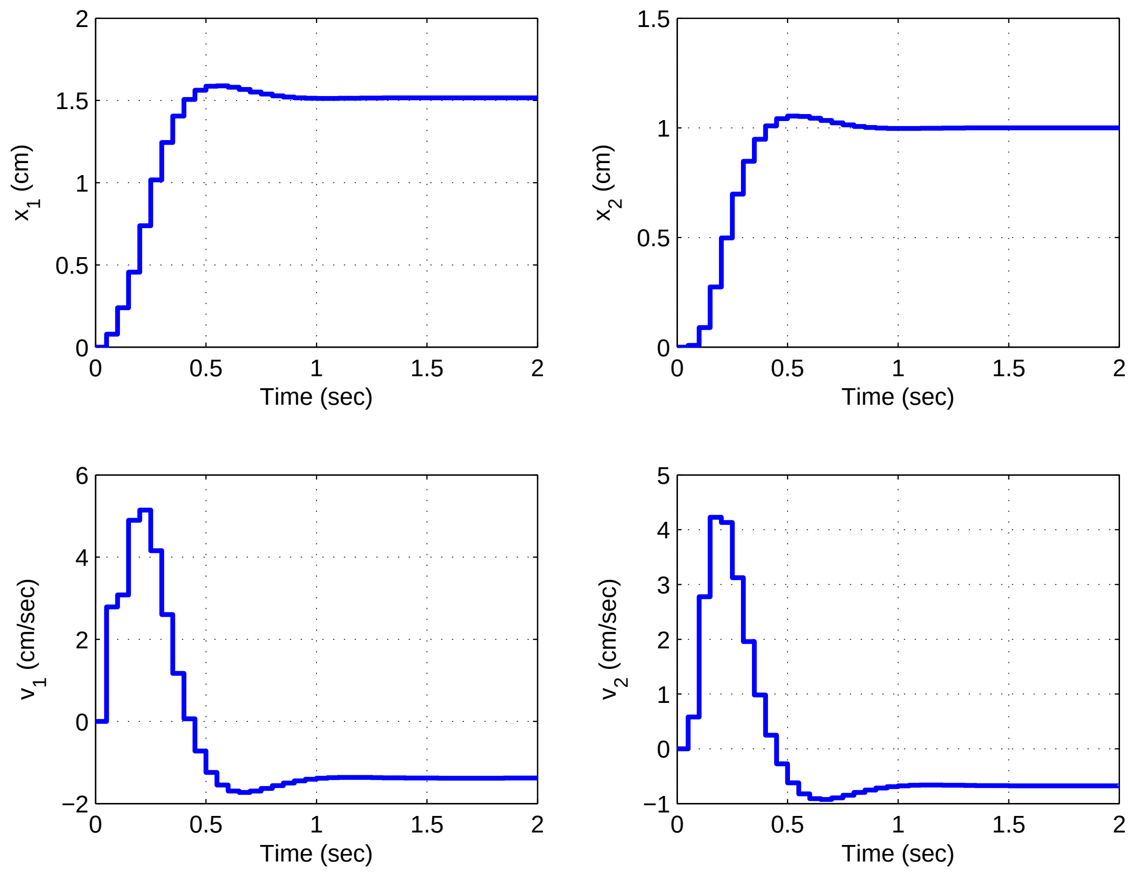

Figure 26: State variable feedback with integral control with poles at 0.1, 0.2, 0.3, 0.4, and 0.5. We are trying to control the position of the third state (x_2). This example assumes our model does not accurately represents the plant. We again have a zero steady state error for a unit step.

which tells us the steady state value of $y(k)$ is the amplitude of the reference input r, and there is no steady state error for a unit step input. Next, we need to determine how to choose K_1 and K_2.

We start by rewriting the integrator equation as

$$v(k) = v(k-1) + r(k) - y(k)$$

or

$$\begin{aligned} v(k+1) &= v(k) + r(k+1) - y(k+1) \\ &= v(k) + r(k+1) - C_y \left[Gx(k) + Hu(k)\right] \\ &= -C_yGx(k) + v(k) - C_yHu(k) + r(k+1) \end{aligned}$$

Next, we write the control vector $u(k)$ as

$$u(k) = -K_2x(k) + K_1v(k)$$

or

$$\begin{aligned} u(k+1) &= -K_2x(k+1) + K_1v(k+1) \\ &= -K_2\left[Gx(k) + Hu(k)\right] + K_1\left[-C_yGx(k) + v(k) - C_yHu(k) + r(k+1)\right] \\ &= \left[-K_2G - K_1C_yG\right]x(k) + \left[-K_2H - K_1C_yH\right]u(k) + K_1v(k) + K_1r(k+1) \\ &= \left[-K_2G - K_1C_yG\right]x(k) + \left[-K_2H - K_1C_yH\right]u(k) + \left[u(k) + K_2x(k)\right] + K_1r(k+1) \\ &= \left[K_2 - K_2G - K_1C_yG\right]x(k) + \left[I_m - K_2H - K_1C_yH\right]u(k) + K_1r(k+1) \end{aligned}$$

From this we can write the state variable model

$$\begin{bmatrix} x(k+1) \\ u(k+1) \end{bmatrix} = \begin{bmatrix} G & H \\ K_2 - K_2G - K_1C_yG & I_m - K_2H - K_1C_yH \end{bmatrix} \begin{bmatrix} x(k) \\ u(k) \end{bmatrix} + \begin{bmatrix} 0 \\ K_1 \end{bmatrix} r(k+1)$$

with the output equation

$$y(k) = \begin{bmatrix} C_y & 0 \end{bmatrix} \begin{bmatrix} x(k) \\ u(k) \end{bmatrix}$$

Now consider the case when the reference input $r(k) = ru(k)$, a step with amplitude r. We then have

$$\begin{bmatrix} x(k+1) \\ u(k+1) \end{bmatrix} = \begin{bmatrix} G & H \\ K_2 - K_2G - K_1C_yG & I_m - K_2H - K_1C_yH \end{bmatrix} \begin{bmatrix} x(k) \\ u(k) \end{bmatrix} + \begin{bmatrix} 0 \\ K_1r \end{bmatrix}$$

As $k \rightarrow \infty$ we have

$$\begin{bmatrix} x(\infty) \\ u(\infty) \end{bmatrix} = \begin{bmatrix} G & H \\ K_2 - K_2G - K_1C_yG & I_m - K_2H - K_1C_yH \end{bmatrix} \begin{bmatrix} x(\infty) \\ u(\infty) \end{bmatrix} + \begin{bmatrix} 0 \\ K_1r \end{bmatrix}$$

Now define the error vectors by

$$\begin{aligned} x_e(k) &= x(k) - x(\infty) \\ u_e(k) &= u(k) - u(\infty) \end{aligned}$$

Subtracting our previous expressions we have

$$\begin{bmatrix} x(k+1)-x(\infty) \\ u(k+1)-u(\infty) \end{bmatrix} = \begin{bmatrix} G & H \\ K_2 - K_2G - K_1C_yG & I_m - K_2H - K_1C_yH \end{bmatrix} \begin{bmatrix} x(k)-x(\infty) \\ u(k)-u(\infty) \end{bmatrix}$$

or

$$\begin{bmatrix} x_e(k+1) \\ u_e(k+1) \end{bmatrix} = \begin{bmatrix} G & H \\ K_2 - K_2G - K_1C_yG & I_m - K_2H - K_1C_yH \end{bmatrix} \begin{bmatrix} x_e(k) \\ u_e(k) \end{bmatrix}$$

We can rewrite this system as

$$\begin{aligned} \begin{bmatrix} x_e(k+1) \\ u_e(k+1) \end{bmatrix} &= \begin{bmatrix} G & H \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_e(k) \\ u_e(k) \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ K_2 - K_2G - K_1C_yG & I_m - K_2H - K_1C_yH \end{bmatrix} \begin{bmatrix} x_e(k) \\ u_e(k) \end{bmatrix} \\ &= \begin{bmatrix} G & H \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_e(k) \\ u_e(k) \end{bmatrix} + \begin{bmatrix} 0 \\ I_m \end{bmatrix} w(k) \end{aligned}$$

where

$$w(k) = \begin{bmatrix} K_2 - K_2G - K_1C_yG & I_m - K_2H - K_1C_yH \end{bmatrix} \begin{bmatrix} x_e(k) \\ u_e(k) \end{bmatrix}$$

Now define

$$\begin{aligned} \xi(k) &= \begin{bmatrix} x_e(k) \\ u_e(k) \end{bmatrix} \\ \hat{G} &= \begin{bmatrix} G & H \\ 0 & 0 \end{bmatrix} \\ \hat{H} &= \begin{bmatrix} 0 \\ I_m \end{bmatrix} \\ \hat{K} &= -\begin{bmatrix} K_2 - K_2G - K_1C_yG & I_m - K_2H - K_1C_yH \end{bmatrix} \end{aligned}$$

The state equations then become

$$\begin{aligned} \xi(k+1) &= \hat{G}\xi(k) + \hat{H}w(k) \\ w(k) &= -\hat{K}\xi(k) \end{aligned}$$

or

$$\begin{aligned} \xi(k+1) &= \hat{G}\xi(k) + \hat{H}\left(-\hat{K}\right)\xi(k) \\ &= \left(\hat{G} - \hat{H}\hat{K}\right)\xi(k) \end{aligned}$$

This is the equation used to replace the poles of $\hat{G}$ using state variable feedback! Thus $\hat{K}$ can be determined by using techniques we've learned before.

Now assume we have determined $\hat{K}$, how do we get K_1 and K_2? First note

$$\begin{bmatrix} K_2 & K_1 \end{bmatrix} \begin{bmatrix} G - I_n & H \\ C_yG & C_yH \end{bmatrix} = \begin{bmatrix} K_2(G-I_n) + K_1C_yG & K_2H + K_1C_yH \end{bmatrix}$$

and

$$\begin{bmatrix} K_1 & K_2 \end{bmatrix} \begin{bmatrix} G - I_n & H \\ C_yG & C_yH \end{bmatrix} - \begin{bmatrix} 0 & I_m \end{bmatrix} = \begin{bmatrix} K_2G - K_2 + K_1C_yG & -I_m + K_2H + K_1C_yH \end{bmatrix} = \hat{K}$$

Hence

$$\hat{K} + \begin{bmatrix} 0 & I_m \end{bmatrix} = \begin{bmatrix} K_2 & K_1 \end{bmatrix} \begin{bmatrix} G - I_n & H \\ C_yG & C_yH \end{bmatrix}$$

or

$$\left\{ \hat{K} + \begin{bmatrix} 0 & I_m \end{bmatrix} \right\} \begin{bmatrix} G - I_n & H \\ C_yG & C_yH \end{bmatrix}^{-1} = \begin{bmatrix} K_2 & K_1 \end{bmatrix}$$

Before we do an example, we need to make a few notes:

- If the system is stable, we are guaranteed that the steady state error in the state of our choice will be zero **eventually**. However, it may take quite a while for this to be achieved. Remember, we are only looking at the steady state behavior.
- If the plant is controllable, including the integrator does not change the controllability.
- If there are more than one inputs ($m > 1$), then K_1 and K_2 are not unique. Choose the pair that seem to work best.
- This method will only track a step input in one state with zero steady state error. Other options are possible, but they are beyond the scope of these notes.

Example. Let's assume

$$G = \begin{bmatrix} 0.1 & 0 \\ 0.2 & 0 \end{bmatrix}, \; H = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \; C_y = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

and we want *deadbeat control* and a system with an integrator. First we need to form $\hat{G}$ and $\hat{H}$,

$$\begin{aligned} \hat{G} &= \begin{bmatrix} G & H \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 0.1 & 0 & 1 \\ 0.2 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \\ \hat{H} &= \begin{bmatrix} 0 \\ I_m \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \end{aligned}$$

Placing the poles we determine that

$$\hat{K} = \begin{bmatrix} 0.01 & 0 & 0.1 \end{bmatrix}$$

We can then form

$$KK = \hat{K} + \begin{bmatrix} 0 & I_m \end{bmatrix} = \begin{bmatrix} 0.01 & 0 & 0.1 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0.01 & 0 & 1.1 \end{bmatrix}$$

and

$$TT = \begin{bmatrix} G - I_n & H \\ C_yG & C_yH \end{bmatrix} = \begin{bmatrix} -0.9 & 0 & 1 \\ 0.2 & -1 & 0 \\ 0.1 & 0 & 1 \end{bmatrix}$$

Then

$$\begin{bmatrix} K_2 & K_1 \end{bmatrix} = KK \times TT^{-1} = \begin{bmatrix} 0.1 & 0 & 1 \end{bmatrix}$$

So $K_2 = \begin{bmatrix} 0.1 & 0 \end{bmatrix}$ and $K_1 = 1$. Note that K_2 is the *beginning* of the $\hat{K}$ array and that K_1 is at the *end* of the $\hat{K}$ array.

8 Full Order State Observers

In order to use state variable feedback, we need to have access to the states. Sometimes we cannot measure all of the states, or it is too expensive. Instead of using the actual states we will use estimates of the states. An observer constructs estimates using knowledge of the system dynamics, the (known) input, and the (measured) output. A *full order observer* is used to estimate *all* of the states. A *minimum order observer*, which will be covered in the next chapter, is used to estimate only some of the states. In order to estimates the states the system must be *observable.*

8.1 Derivation of Observer

Assume we have the system modeled by

$$\begin{aligned} x(k+1) &= Gx(k) + Hu(k) \\ y(k) &= Cx(k) \end{aligned}$$

We will assume our estimated state, $\tilde{x}(k)$ follows the same dynamical model, but we will include a term to correct for differences between the real (measured) output of the system and the estimated state of the system,

$$\tilde{x}(k+1) = G\tilde{x}(k) + Hu(k) + K_e\left[y(k) - \tilde{y}(k)\right]$$

noindent where

$$\begin{aligned} \tilde{x}(k) &= \text{the estimated state} \\ \tilde{y}(k) &= \text{the output based on the estimated state,}\, \tilde{y}(k) = C\tilde{x}(k) \\ K_e &= \text{observer gain matrix} \end{aligned}$$

A signal flow model of the original system (plant) using a full order observer to estimate the states used in state variable is shown in Figure 27. As the figure illustrated, when using a full order observer with state variable feedback, it is the estimated state that is used, i.e.,

$$u(k) = G_{pf}r(k) - K\tilde{x}(k)$$

What remains is to determine K_e.

Let's define the error signal $e(k)$ to be the difference between the true state, $x(k)$, and the estimate of the true state, $\tilde{x}(k)$,

$$e(k) = x(k) - \tilde{x}(k)$$

Then we have

$$\begin{aligned} e(k+1) &= x(k+1) - \tilde{x}(k+1) \\ &= [Gx(k) + Hu(k)] - [G\tilde{x}(k) + Hu(k) + K_e\{y(k) - \tilde{y}(k)\}] \\ &= G\left(x(k) - \tilde{x}(k)\right) - K_e\{y(k) - \tilde{y}(k)\} \\ &= G\left(x(k) - \tilde{x}(k)\right) - K_e\{Cx(k) - C\tilde{x}(k)\} \\ &= G\left(x(k) - \tilde{x}(k)\right) - K_eC\left(x(k) - \tilde{x}(k)\right) \\ &= \left(G - K_eC\right)\left(x(k) - \tilde{x}(k)\right) \\ &= \left(G - K_eC\right)e(k) \end{aligned}$$

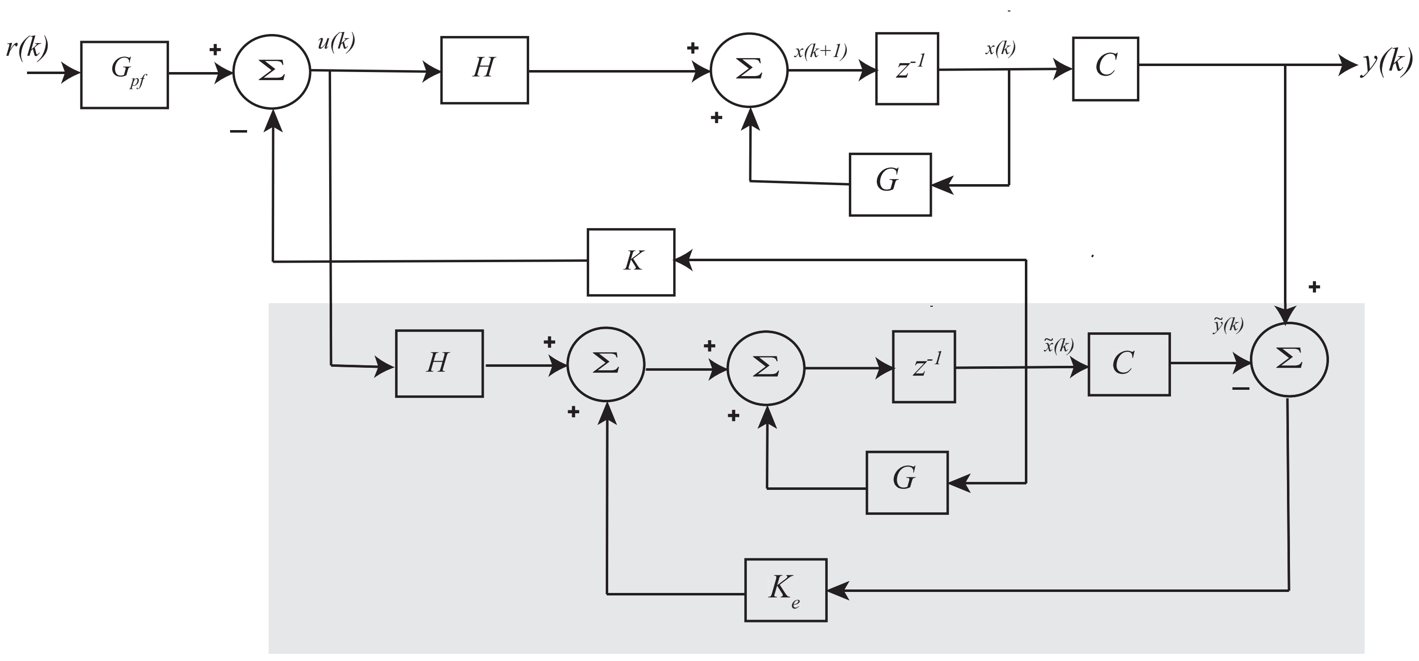

Figure 27: State variable feedback using full order observer (the shaded area) to estimate the states.

so the error has the dynamics

$$e(k+1) = (G - K_eC)\,e(k)$$

Hence the dynamics of the error is determined by the eigenvalues (poles) of the matrix $G - K_eC$. If the eigenvalues (poles) of $G - K_eC$ are within the unit circle (have magnitude less than one), then $e(k) \to 0$ as $k \to \infty$. If $e(k) \to 0$, then $x(k) - \tilde{x}(k) \to 0$, or $\tilde{x}(k) \to x(k)$, i.e., the estimated states converge to the true states. The closer the eigenvalues of $G - K_eC$ are to the origin, the more rapidly the estimates converge to the true states. However, it is important to keep in mind that the accuracy of the observer is dependent on how accurately the mathematical model represents the true system.

8.2 Choosing K_e

We need to determine how to choose K_e to obtain the observer performance we want. We want to use K_e to place the poles of the matrix $G - K_eC$. When we designed state variable feedback systems, we learned techniques to choose K to place the poles of the matrix $G - HK$. If we compare this with the observer error matrix, we have

$G - HK$	state variable feedback matrix
$G - K_eC$	observer error dynamic matrix

If we take the transpose of the observer error dynamic matrix we have

$$(G - K_eC)^T = G^T - C^TK_e^T$$

If we rename the variables $\tilde{G} = G^T$, $\tilde{K} = K_e^T$, and $\tilde{H} = C^T$, then we have

$$\tilde{G} - \tilde{H}\tilde{K}$$

and we can then use the state variable techniques we have already used to determine $\tilde{K} = K_e^T$.

Example. Design and observer with deadbeat response for the following system:

$$\begin{aligned} x(k+1) &= \begin{bmatrix} 0.1 & 0.2 \\ 0 & 0.1 \end{bmatrix} x(k) + \begin{bmatrix} 0.1 \\ 0.2 \end{bmatrix} u(k) \\ y(k) &= \begin{bmatrix} 1 & 0 \end{bmatrix} \end{aligned}$$

The first thing we need to do is be sure the system is observable or we will not be able to construct the observer. The observability matrix is

$$\begin{bmatrix} C \\ CG \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0.1 & 0.2 \end{bmatrix}$$

which has rank two, so the system is observable. We need to place the poles (eigenvalues) of $\tilde{G} - \tilde{H}\tilde{K}$ at zero. We have

$$\begin{aligned} \tilde{G} &= G^T = \begin{bmatrix} 0.1 & 0 \\ 0.2 & 0.1 \end{bmatrix} \\ \tilde{H} &= C^T = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \end{aligned}$$

We then need to find (for $\lambda = 0$)

$$\begin{bmatrix} \lambda I - \tilde{G} & \vdots & \tilde{H} \end{bmatrix} = \begin{bmatrix} -0.1 & 0 & 1 \\ -0.2 & -0.1 & 0 \end{bmatrix}$$

There is only one vector in the null space of this matrix, we will choose

$$\xi = \begin{bmatrix} 1 \\ -2 \\ 0.1 \end{bmatrix}$$

We then generate a generalized eigenvector using

$$\begin{bmatrix} -0.1 & 0 & 1 \\ -0.2 & -0.1 & 0 \end{bmatrix} \xi_g = \begin{bmatrix} 1 \\ -2 \end{bmatrix}$$

from which we get

$$\xi_g = \begin{bmatrix} 0 \\ 20 \\ 1 \end{bmatrix}$$

Finally we have

$$\begin{aligned}
\tilde{K}\begin{bmatrix} 1 \\ -2 \end{bmatrix} &= 0.1 \\
\tilde{K}\begin{bmatrix} 0 \\ 20 \end{bmatrix} &= 1
\end{aligned}$$

which we combine to get

$$\begin{aligned}
\tilde{K}\begin{bmatrix} 1 & 0 \\ -2 & 20 \end{bmatrix} &= \begin{bmatrix} 0.1 & 1 \end{bmatrix} \\
\tilde{K} &= \begin{bmatrix} 0.2 & 0.05 \end{bmatrix} \\
K_e &= \tilde{K}^T = \begin{bmatrix} 0.2 \\ 0.05 \end{bmatrix}
\end{aligned}$$

8.3 State Feedback and Observers

In what follows we assume the system model

$$\begin{aligned}
x(k+1) &= Gx(k) + Hu(k) \\
y(k) &= Cx(k) + Du(k)
\end{aligned}$$

is both controllable and observable. Assume we are using state variable feedback with a reference input $r(k)$. Then we have

$$u(k) = G_{pf}r(k) - Kx(k)$$

Next assume that instead of the actual states being used in the feedback, we are using the estimated states, so

$$u(k) = G_{pf}r(k) - K\tilde{x}(k)$$

We then have

$$\begin{aligned}
x(k+1) &= Gx(k) + H\left[G_{pf}r(k) - K\tilde{x}(k)\right] \\
&= Gx(k) - HK\tilde{x}(k) + HG_{pf}r(k) \\
&= Gx(k) - HK\tilde{x}(k) \underbrace{-HKx(k) + HKx(k)}_{subtract\ and\ add} + HG_{pf}r(k) \\
&= (G - HK)\,x(k) + HK\,(x(k) - \tilde{x}(k)) + HG_{pf}r(k) \\
&= (G - HK)\,x(k) + HKe(k) + HG_{pf}r(k)
\end{aligned}$$

From the observer equation we have

$$e(k+1) = (G - K_eC)e(k)$$

Combining these we have the state model

$$\begin{bmatrix} x(k+1) \\ e(k+1) \end{bmatrix} = \begin{bmatrix} G - HK & HK \\ 0 & G - K_eC \end{bmatrix}\begin{bmatrix} x(k) \\ e(k) \end{bmatrix} + \begin{bmatrix} HG_{pf} \\ 0 \end{bmatrix} r(k)$$

The characteristic equation for this system is

$$\det \begin{bmatrix} zI - G + HK & -HK \\ 0 & zI - G + K_eC \end{bmatrix} = 0$$

or, using properties of determinants,

$$\underbrace{\det\left[zI - G + HK\right]}_{state\ \ feedback} \underbrace{\det\left[zI - G + K_eC\right]}_{observer} = 0$$

The first determinant is what we use to assign the state feedback poles, while the second determinant is what we use to assign the observer poles. What we have shown is that we can assign the closed loop system (state variable poles) and the observer pole independently. This important result is called the *separation principle.*

8.4 General Guidelines for Observer Pole Locations

Now that we know how to construct an observer to estimate the system poles, we need some guidelines for choosing the pole locations for the observer. However, as in most engineering systems we have a conflict:

- Choose K_e so the poles of $G - K_eC$ are small (near the origin).
  - This produces estimates that converge to the true states more quickly, which is generally good.
  - This increases the system bandwidth which makes the system less robust and more sensitive to external disturbances, noise (or noisy measurements), or errors in the mathematical model of the system. This is generally bad.
- Choose K_e so the poles of $G - K_eC$ are large (near the unit circle).
  - This produces estimates that converge to the true states more slowly. This can produce unsatisfactory results using state variable feedback since the estimated states be used do not accurately reflect the state of the system.
  - This reduces the system bandwidth which generally makes the system more robust and less sensitive to external disturbances, noise, and mathematical errors in the model of the system.

As a general rule, the poles (eigenvalues) of the observer should be smaller in magnitude by a factor of three or four than the poles of the state feedback system. However, this is just a general guideline. Often you need to use some intelligent "guess and check" methods.

8.5 Full Order Observers with Integral Control

It is possible to use a full order observer to estimate the states of a system, use state variable feedback to place the poles of the closed loop system based on the estimated states, and include integral control so one of the states will have zero steady state error for a unit step input. Figure 8.5 displays a control structure that can be used for this purpose. Note that C is used in

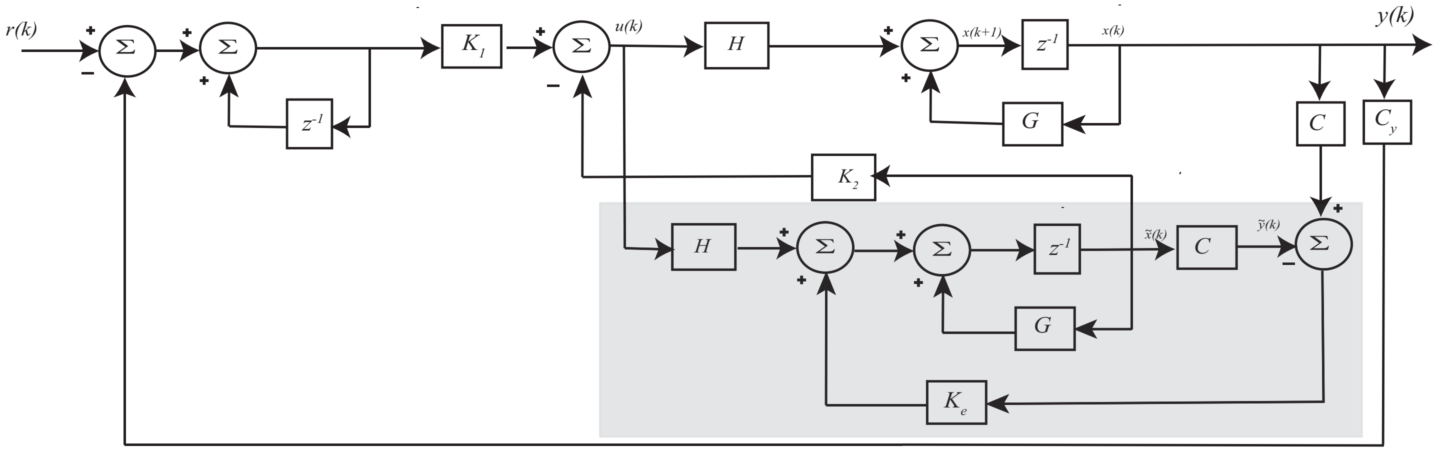

Figure 28: Structure for use of a full order observer, state variable feedback, and integral control.

the observer to indicate which states (outputs) will be input to the observer, and C_y indicates which state (or linear combination of states) will have zero steady state error. Note also that the observer is constructed independently of the state feedback gain, and the state feedback gains (K_1 and K_2) are constructed independently of the use of an observer.

8.6 Simulations

In this section, we will assume we are using our standard model,

$$\begin{bmatrix} x_1(k) \\ v_1(k) \\ x_2(k) \\ v_2(k) \end{bmatrix} = \begin{bmatrix} 0.4964 & 0.0391 & 0.5153 & 0.0095 \\ -15.5298 & 0.4361 & 14.7498 & 0.5022 \\ 0.4452 & 0.0082 & 0.3068 & 0.0358 \\ 12.7738 & 0.4326 & -21.0039 & 0.2572 \end{bmatrix} \begin{bmatrix} x_1(k) \\ v_1(k) \\ x_2(k) \\ v_2(k) \end{bmatrix} + \begin{bmatrix} 4.0181 \\ 142.8037 \\ 0.3955 \\ 29.8022 \end{bmatrix} u(k)$$

The sample interval is $T_s = 0.05$ seconds. We have not determined what the input to the observer will be yet, and have not determined which state we want to have zero steady state error.

Example. Let's assume we want the output of our system to be the first state (x_1) and we want the observer to be able to use only the third state (x_2). The we would have

$$\begin{aligned} C_y &= \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix} \\ C &= \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix} \end{aligned}$$

Let's assume the system starts at rest (all initial conditions are zero) and we estimate the initial states as $\tilde{x}_1 = 1.0$ cm, $\tilde{v}_1 = -10.0$ cm/sec, $\tilde{x}_2 = -1.0$ cm and $\tilde{v}_2 = 10.0$ cm/sec. Our estimates of the initial states (the starting points for our observer) are a bit silly, but they will better allow us to see how quickly our estimated states converge to our true states. If we place the closed loop poles all at 0.3, and the observer poles all at 0.3 then we have

$$K = \begin{bmatrix} -0.0885 & 0.0049 & 0.1038 & -0.0029 \end{bmatrix}$$

$$
\begin{aligned}
K_e^T &= \left[\begin{array}{cccc} 0.2839 & 17.1162 & 0.2964 & -24.1797 \end{array} \right] \\
G_{pf} &= 0.0206
\end{aligned}
$$

and the unit step response is shown in Figure 29. This figure demonstrates that the estimated states do converge to the real (true) system states, and the steady state error in the first state is zero. If we move the observer poles all to 0.1, then we have

$$
K_e^T = \left[\begin{array}{cccc} 0.5960 & 9.4168 & 1.0964 & -13.1210 \end{array} \right]
$$

Clearly K and G_{pf} do not change. The unit step response for this system is shown in Figure 30. Finally, if we move the observer states all to 0.05, we have

$$
K_e^T = \left[\begin{array}{cccc} 0.7570 & 7.5642 & 1.2964 & -9.3092 \end{array} \right]
$$

The step response for this system is shown in Figure 31. Comparing the results from these three figures, it is clear that as the observer poles are moved closer to the origin the observed states converge to the true states more quickly. However, it should be pointed out that this also depends on having a sufficiently accurate model.

If we wanted the output of our system to be the third state (x_2) and wanted our observer to be able to use both the first and third states, then we would have

$$
\begin{aligned}
C_y &= \left[\begin{array}{cccc} 0 & 0 & 1 & 0 \end{array} \right] \\
C &= \left[\begin{array}{cccc} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{array} \right]
\end{aligned}
$$

Our final simulation results are for the original system using both an observer and an integrator. The observer uses the first and third states (x_1 and x_2) and the output is the first state (x_1). The initial states are again $x_1 = 1.0$ cm, $v_1 = -10.0$ cm/sec, $x_2 = -1.0$ cm, and $v_2 = 10.0$ cm/sec. The system poles are all at 0.2 and the observer poles are at 0.06, 0.08, 0.1, and 0.12. The resulting state feedback and observer gains are

$$
\begin{aligned}
K_1 &= 0.0281 \\
K_2 &= \left[\begin{array}{cccc} 0.0567 & 0.0078 & 0.0109 & 0.0013 \end{array} \right] \\
K_e^T &= \left[\begin{array}{cccc} 0.7521 & -8.1982 & 0.8629 & 17.2414 \\ 0.9909 & 19.8096 & 0.3843 & -15.6217 \end{array} \right]
\end{aligned}
$$

and the simulation results are shown in Figure 32.

We should point out that even though we may know a state (or states) that are input to the full order observer, we are still estimating all of those states in the full order observer. There are two fairly good reasons for this

- There are often errors between our model of the system and the real system. Estimating all of the states sometimes helps. If we are going to force a state to be a specific value and the model is in error, this may have even larger errors in the other estimated states.
- Utilizing the known states in an observer (not trying to estimate them with the observer) leads to a *minimum order observer*, which is the subject of the next chapter. However, the math gets alot more complicated and you have to decide if it is really worth it to you as a designer.

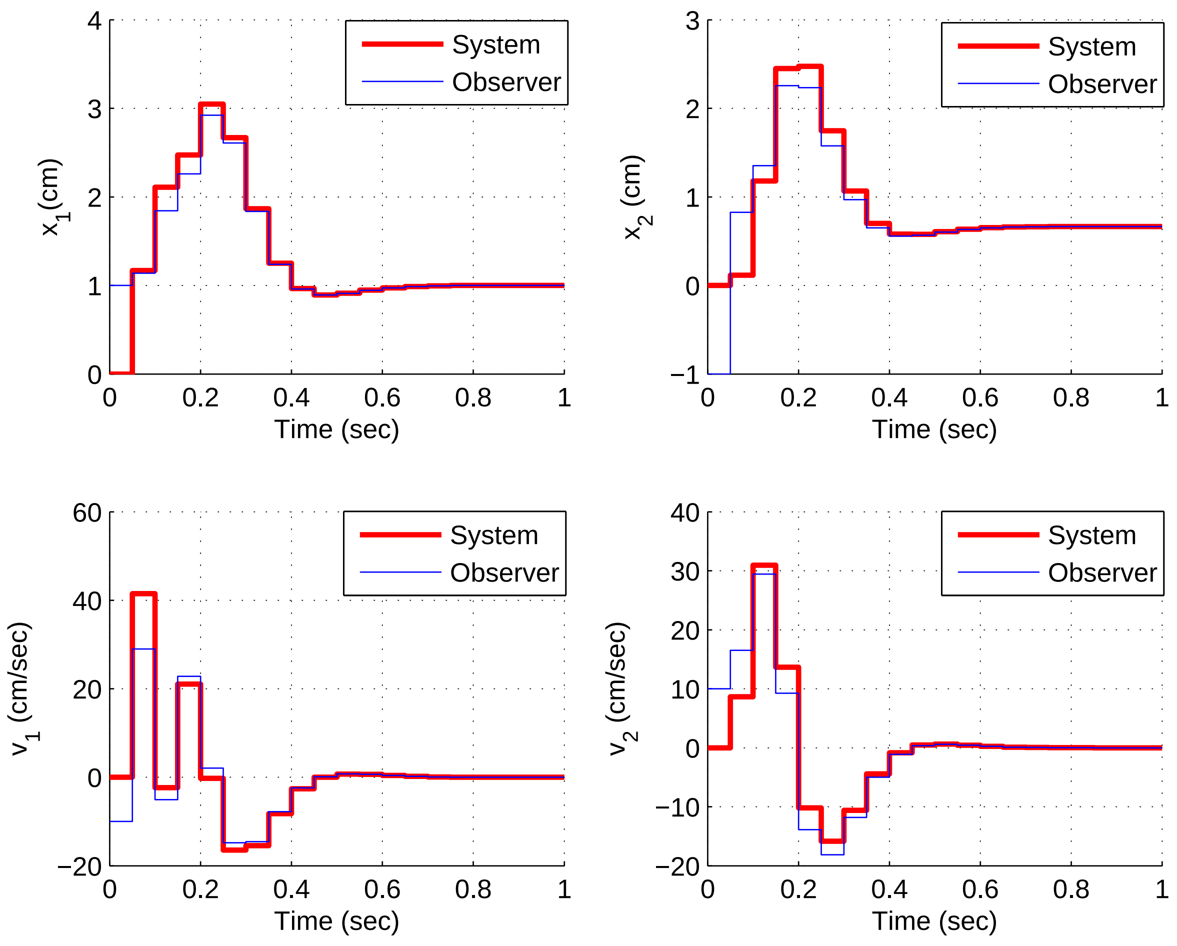

Figure 29: State response for state variable system with observer. The poles of the system and the observer are all at 0.3. The input to the observer is the first state, and the first state is the output of the system.

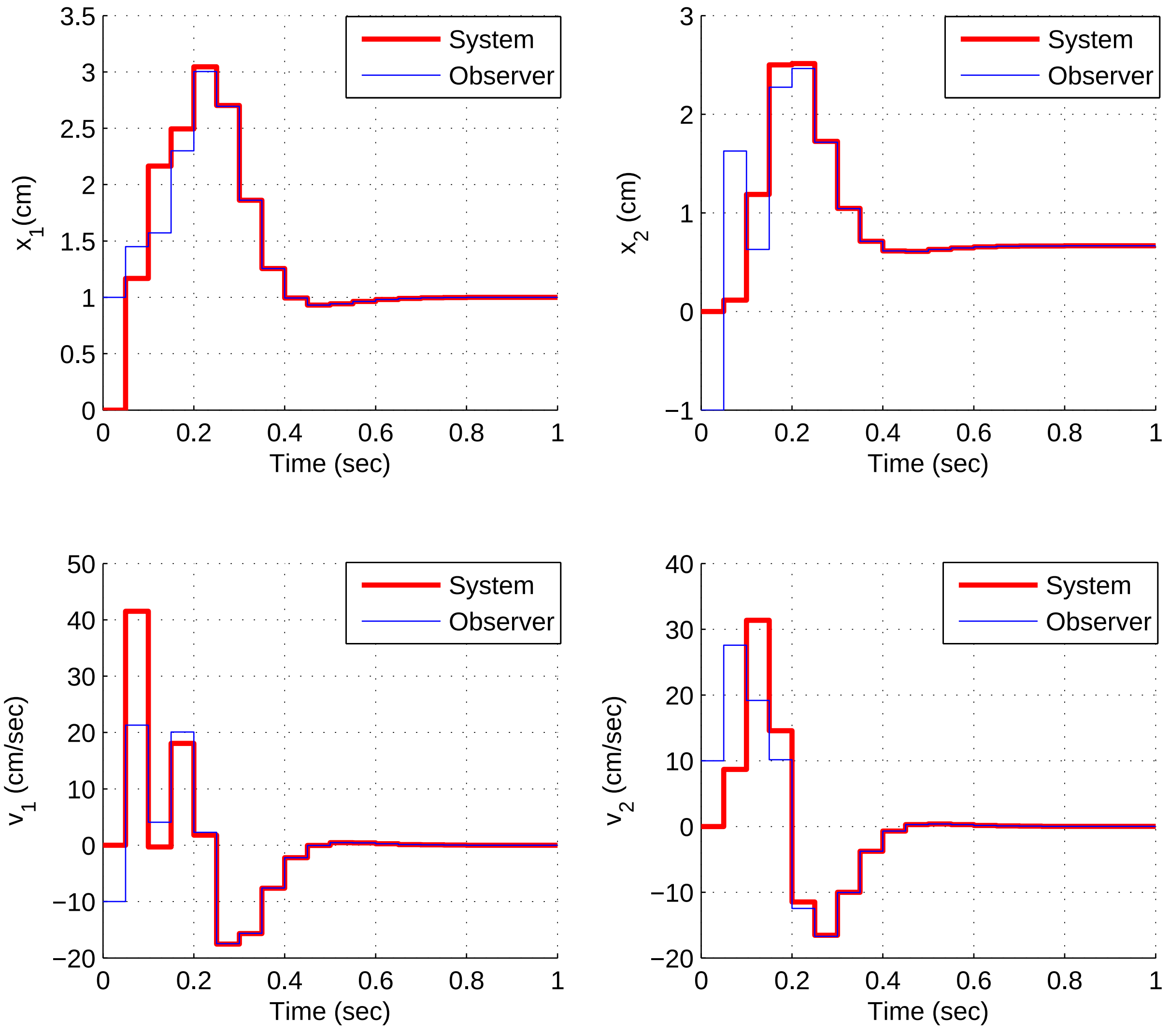

Figure 30: State response for state variable system with observer. The poles of the system are all at 0.3 and the pole of the observer are all at 0.1. The input to the observer is the first state, and the first state is the output of the system.

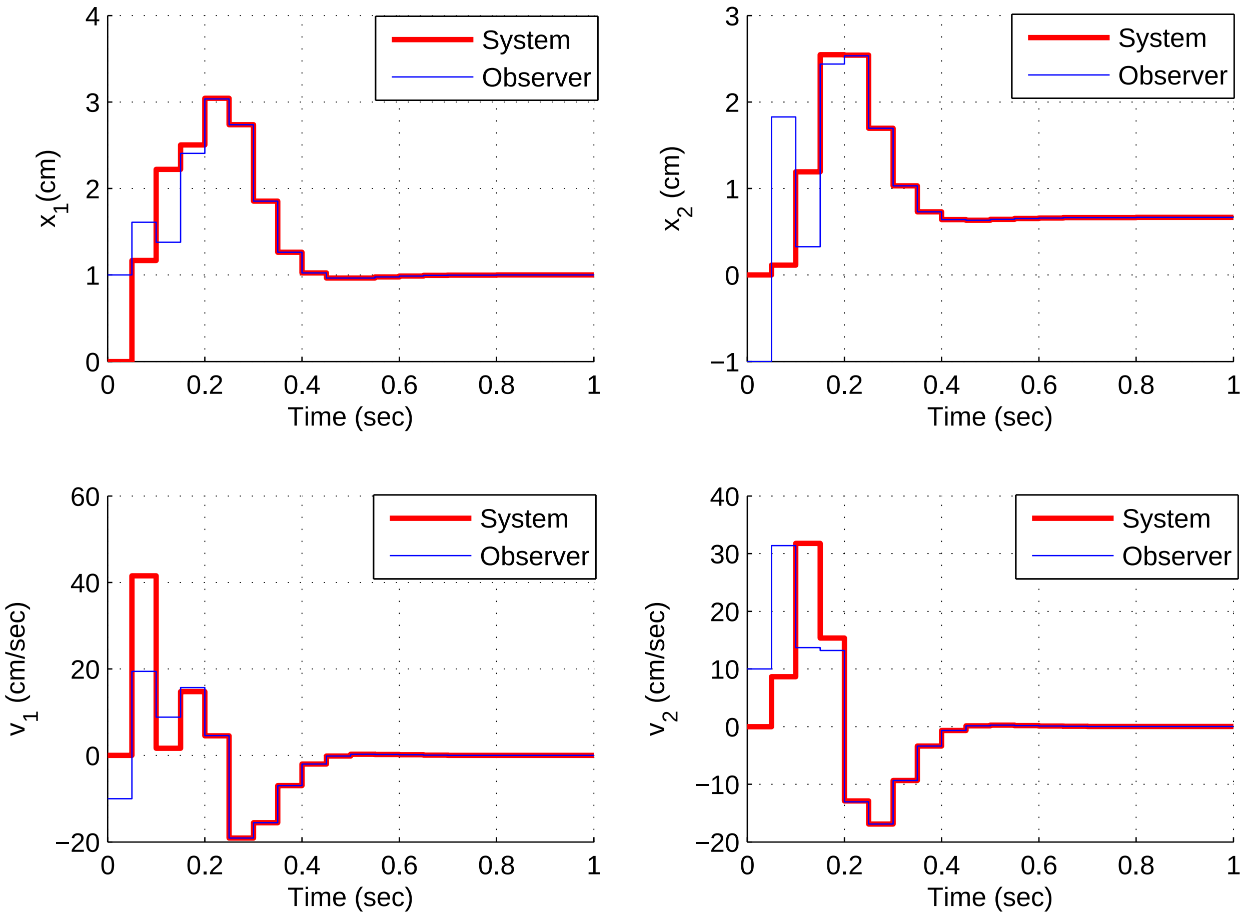

Figure 31: State response for state variable system with observer. The poles of the system are all at 0.3 and the pole of the observer are all at 0.05. The input to the observer is the first state, and the first state is the output of the system.

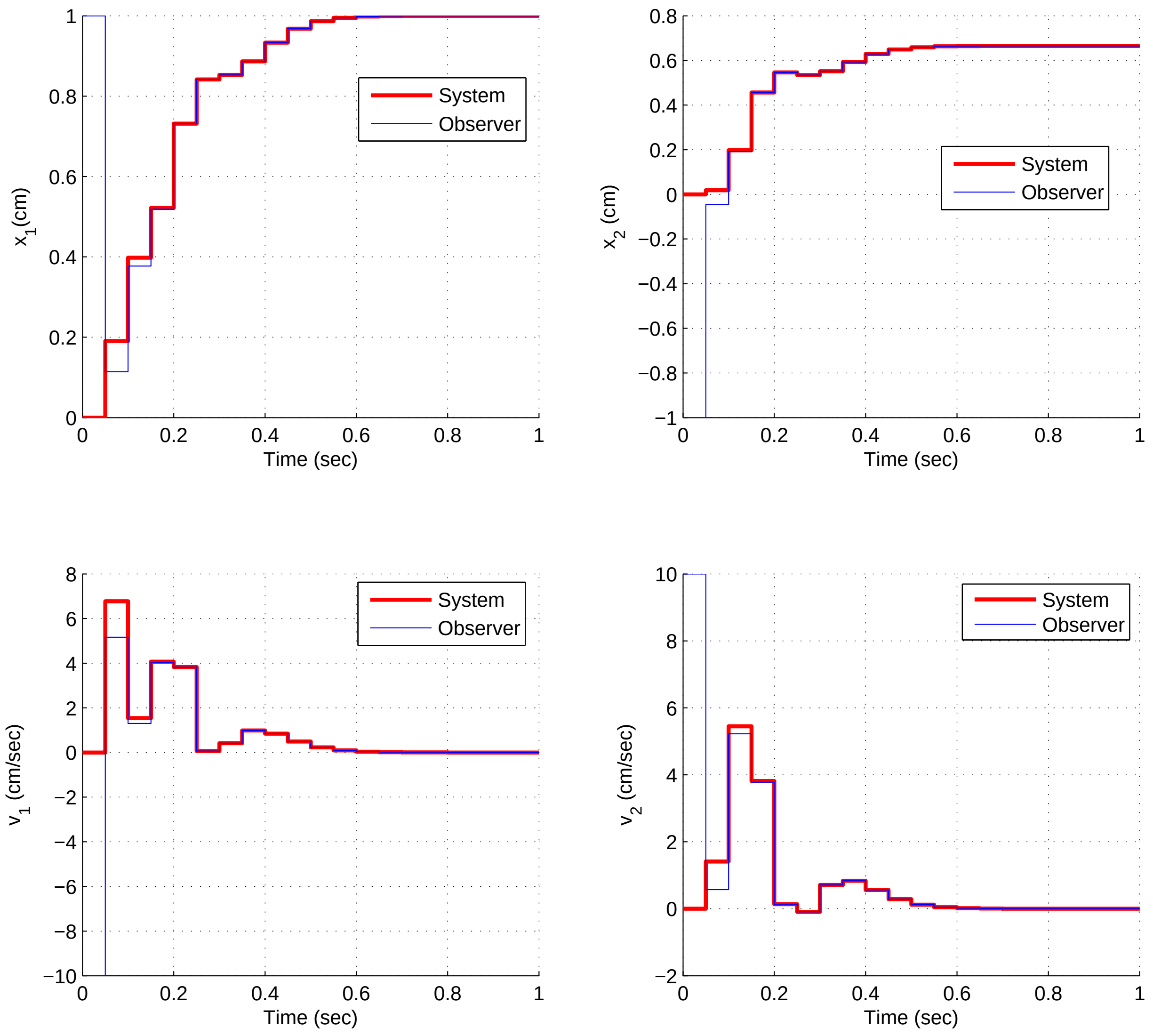

Figure 32: State response for state variable system with observer and integral control. The poles of the system are all at 0.1 and the pole of the observer are at 0.06, 0.08, 0.10, 0.12. The input to the observer is the first and third states, and the first state is the output of the system.

9 Minimum Order Observers

For full order observers we estimate *all* of the states based on the system input $u(k)$, output $y(k)$, and the mathematical model of the system. However, sometimes the states are easily measured and we would like to use the actual state rather than an estimate. For a *minimum order observer* we only estimate those states we do not know. Hence the state vector used in the state variable feedback will contain some states that are known and some states that are estimated using the observer.

Minimum order observers tend to be more sensitive to errors in the mathematical model of the system, so often only one or two known states are used in a minimum order observer, and the remaining states are estimated. In addition, although estimating fewer states and using known states may seem easier, as you will see it is really fairly complicated.

We are going to have to rewrite the state equations with all of the known states first, and the unknown states second. For example, assume $q(k)$ is our state vector, and

$$\begin{aligned} q_1(k) &= x_1(k) \\ q_2(k) &= \dot{x}_1(k) \\ q_3(k) &= x_2(k) \\ q_4(k) &= \dot{x}_2(k) \end{aligned}$$

where $\dot{x}_i(k)$ is the discrete-time representation of the derivative of $x_i(k)$. Assume our initial discrete-time state equations are

$$\begin{bmatrix} q_1(k+1) \\ q_2(k+1) \\ q_3(k+1) \\ q_4(k+1) \end{bmatrix} = \begin{bmatrix} g_{11} & g_{12} & g_{13} & g_{14} \\ g_{21} & g_{22} & g_{23} & g_{24} \\ g_{31} & g_{32} & g_{33} & g_{34} \\ g_{41} & g_{42} & g_{43} & g_{44} \end{bmatrix} \begin{bmatrix} q_1(k) \\ q_2(k) \\ q_3(k) \\ q_4(k) \end{bmatrix} + \begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \end{bmatrix} u(k)$$

$$\begin{bmatrix} y_1(k) \\ y_2(k) \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & c_{13} & c_{14} \\ c_{21} & c_{22} & c_{23} & c_{24} \end{bmatrix} \begin{bmatrix} q_1(k) \\ q_2(k) \\ q_3(k) \\ q_4(k) \end{bmatrix} + \begin{bmatrix} d_1 \\ d_2 \end{bmatrix} u(k)$$

Let's next assume that the positions are easy to measure, but the velocities are difficult to measure. Hence we will assume our known states are $q_1(k)$ and $q_3(k)$, and our unknown states are $q_2(k)$ and $q_4(k)$, Hence our new state vector will be

$$q(k) = \begin{bmatrix} q_1(k) \\ q_3(k) \\ q_2(k) \\ q_4(k) \end{bmatrix}$$

We need to rearrange our state variable system so the known states are first, and the unknown states are second. This means the G, H, and C matrices will have to change, and we do this by rearranging the rows and columns to match what we have done to the states. We rewrite the

G, H, and C matrices so the rows are in the correct order (1, 3, 2, 4) and the the columns are in the correct order (1,3,2,4),

$$\begin{bmatrix} g_{11} & g_{12} & g_{13} & g_{14} \\ g_{21} & g_{22} & g_{23} & g_{24} \\ g_{31} & g_{32} & g_{33} & g_{34} \\ g_{41} & g_{42} & g_{43} & g_{44} \end{bmatrix} \rightarrow \begin{bmatrix} g_{11} & g_{12} & g_{13} & g_{14} \\ g_{31} & g_{32} & g_{33} & g_{34} \\ g_{21} & g_{22} & g_{23} & g_{24} \\ g_{41} & g_{42} & g_{43} & g_{44} \end{bmatrix} \rightarrow \begin{bmatrix} g_{11} & g_{13} & g_{12} & g_{14} \\ g_{31} & g_{33} & g_{32} & g_{34} \\ g_{21} & g_{23} & g_{22} & g_{24} \\ g_{41} & g_{43} & g_{42} & g_{44} \end{bmatrix}$$

$$\begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \end{bmatrix} \rightarrow \begin{bmatrix} h_1 \\ h_3 \\ h_2 \\ h_4 \end{bmatrix}$$

$$\begin{bmatrix} c_{11} & c_{12} & c_{13} & c_{14} \\ c_{21} & c_{22} & c_{23} & c_{24} \end{bmatrix} \rightarrow \begin{bmatrix} c_{11} & c_{13} & c_{12} & c_{14} \\ c_{21} & c_{23} & c_{22} & c_{24} \end{bmatrix}$$

Our new state variable description is then

$$\begin{bmatrix} q_1(k+1) \\ q_3(k+1) \\ q_2(k+1) \\ q_4(k+1) \end{bmatrix} = \begin{bmatrix} g_{11} & g_{13} & g_{12} & g_{14} \\ g_{31} & g_{33} & g_{32} & g_{34} \\ g_{21} & g_{23} & g_{22} & g_{24} \\ g_{41} & g_{43} & g_{42} & g_{44} \end{bmatrix} \begin{bmatrix} q_1(k) \\ q_3(k) \\ q_2(k) \\ q_4(k) \end{bmatrix} + \begin{bmatrix} h_1 \\ h_3 \\ h_2 \\ h_4 \end{bmatrix} u(k)$$

$$\begin{bmatrix} y_1(k) \\ y_2(k) \end{bmatrix} = \begin{bmatrix} c_{11} & c_{13} & c_{12} & c_{14} \\ c_{21} & c_{23} & c_{22} & c_{24} \end{bmatrix} \begin{bmatrix} q_1(k) \\ q_3(k) \\ q_2(k) \\ q_4(k) \end{bmatrix} + \begin{bmatrix} d_1 \\ d_2 \end{bmatrix} u(k)$$

Of course, at this point we would just rename the state variables in our system,

$$\begin{aligned} q_1(k) &= x_1(k) \\ q_2(k) &= x_2(k) \\ q_3(k) &= \dot{x}_1(k) \\ q_4(k) &= \dot{x}_2(k) \end{aligned}$$

and then

$$\begin{bmatrix} q_1(k+1) \\ q_2(k+1) \\ q_2(k+1) \\ q_4(k+1) \end{bmatrix} = \begin{bmatrix} g_{11} & g_{13} & g_{12} & g_{14} \\ g_{31} & g_{33} & g_{32} & g_{34} \\ g_{21} & g_{23} & g_{22} & g_{24} \\ g_{41} & g_{43} & g_{42} & g_{44} \end{bmatrix} \begin{bmatrix} q_1(k) \\ q_3(k) \\ q_2(k) \\ q_4(k) \end{bmatrix} + \begin{bmatrix} h_1 \\ h_3 \\ h_2 \\ h_4 \end{bmatrix} u(k)$$

$$\begin{bmatrix} y_1(k) \\ y_2(k) \end{bmatrix} = \begin{bmatrix} c_{11} & c_{13} & c_{12} & c_{14} \\ c_{21} & c_{23} & c_{22} & c_{24} \end{bmatrix} \begin{bmatrix} q_1(k) \\ q_2(k) \\ q_3(k) \\ q_4(k) \end{bmatrix} + \begin{bmatrix} d_1 \\ d_2 \end{bmatrix} u(k)$$

and then

$$G = \begin{bmatrix} g_{11} & g_{13} & g_{12} & g_{14} \\ g_{31} & g_{33} & g_{32} & g_{34} \\ g_{21} & g_{23} & g_{22} & g_{24} \\ g_{41} & g_{43} & g_{42} & g_{44} \end{bmatrix}$$

$$
\begin{aligned}
H &= \begin{bmatrix} h_1 \\ h_3 \\ h_2 \\ h_4 \end{bmatrix} \\
C &= \begin{bmatrix} c_{11} & c_{13} & c_{12} & c_{14} \\ c_{21} & c_{23} & c_{22} & c_{24} \end{bmatrix}
\end{aligned}
$$

In the next section will will do this more symbolically, but when we rearrange the states these are the types of manipulations that are necessary.

9.1 Basic Structure

Let's assume we have rearranged our states so all of the known states are in the vector $x_a(k)$ and all of the unknown states, i.e., the states our observer needs to estimate, are in the vector $x_b(k)$, so the state vector $x(k)$ can be written

$$
x(k) = \begin{bmatrix} x_a(k) \\ x_b(k) \end{bmatrix}
$$

The state equations can then be written

$$
\begin{aligned}
\begin{bmatrix} x_a(k+1) \\ x_b(k+1) \end{bmatrix} &= \begin{bmatrix} G_{aa} & G_{ab} \\ G_{ba} & G_{bb} \end{bmatrix} \begin{bmatrix} x_a(k) \\ x_b(k) \end{bmatrix} + \begin{bmatrix} H_a \\ H_b \end{bmatrix} u(k) \\
y(k) &= \begin{bmatrix} C & 0 \end{bmatrix} \begin{bmatrix} x_a(k) \\ x_b(k) \end{bmatrix}
\end{aligned}
$$

We will assume we want the output of our system to be the known states. One of the most difficult parts in implementing the minimum order observer is getting the correct sized matrices, so let's determine what we know so far. We will assume that $x(k) \in \mathcal{R}^n$ and that we know (have measured) m states. Then we have

$$
\begin{aligned}
x(k) &\in \mathcal{R}^n & & \\
x_a(k) &\in \mathcal{R}^m & & x_b(k) \in \mathcal{R}^{n-m} \\
G_{aa} &\in \mathcal{R}^{m\times m} & & G_{ab} \in \mathcal{R}^{m\times(n-m)} \\
G_{ba} &\in \mathcal{R}^{(n-m)\times m} & & G_{bb} \in \mathcal{R}^{(n-m)\times(n-m)} \\
H_a &\in \mathcal{R}^m & & H_b \in \mathcal{R}^{n-m} \\
C &= I_m & & \text{an } m \times m \text{ identity matrix}
\end{aligned}
$$

The equation for the first state is

$$
x_a(k+1) = G_{aa}x_a(k) + G_{ab}x_b(k) + H_a u(k)
$$

which we can rewrite as

$$
\underbrace{x_a(k+1) - G_{aa}x_a(k) - H_a u(k)}_{\textit{these are known or measured}} = G_{ab}x_b(k)
$$

Let's define $\tilde{y}(k) = x_a(k+1) - G_{aa}x_a(k) - H_a u(k)$ so

$$\tilde{y}(k) = G_{ab}x_b(k) = \tilde{C}x_b(k) \text{ where } \tilde{C} = G_{ab}$$

The equation for the remaining (unknown) states is

$$x_b(k+1) = G_{ba}x_a(k) + G_{bb}x_b(k) + H_b u(k)$$

Define $\tilde{u}(k) = G_{ba}x_a(k) + H_b u(k)$ so that

$$x_b(k) = G_{bb}x_b(k) + \tilde{u}(k) = \tilde{G}x_b(k) + \tilde{H}\tilde{u}(k) \text{ where } \tilde{G} = G_{bb}, \tilde{H} = I_{n-m}$$

Hence we have the system (for the unknown states)

$$\begin{aligned} x_b(k+1) &= \tilde{G}x_b(k) + \tilde{H}\tilde{u}(k) \\ \tilde{y}(k) &= \tilde{C}x_b(k) \end{aligned}$$

where

$$\begin{aligned} \tilde{G} &= G_{bb} \\ \tilde{H} &= I_{n-m} \\ \tilde{C} &= G_{ab} \\ \tilde{u}(k) &= G_{ba}x_a(k) + H_b u(k) \\ \tilde{y}(k) &= x_a(k+1) - G_{aa}x_a(k) - H_a u(k) \end{aligned}$$

Now for the state variable system

$$\begin{aligned} x(k+1) &= Gx(k) + Hu(k) \\ y(k) &= Cx(k) \end{aligned}$$

we derived the observer equation

$$\tilde{x}(k+1) = (G - K_eC)\tilde{x}(k) + Hu(k) + K_e y(k)$$

For our system this becomes

$$\tilde{x}_b(k+1) = \left(\tilde{G} - K_e\tilde{C}\right)\tilde{x}_b(k) + \tilde{H}\tilde{u}(k) + K_e\tilde{y}(k)$$

or, substituting in for most of the tilde variables

$$\tilde{x}_b(k+1) = \left[G_{bb} - K_eG_{ab}\right]\tilde{x}_b(k) + \left[G_{ba}x_a(k) + H_b u(k)\right] + K_e\left[x_a(k+1) - G_{aa}x_a(k) - H_a u(k)\right]$$

Note that we need $x_a(k+1)$ (or equivalently $y(k+1)$) to determine $\tilde{x}_b(k+1)$. This is awkward, so we will have to modify this expression a bit. We can rewrite this expression as

$$\begin{aligned} \tilde{x}_b(k+1) - K_e x_a(k+1) &= \left(G_{bb} - K_eG_{ab}\right)\tilde{x}_b(k) + \left(G_{ba} - K_eG_{aa}\right)x_a(k) + \left(H_b - K_eH_a\right)u(k) \\ &= \left(G_{bb} - K_eG_{ab}\right)\tilde{x}_b(k) + \left(G_{ba} - K_eG_{aa}\right)x_a(k) + \left(H_b - K_eH_a\right)u(k) \\ &\quad + \underbrace{\left(G_{bb} - K_eG_{ab}\right)K_e x_a(k) - \left(G_{bb} - K_eG_{ab}\right)K_e x_a(k)}_{add\ and\ subtract} \\ &= \left(G_bb - K_eG_{ab}\right)\left(\tilde{x}_b(k) - K_e x_a(k)\right) + \left(G_{bb} - K_eG_{ab}\right)K_e x_a(k) \\ &\quad + \left(G_{ba} - K_eG_{aa}\right)x_a(k) + \left(H_b - K_eH_a\right)u(k) \\ &= \left(G_{bb} - K_eG_{ab}\right)\left(\tilde{x}_b(k) - K_e x_a(k)\right) + \left(H_b - K_eH_a\right)u(k) \\ &\quad + \left[\left(G_{bb} - K_eG_{ab}\right)K_e + \left(G_{ba} - K_eG_{aa}\right)\right]x_a(k) \end{aligned}$$

Now we define the new variables

$$\eta(k) = x_b(k) - K_e x_a(k) = x_b(k) - K_e y(k)$$

and

$$\tilde{\eta}(k) = \tilde{x}_b(k) - K_e x_a(k) = \tilde{x}_b(k) - K_e y(k)$$

where we have used the fact that $y(k) = x_a(k)$. Finally we can write a state variable equation for $\tilde{\eta}$, with both $x_a(k)$ and $u(k)$ as inputs, as follows:

$$\begin{aligned} \tilde{\eta}(k+1) &= (G_{bb} - K_e G_{ab})\,\tilde{\eta}(k) + \left[(G_{bb} - K_e G_{ab})\,K_e + (G_{ba} - K_e G_{aa})\right] x_a(k) \\ &\quad + (H_b - K_e H_a)\,u(k) \end{aligned}$$

If we define

$$\begin{aligned} \tilde{G} &= G_{bb} - K_e G_{ab} \\ \tilde{H} &= \tilde{G} K_e + (G_{ba} - K_e G_{aa}) \\ \tilde{F} &= H_b - K_e H_a \end{aligned}$$

Then

$$\tilde{\eta}(k+1) = \tilde{G}\tilde{\eta}(k) + \tilde{H} x_a(k) + \tilde{F} u(k)$$

and

$$\tilde{x}(k) = \begin{bmatrix} x_a(k) \\ \tilde{x}_b(k) \end{bmatrix} = \begin{bmatrix} y(k) \\ \tilde{\eta}(k) + K_e y(k) \end{bmatrix} = \begin{bmatrix} 0 \\ I_{n-m} \end{bmatrix} \tilde{\eta}(k) + \begin{bmatrix} I_m \\ K_e \end{bmatrix} y(k)$$

or

$$\tilde{x}(k) = \tilde{C}\tilde{\eta}(k) + \tilde{D} y(k)$$

where

$$\tilde{C} = \begin{bmatrix} 0 \\ I_{n-m} \end{bmatrix} \qquad \tilde{D} = \begin{bmatrix} I_m \\ K_e \end{bmatrix}$$

Note that $\tilde{x}(k)$ is the *output* of the observer. A graphical implementation of the minimum order observer is shown in Figure 9.1. All that remains now is to find K_e for the observer.

9.2 Determining K_e

To find an equation for K_e, we begin with the following previously derived expressions:

$$\begin{aligned} \tilde{x}_b(k+1) &= \left[G_{bb} - K_e G_{ab}\right] \tilde{x}_b(k) + \left[G_{ba} x_a(k) + H_b u(k)\right] + K_e\left[x_a(k+1) - G_{aa} x_a(k) - H_a u(k)\right] \\ x_b(k+1) &= G_{ba} x_a(k) + G_{bb} x_b(k) + H_b u(k) \\ G_{ab} x_b(k) &= x_a(k+1) - G_{aa} x_a(k) - H_a u(k) \end{aligned}$$

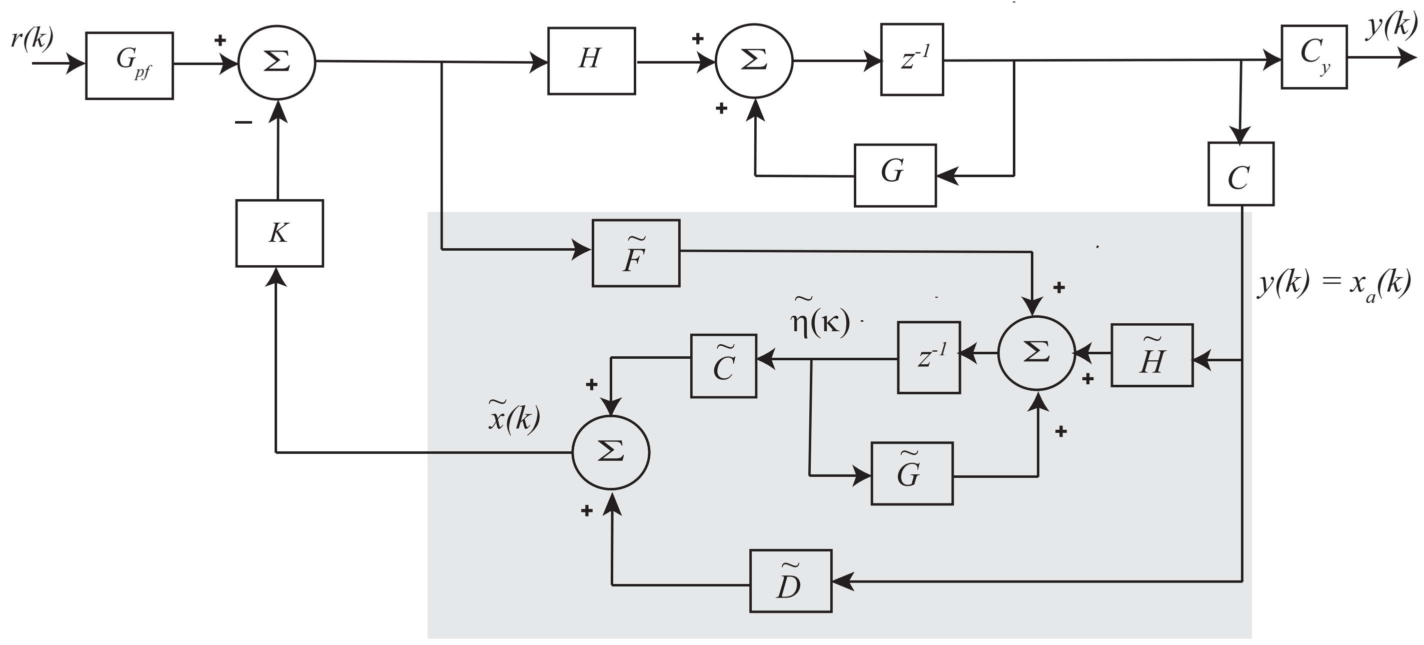

Figure 33: Structure for use of a minimum order observer and state variable feedback. The minimum order observer is in the shaded area.

We then define the observer error $e(k)$ as the difference between the true state $x_b(k)$ and the state estimated from the observer, $\tilde{x}_b(k)$. Hence we have $e(k) = x_b(k) - \tilde{x}_b(k)$, or

$$\begin{aligned}
e(k+1) &= x_b(k+1) - \tilde{x}_b(k+1) \\
&= \{G_{ba}x_a(k) + G_{bb}x_b(k) + H_b u(k)\} \\
&\quad - \{[G_{bb} - K_e G_{ab}]\,\tilde{x}_b(k) + [G_{ba}x_a(k) + H_b u(k)] + K_e\,[x_a(k+1) - G_{aa}x_a(k) - H_a u(k)]\} \\
&= G_{bb}\,(x_b(k) - \tilde{x}_b(k)) + K_e G_{ab}\tilde{x}_b(k) - K_e\,[x_a(k+1) - G_{aa}x_a(k) - H_a u(k)] \\
&= G_{bb}\,(x_b(k) - \tilde{x}_b(k)) + K_e G_{ab}\tilde{x}_b(k) - K_e G_{ab} x_b(k) \\
&= (G_{bb} - K_e G_{ab})\,(x_b(k) - \tilde{x}_b(k))
\end{aligned}$$

or

$$e(k+1) = (G_{bb} - K_e G_{ab})\,e(k)$$

This is the observer error equation! For the full order observer we had the equation

$$e(k+1) = (G - K_e C)\,e(k)$$

We need to use K_e to place the the poles of the minimum order observer. We do this exactly as we did for the full order observer using G_{bb} as G, and using G_{ab} as C.

Finally, for this minimum order observer to work, the rank of the matrix

$$\begin{bmatrix} G_{ab} \\ G_{ab}G_{bb} \\ G_{ab}G_{bb}^2 \\ \vdots \\ G_{ab}G_{bb}^{n-m-1} \end{bmatrix}$$

has rank $n-m$. We only need this to have rank $n-m$ since we are only trying to find $n-m$ states.

9.3 Examples

Example. For the system defined by

$$\begin{aligned} x(k+1) &= \begin{bmatrix} 0 & 0.1 & 0.2 \\ 0.2 & 0 & 0.1 \\ 0.1 & 0.1 & 0.1 \end{bmatrix} x(k) + \begin{bmatrix} 0.1 \\ 0.2 \\ 0.3 \end{bmatrix} u(k) \\ y(k) &= \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \end{aligned}$$

assume we can measure states one and three and want to design a minimum order deadbeat observer to estimate the second state. The system output is to remain the same.

Rearranging our system so the know states are first and the unknown state is second, we have the state variable model

$$\begin{bmatrix} x_1(k+1) \\ x_3(k+1) \\ x_2(k+1) \end{bmatrix} = \begin{bmatrix} 0 & 0.2 & 0.1 \\ 0.1 & 0.1 & 0.1 \\ 0.2 & 0.1 & 0 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_3(k) \\ x_2(k) \end{bmatrix} + \begin{bmatrix} 0.1 \\ 0.3 \\ 0.2 \end{bmatrix} u(k)$$

Since the output for the system is $x_3(k)$, we have

$$y(k) = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_3(k) \\ x_2(k) \end{bmatrix}$$

Hence we have

$$C_y = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}$$

For the observer we have

$$y(k) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_3(k) \\ x_2(k) \end{bmatrix}$$

so

$$C = \begin{bmatrix} I_m & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

We can identify the submatrices as follows:

$$\begin{array}{ll} G_{aa} = \begin{bmatrix} 0 & 0.2 \\ 0.1 & 0.1 \end{bmatrix} & G_{ab} = \begin{bmatrix} 0.1 \\ 0.1 \end{bmatrix} \\ G_{ba} = \begin{bmatrix} 0.2 & 0.1 \end{bmatrix} & G_{bb} = 0 \\ H_a = \begin{bmatrix} 0.1 \\ 0.3 \end{bmatrix} & H_b = 0.2 \end{array}$$

Next we check to be sure the system is observable. We have

$$G_{ab} = \begin{bmatrix} 0.1 \\ 0.1 \end{bmatrix} \quad G_{ab}G_{bb} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \begin{bmatrix} G_{ab} \\ G_{ab}G_{bb} \end{bmatrix} = \begin{bmatrix} 0.1 \\ 0.1 \\ 0 \\ 0 \end{bmatrix}$$

which has rank one, which is equal to $n - m = 3 - 2 = 1$. Hence we will be able to construct an observer. The error equation for the observer is then

$$e(k+1) = (G_{bb} - K_e G_{ab})\, e(k)$$

Taking z-transforms of this scalar equation, and assuming the initial condition $e(0)$, we have

$$zE(z) - ze(0) = (G_{bb} - K_e G_{ab})\, E(z)$$

or

$$E(z) = \frac{ze(0)}{z - G_{bb} - K_e G_{ab}}$$

For deadbeat response we want all of the poles at zero, so we want

$$G_{bb} - K_e G_{ab} = 0 - \begin{bmatrix} K_{e,1} K_{e,2} \end{bmatrix} \begin{bmatrix} 0.1 \\ 0.1 \end{bmatrix} = 0$$

One solution to this is $K_e = \begin{bmatrix} 1 & -1 \end{bmatrix}$. We then have

$$\begin{aligned} \tilde{G} &= G_{bb} - K_e G_{ab} = 0 \\ \tilde{H} &= \tilde{G}K_e + G_{ba} - K_e G_{aa} = \begin{bmatrix} 0.3 & 0 \end{bmatrix} \\ \tilde{F} &= H_b - K_e H_a = 0.4 \\ \tilde{C} &= \begin{bmatrix} 0 \\ I_{n-m} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \\ \tilde{D} &= \begin{bmatrix} I_m \\ K_e \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & -1 \end{bmatrix} \end{aligned}$$

Example. For the original system in the previous example, assume the only known state is the third state and we want to design a minimum order deadbeat controller to estimate states one and two. Assume the output of the original system is state two. The new system description is

$$\begin{bmatrix} x_3(k) \\ x_1(k) \\ x_2(k) \end{bmatrix} = \begin{bmatrix} 0.1 & 0.1 & 0.1 \\ 0.2 & 0 & 0.1 \\ 0.1 & 0.2 & 0 \end{bmatrix} \begin{bmatrix} x_3(k) \\ x_1(k) \\ x_2(k) \end{bmatrix} + \begin{bmatrix} 0.3 \\ 0.1 \\ 0.2 \end{bmatrix} u(k)$$

Since the output for the system is $x_2(k)$, we have

$$y(k) = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_3(k) \\ x_1(k) \\ x_2(k) \end{bmatrix}$$

Hence we have

$$C_y = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}$$

For the observer we have

$$y(k) = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_3(k) \\ x_1(k) \\ x_2(k) \end{bmatrix}$$

so

$$C = \begin{bmatrix} I_m & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$$

We can identify the submatrices as follows:

$$G_{aa} = 0.1 \qquad G_{ab} = \begin{bmatrix} 0.1 & 0.1 \end{bmatrix}$$

$$G_{ba} = \begin{bmatrix} 0.2 \\ 0.1 \end{bmatrix} \qquad G_{bb} = \begin{bmatrix} 0 & 0.1 \\ 0.2 & 0 \end{bmatrix}$$

$$H_a = 0.3 \quad H_b \begin{bmatrix} 0.1 \\ 0.2 \end{bmatrix}$$

Next we check to be sure the system is observable. We have

$$G_{ab} = \begin{bmatrix} 0.1 & 0.1 \end{bmatrix} \quad G_{ab}G_{bb} = \begin{bmatrix} 0.02 & 0.01 \end{bmatrix} \quad \begin{bmatrix} G_{ab} \\ G_{ab}G_{bb} \end{bmatrix} = \begin{bmatrix} 0.1 & 0.1 \\ 0.02 & 0.01 \end{bmatrix}$$

which has rank two, which is equal to $n - m = 3 - 1 = 2$. Hence we will be able to construct an observer. The error equation for the observer is then

$$e(k+1) = (G_{bb} - K_e G_{ab})\, e(k)$$

To make this look like pole placement, we identify $G = G_{bb}^T$, $H = G_{ab}^T$, and $K = K_e^T$, so

$$G = \begin{bmatrix} 0 & 0.2 \\ 0.1 & 0 \end{bmatrix}, \quad H = \begin{bmatrix} 0.1 \\ 0.1 \end{bmatrix}$$

We then have, for $\lambda = 0$,

$$\begin{bmatrix} \lambda I - G & \vdots & H \end{bmatrix} \xi = \begin{bmatrix} 0 & -0.2 & 0.1 \\ -0.1 & 0 & 0.1 \end{bmatrix} \xi = 0$$

The dimension of the null space for this matrix is one, so we will only find one vector in the null space. One such vector is

$$\xi = \begin{bmatrix} 2 & 1 & 2 \end{bmatrix}^T$$

Finding a generalized eigenvector we then solve

$$\begin{bmatrix} 0 & -0.2 & 0.1 \\ -0.1 & 0 & 0.1 \end{bmatrix} \xi_g = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$

which has solution

$$\xi_g = \begin{bmatrix} -10 & -10 & 0 \end{bmatrix}^T$$

Putting these together we have

$$K \begin{bmatrix} 2 & -10 \\ 1 & -10 \end{bmatrix} = \begin{bmatrix} 2 & 0 \end{bmatrix}$$

or $K = \begin{bmatrix} 2 & -2 \end{bmatrix}$. To get K_e we use the fact that $K_e = K^T$, so

$$K_e = \begin{bmatrix} 2 \\ -2 \end{bmatrix}$$

We then have

$$\begin{aligned}
\tilde{G} &= G_{bb} - K_e G_{ab} = \begin{bmatrix} -0.2 - 0.1 & \\ 0.4 & 0.2 \end{bmatrix} \\
\tilde{H} &= \tilde{G} K_e + G_{ba} - K_e G_{aa} = \begin{bmatrix} -0.2 \\ 0.7 \end{bmatrix} \\
\tilde{F} &= H_b - K_e H_a = \begin{bmatrix} -0.5 \\ 0.8 \end{bmatrix} \\
\tilde{C} &= \begin{bmatrix} 0 \\ I_{n-m} \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \\
\tilde{D} &= \begin{bmatrix} I_m \\ K_e \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ -2 \end{bmatrix}
\end{aligned}$$

9.4 Minimum Order Observers with Integral Control

It is possible to use a minimum order observer to estimate the states of a system, use state variable feedback to place the poles of the closed loop system based on the estimated states, and include integral control so one of the states will have zero steady state error for a unit step input. Figure 9.4 displays a control structure that can be used for this purpose. Note that C is used in the observer to indicate which states are known and will be input to the observer, and C_y indicates which state (or linear combination of states) will have zero steady state error. Note that the observer is constructed independently of the state feedback gain, and the state feedback gains (K_1 and K_2) are constructed independently of the use of an observer.

9.5 Simulations

In this section, we will assume we are using our standard model,

$$\begin{bmatrix} x_1(k) \\ v_1(k) \\ x_2(k) \\ v_2(k) \end{bmatrix} = \begin{bmatrix} 0.4964 & 0.0391 & 0.5153 & 0.0095 \\ -15.5298 & 0.4361 & 14.7498 & 0.5022 \\ 0.4452 & 0.0082 & 0.3068 & 0.0358 \\ 12.7738 & 0.4326 & -21.0039 & 0.2572 \end{bmatrix} \begin{bmatrix} x_1(k) \\ v_1(k) \\ x_2(k) \\ v_2(k) \end{bmatrix} + \begin{bmatrix} 4.0181 \\ 142.8037 \\ 0.3955 \\ 29.8022 \end{bmatrix} u(k)$$

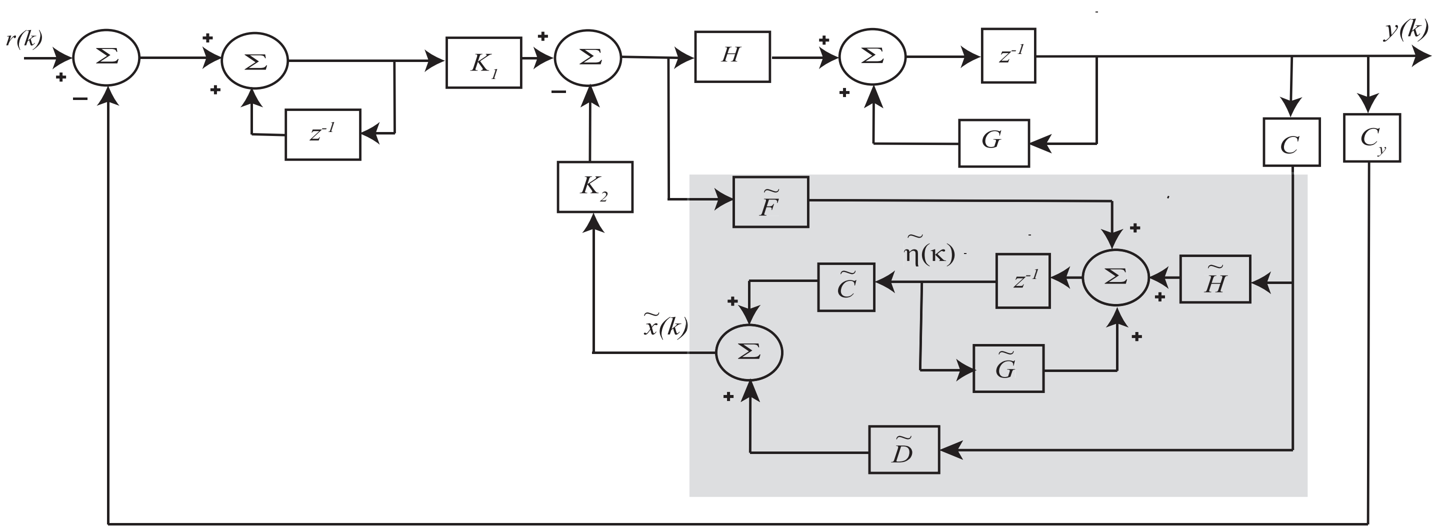

Figure 34: Structure for use of a minimum order observer, state variable feedback, and integral control. The minimum order observer is in the shaded region.

The sample interval is $T_s = 0.05$ seconds. We have not determined what the input to the observer will be yet, and have not determined which state we want to have zero steady state error. The first thing we need to do is to determine which states are assumed to be known and which states we are estimating. We then need to rearrange our states so the known states are at the top and the unknown states follow. For the purposes of using the same model in all of our examples, we will put the states in the following order: $x_2(k)$, $x_1(k)$, $v_1(k)$, and then $v_2(k)$. Our new state variable model is then

$$\begin{bmatrix} x_2(k) \\ x_1(k) \\ v_1(k) \\ v_2(k) \end{bmatrix} = \begin{bmatrix} 0.3068 & 0.4452 & 0.0082 & 0.0358 \\ 0.5153 & 0.4964 & 0.0391 & 0.0095 \\ 14.7498 & -15.5298 & 0.4361 & 0.5022 \\ -21.0039 & 12.7738 & 0.4326 & 0.2572 \end{bmatrix} \begin{bmatrix} x_2(k) \\ x_1(k) \\ v_1(k) \\ v_2(k) \end{bmatrix} + \begin{bmatrix} 0.3955 \\ 4.0181 \\ 142.8037 \\ 29.8022 \end{bmatrix} u(k)$$

Example. Let's assume we want the output of our system to be the first state (x_2) and we assume that this state is also the measured (known) state. Then we would have

$$\begin{aligned} C_y &= \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix} \\ C &= \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix} \end{aligned}$$

If we place the closed loop poles all at 0.5, and the observer poles at 0.1, 0.15, and 0.2 then we have

$$\begin{aligned} K &= \begin{bmatrix} 0.2421 & -0.1957 & 0.0017 & -0.0018 \end{bmatrix} \\ K_e^T &= \begin{bmatrix} 0.5521 & -10.0558 & 16.0980 \end{bmatrix} \end{aligned}$$

$$
\begin{aligned}
G_{pf} &= 0.0081 \\
\tilde{G} &= \begin{bmatrix} 0.2506 & 0.0346 & -0.0103 \\ -11.0532 & 0.5181 & 0.8619 \\ 5.6074 & 0.3013 & -0.3187 \end{bmatrix} \\
\tilde{H} &= \begin{bmatrix} -0.0290 \\ 20.3972 \\ -31.0063 \end{bmatrix} \\
\tilde{F} &= \begin{bmatrix} 3.7997 \\ 146.7809 \\ 23.4352 \end{bmatrix} \\
\tilde{C} &= \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \\
\tilde{D} &= \begin{bmatrix} 1.000 \\ 0.5521 \\ -10.0558 \\ 16.0980 \end{bmatrix}
\end{aligned}
$$

Let's assume the system starts with all initial conditions zero except $x_2 = -1.0$ cm, and we estimate the initial states as $\tilde{x}_1 = 1.0$ cm, $\tilde{v}_1 = -10.0$ cm/sec, and $\tilde{v}_2 = 10.0$ cm/sec. Our estimates of the initial states (the starting points for our observer) are a bit silly, but they will better allow us to see how quickly our estimated states converge to our true states. and the unit step response is shown in Figure 35. This figure demonstrates that the estimated states do converge to the real (true) system states, and the steady state error in the first state (x_2) is zero. Note that there is no error in the estimate of x_2 since the minimum order observer assumed the state was measured. Note also that out initial estimates for the unknown states (x_1, v_1 and v_2) are modified before they are saved to be plotted. Hence these estimated states appear to be incorrect based on our prescribed initial conditions.

Next, let's assume the same case as before, except we now assume both x_1 and x_2 are known. Leaving the state variables in the same order as before, we have

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

We again place all of the system poles at 0.5 and the minimum order observer poles at 0.1 and 0.15. We then have

$$
\begin{aligned}
K &= \begin{bmatrix} 0.2421 & -0.1957 & 0.0017 & -0.0018 \end{bmatrix} \\
K_e &= \begin{bmatrix} 12.4487 & 6.0026 \\ 0.0714 & 11.0545 \end{bmatrix} \\
G_{pf} &= 0.0081 \\
\tilde{G} &= \begin{bmatrix} 0.1000 & 0.000 \\ 0.000 & 0.1500 \end{bmatrix}
\end{aligned}
$$

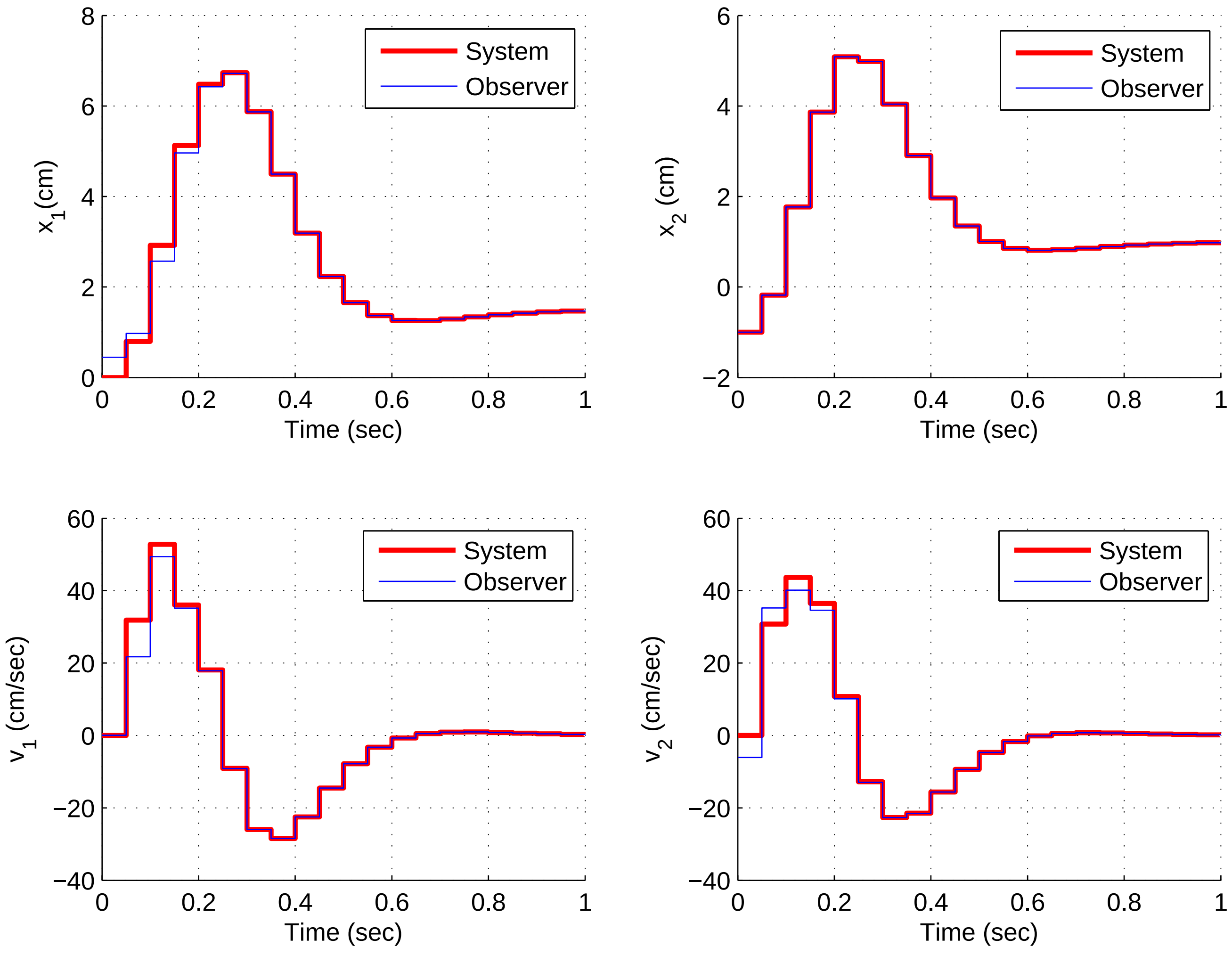

Figure 35: State response for state variable system with minimum order observer with state x_2 assumed to be known, and x_2 is the system output. The poles of the system are all at 0.5 and the poles of the observer are at 0.1, 0.15, and 0.2.

$$
\begin{aligned}
\tilde{H} &= \begin{bmatrix} 9.0830 & -23.4507 \\ -26.7114 & 8.9132 \end{bmatrix} \\
\tilde{F} &= \begin{bmatrix} 113.7613 \\ -14.6438 \end{bmatrix} \\
\tilde{C} &= \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \\
\tilde{D} &= \begin{bmatrix} 1.000 & 0.000 \\ 0.000 & 1.000 \\ 12.4487 & 6.0026 \\ 0.0714 & 11.0545 \end{bmatrix}
\end{aligned}
$$

The unit step response of the system is shown in Figure 36. Note that there is no error in the estimates for x_1 and x_2 since these states are assume to be known.

Our final simulation results are for the original system using both an observer and an integrator. The observer uses the first two states (x_1 and x_2) and the output is the second state (x_1). With all parameters as before (system poles at 0.5 and observer poles at 0.1 and 0.15), we have

$$
\begin{aligned}
K_1 &= 0.0027 \\
K_2 &= \begin{bmatrix} 0.1664 & -0.1307 & 0.0036 & -0.0021 \end{bmatrix}
\end{aligned}
$$

and the simulation results are shown in Figure 37. All of the other parameters are the same as in the previous example.

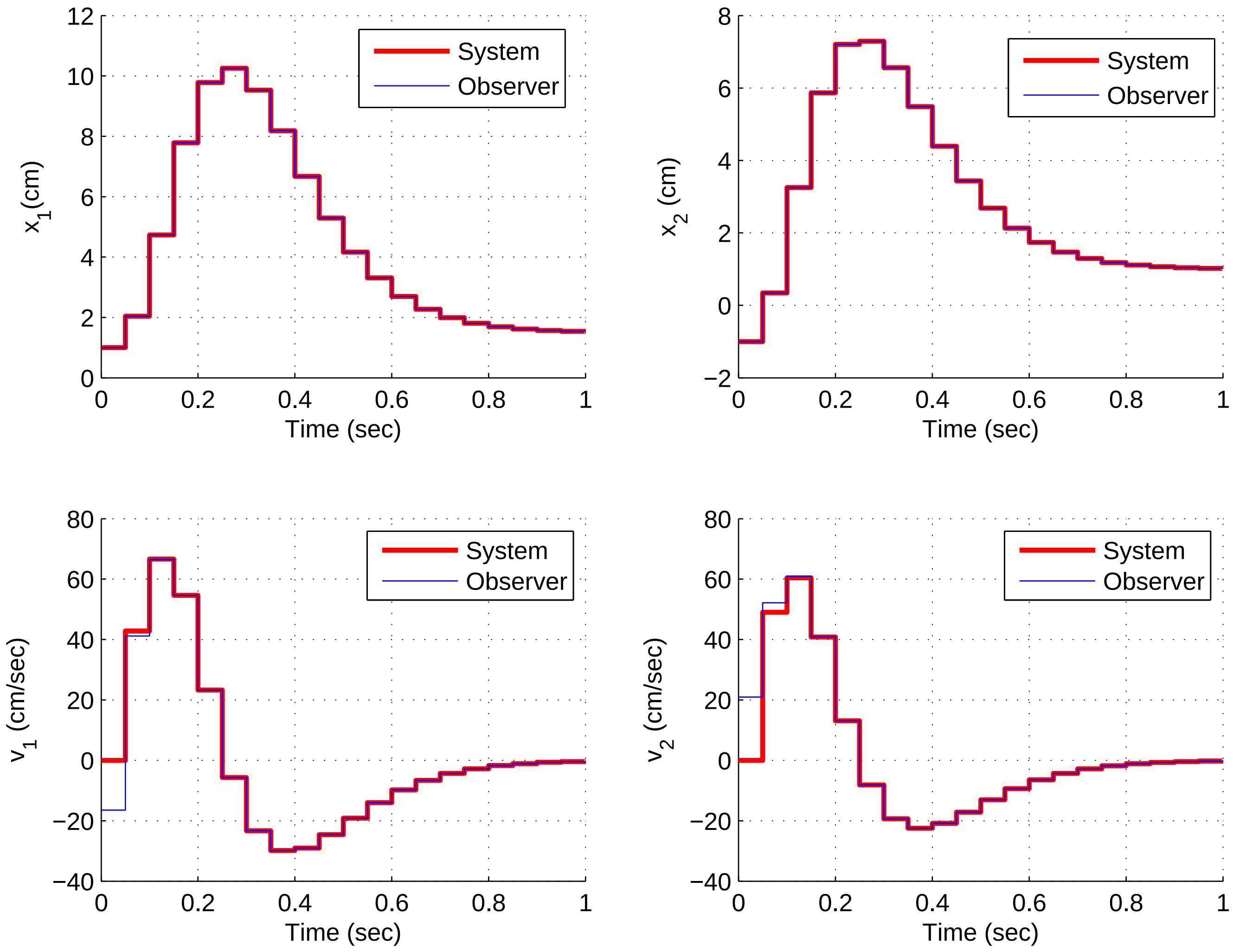

Figure 36: State response for state variable system with minimum order observer with states x_2 and x_1 assumed to be known. The output is x_2. The poles of the system are all at 0.5 and the poles of the observer are at 0.1 and 0.15.

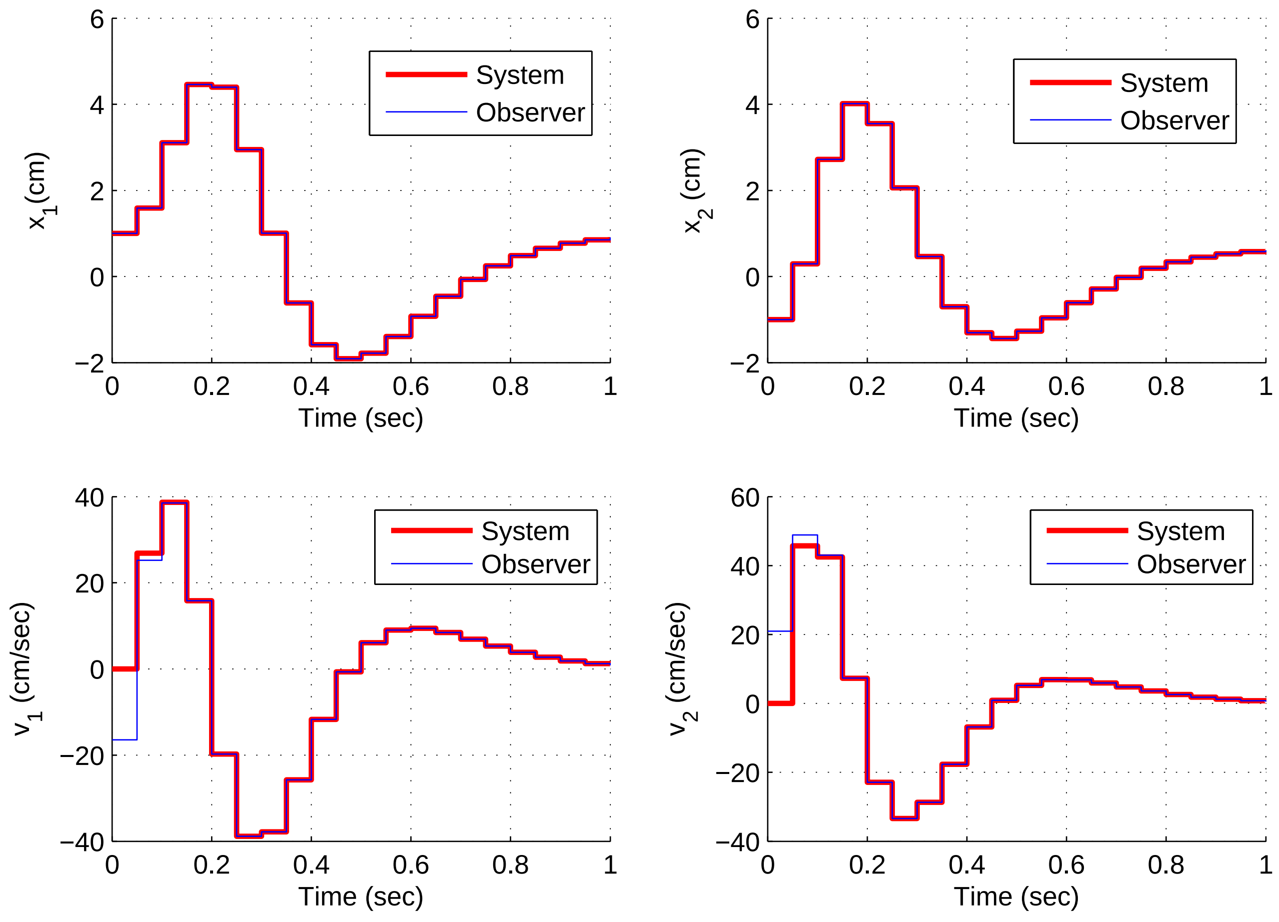

Figure 37: State response for state variable system with observer and integral control. The poles of the system are all at 0.5 and the poles of the observer are at 0.1 and 0.15. The input to the observer is the first and second states (x_2 and x_1), and the second state (x_1)is the output of the system.

10 Transfer Functions of Observer Based Controllers

Although we have focused on state variable methods, sometimes it is convenient to determine the transfer function of an observer based controller as an alternative to design using classical transfer function methods. The transfer function approach is for a single-input single-output system. This is usually easier to implement than a state variable approach. Sometimes the transfer function of the observer based controller works just fine, while other times the performance is not acceptable. In addition, the observer based controller may, however, be unstable or make the system unstable. The ability to construct the observer based controller is sometimes useful, and it is just another possible solution to a problem. It should be point out that we will derive the observer-controller transfer function using a regulator (a system with a zero input), but theses systems can be tried with either type of application. Finally, it is important to realize that for these systems, we must have $C_y = C$. That is, the observer must use the output of the system.

10.1 Full Order Observers

Consider the state variable model

$$\begin{aligned} x(k+1) &= Gx(k) + Hu(k) \\ y(k) &= Cx(k) \end{aligned}$$

As we have seen before, the full order observer follows the following dynamical equation

$$\begin{aligned} \tilde{x}(k+1) &= G\tilde{x}(k) + Hu(k) + K_e\left(y(k) - \tilde{y}(k)\right) \\ &= G\tilde{x}(k) + Hu(k) + K_e\left(y(k) - C\tilde{x}(k)\right) \\ &= \left(G - K_eC\right)\tilde{x}(k) + Hu(k) + K_ey(k) \end{aligned}$$

If we assume the input is zero, i.e., we have a regulator and just want to hold the system at zero, then $u(k) = -K\tilde{x}(k)$, and we have

$$\tilde{x}(k+1) = \left(G - K_eC - HK\right)\tilde{x}(k) + K_ey(k)$$

We know want to find the transfer function between the input to our system $y(k)$ and the output (input to the plant) $u(k)$. Taking z transforms we have

$$\begin{aligned} z\tilde{X}(z) &= \left(G - K_eC - HK\right)\tilde{X}(z) + K_eY(z) \\ \left(zI - G + K_eC + HK\right)\tilde{X}(z) &= K_eY(z) \\ \tilde{X}(z) &= \left(zI - G + K_eC + HK\right)^{-1}K_eY(z) \end{aligned}$$

Then since $u(k) = -K\tilde{x}(k)$ we have

$$U(z) = -K\left(zI - G + K_eC + HK\right)^{-1}K_eY(z)$$

Hence for scalar input and output we have

$$\frac{U(z)}{-Y(z)} = K\left(zI - G + K_eC + HK\right)^{-1}K_e$$

This transfer function of observer based controllers is often implemented as shown in Figure 10.1 (Configuration A or Configuration B). Note that both of these configurations can be used as controllers also (with a non-zero input), but they may not work very well. In addition, the second configuration (with the transfer function in the feedback path) often works better. Finally, it should again be pointed out that although the poles for both the observer and controller are designed to be stable, we cannot guarantee the stability of the transfer function. For these transfer functions, the only thing to do is try them and then simulate the systems.

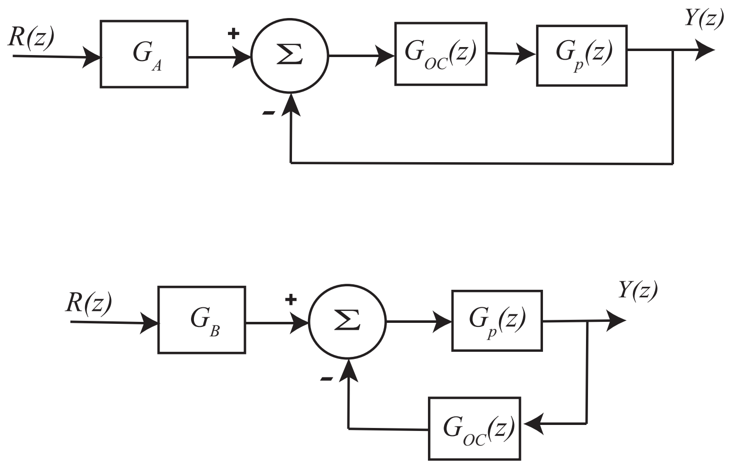

Figure 38: Two common structures used for observer based controllers. We will refer to the top structure as *Configuration A*, and the bottom structure as *Configuration B*.

It can be easily shown that the correct prefilters will be

$$\begin{aligned} G_A &= \frac{1+G_{oc}(1)G_p(1)}{G_{oc}(1)G_p(1)} \\ G_B &= \frac{1+G_{oc}(1)G_p(1)}{G_p(1)} \end{aligned}$$

We can use Matlab to determine the transfer function by using the following commands:

```
GG = G - Ke*C-H*K;
HH = Ke;
CC = K;
DD = 0;
[num,den] = ss2tf(GG,HH,CC,DD);
Gc = tf(num,den,Ts);
```

10.2 Simulations

Assume we have our usual discrete-time state variable model given by

$$\begin{bmatrix} x_1(k) \\ v_1(k) \\ x_2(k) \\ v_2(k) \end{bmatrix} = \begin{bmatrix} 0.4964 & 0.0391 & 0.5153 & 0.0095 \\ -15.5298 & 0.4361 & 14.7498 & 0.5022 \\ 0.4452 & 0.0082 & 0.3068 & 0.0358 \\ 12.7738 & 0.4326 & -21.0039 & 0.2572 \end{bmatrix} \begin{bmatrix} x_1(k) \\ v_1(k) \\ x_2(k) \\ v_2(k) \end{bmatrix} + \begin{bmatrix} 4.0181 \\ 142.8037 \\ 0.3955 \\ 29.8022 \end{bmatrix} u(k)$$

The sample interval is $T_s = 0.05$ seconds. Let's assume we want our output to be the first state (x_1), so we want

$$C = C_y = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}$$

In the following examples we will assume all of the observer poles are located at 0.1, 0.11, 0.12, and 0.13. If we place all of the system poles at 0.6, then the step response of the true observer and controller and the observer controller transfer function (configurations A and B) will produce the results shown in Figure 39. As this figure shows, both of the transfer function based designs are significantly worse than the true observer controller. The observer controller transfer function is

$$G_{oc}(z) = \frac{-0.2175z^3 + 0.5643z^2 - 0.5364z + 0.1866}{z^4 - 1.364z^3 + 0.6849z^2 + 0.5759z - 0.7885}$$

Figure 40 shows the same system with all of the state variable feedback poles at 0.1. In this instance, all three systems appear to work nearly the same. Figures 41 and 42 show the same configurations using the third state (x_2) as the system output.

10.3 Reduced Order Observers

For the reduced order observer we have the equation

$$\tilde{\eta}(k+1) = \tilde{G}\tilde{\eta}(k) + \tilde{H}x_a(k) + \tilde{F}u(k)$$

where

$$\begin{aligned} \tilde{G} &= G_{bb} - K_e G_{ab} \\ \tilde{H} &= \tilde{G}K_e + (G_{ba} - K_e G_{aa}) \\ \tilde{F} &= H_b - K_e H_a \\ \tilde{\eta}(k) &= \tilde{x}_b(k) - K_e y(k) \end{aligned}$$

Assuming again no input to the system (i.e., a regulator problem) we have

$$\begin{aligned} u(k) &= -K\tilde{x}(k) = \begin{bmatrix} K_a & K_b \end{bmatrix} \begin{bmatrix} x_a(k) \\ x_b(k) \end{bmatrix} \\ &= -K_a x_a(k) - K_b x_b(k) \end{aligned}$$

Since $y(k) = x_a(k)$ for the minimum order observer we have

$$\begin{aligned} u(k) &= -K_a y(k) - K_b \tilde{x}_b(k) \\ &= -K_a y(k) - K_b \left[\tilde{\eta}(k) + K_e y(k)\right] \\ &= -\left[K_a + K_b K_e\right] y(k) - K_b \tilde{\eta}(k) \end{aligned}$$

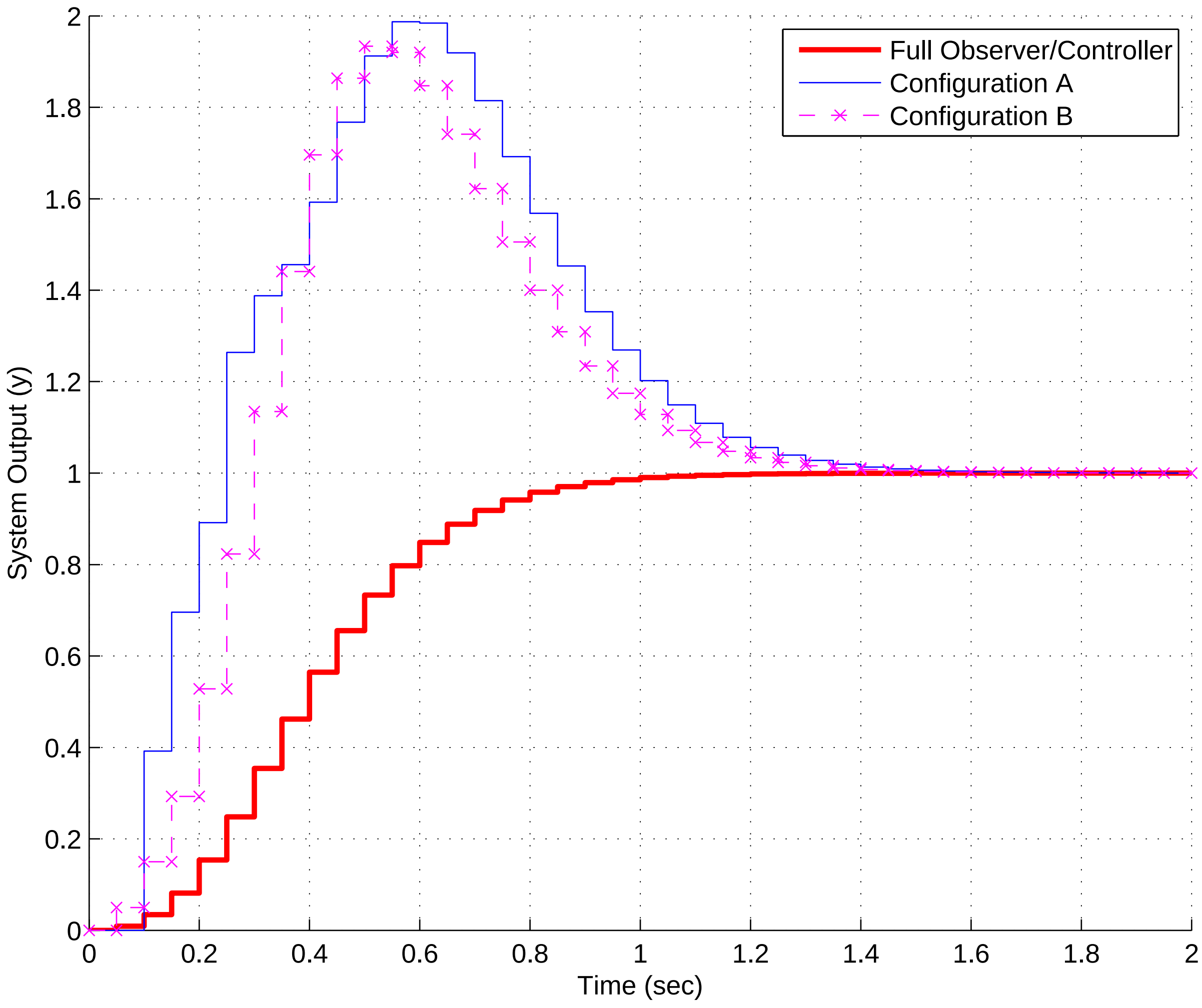

Figure 39: Closed loop step response of fourth order system with true observer controller and with the observer controller transfer function. The output is the first state (x_1). All of the system poles are at 0.6 and the observer poles are at 0.1, 0.11, 0.12, and 0.13.

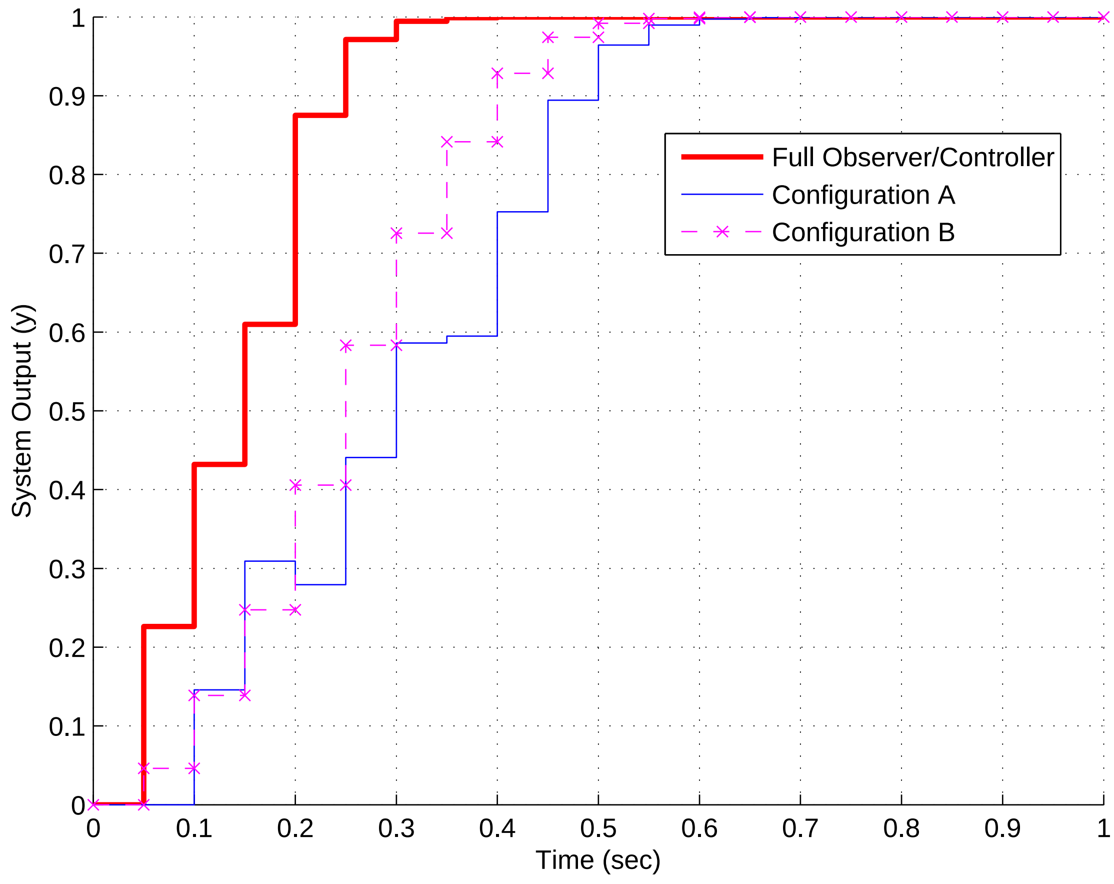

Figure 40: Closed loop step response of fourth order system with true observer controller and with the observer controller transfer function. The output is the first state (x_1). All of the system poles are at 0.1 and the observer poles are at 0.1, 0.11, 0.12, and 0.13.

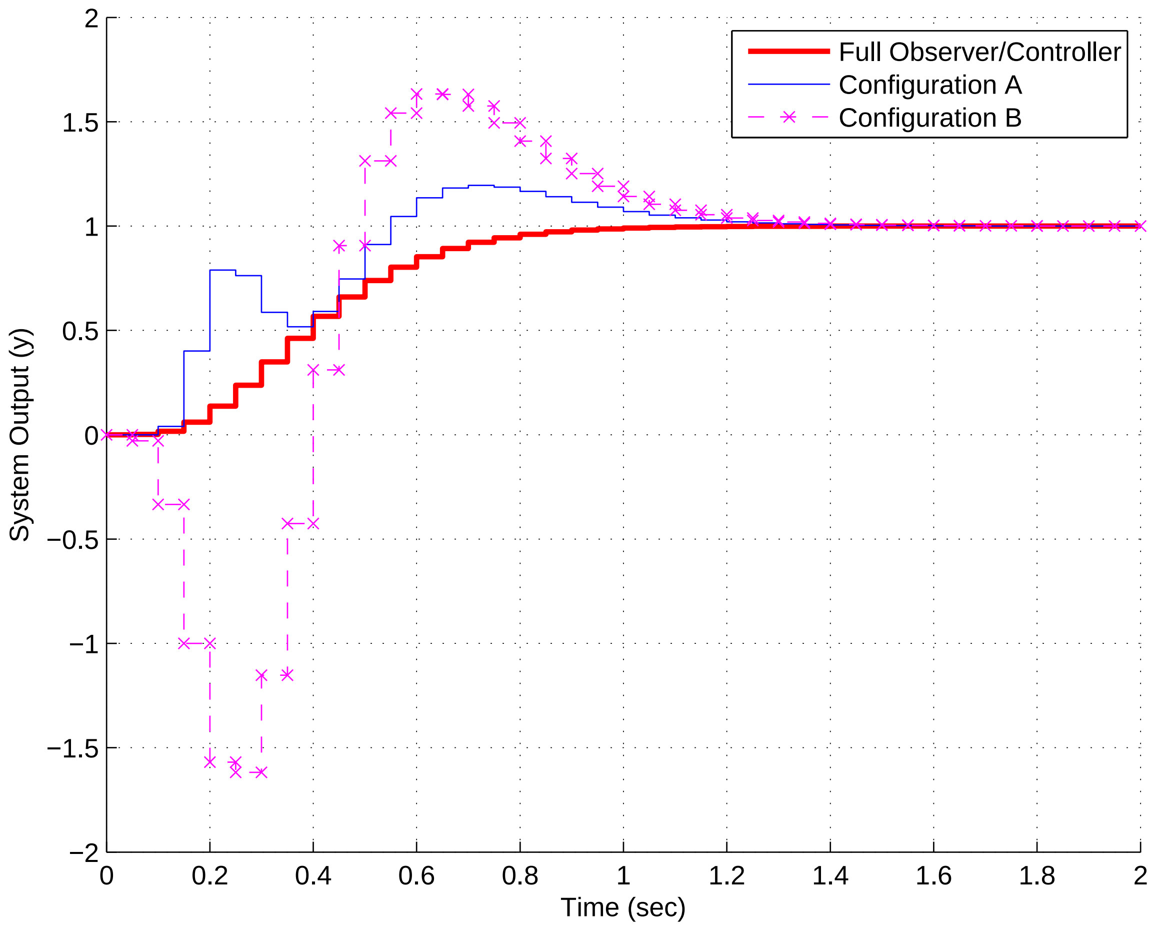

Figure 41: Closed loop step response of fourth order system with true observer controller and with the observer controller transfer function. The output is the third state (x_2). All of the system poles are at 0.1 and the observer poles are at 0.1, 0.11, 0.12, and 0.13.

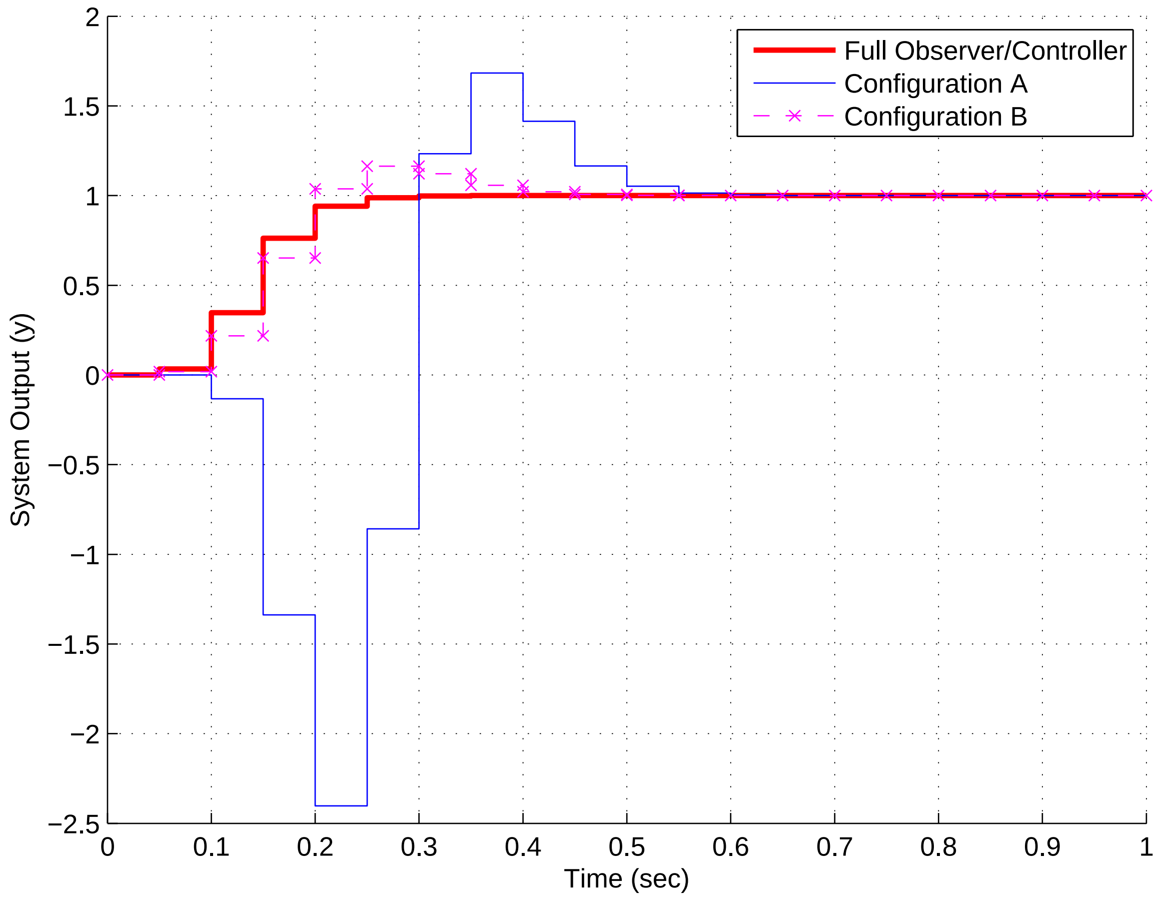

Figure 42: Closed loop step response of fourth order system with true observer controller and with the observer controller transfer function. The output is the third state (x_2). All of the system poles are at 0.1 and the observer poles are at 0.1, 0.11, 0.12, and 0.13.

Substituting this relationship into the relationship for $\tilde{\eta}(k+1)$ we have

$$\begin{aligned}\tilde{\eta}(k+1) &= \tilde{G}\tilde{\eta}(k) + \tilde{H}y(k) + \tilde{F}\left[-\left(K_a + K_bK_e\right)y(k) - K_b\tilde{\eta}(k)\right] \\ &= \left[\tilde{G} - \tilde{F}K_b\right]\tilde{\eta}(k) + \left[\tilde{H} - \tilde{F}\left(K_a + K_bK_e\right)\right]y(k)\end{aligned}$$

Now taking z-transforms we have

$$\begin{aligned}z\tilde{\eta}(z) &= \left[\tilde{G} - \tilde{F}K_b\right]\tilde{\eta}(z) + \left[\tilde{H} - \tilde{F}\left(K_a + K_bK_e\right)\right]Y(z) \\ \tilde{\eta}(z) &= \left[zI - \tilde{G} + \tilde{F}K_b\right]^{-1}\left[\tilde{H} - \tilde{F}\left(K_a + K_bK_e\right)\right]Y(z)\end{aligned}$$

and

$$\begin{aligned}U(z) &= -\left[K_a + K_bK_e\right]Y(z) - K_b\tilde{\eta}(z) \\ &= -\left[K_a + K_bK_e\right]Y(z) - K_b\left[zI - \tilde{G} + \tilde{F}K_b\right]^{-1}\left[\tilde{H} - \tilde{F}\left(K_a + K_bK_e\right)\right]Y(z)\end{aligned}$$

Finally, the transfer function we want is given by

$$\frac{U(z)}{-Y(z)} = \left[K_a + K_bK_e\right] + K_b\left[zI - \tilde{G} + \tilde{F}K_b\right]^{-1}\left[\tilde{H} - \tilde{F}\left(K_a + K_bK_e\right)\right]$$

This transfer function observer based controller can then be implemented in exactly the same way as for the full order observer based controller. Again, there is no guarantee that the resulting controller/regulator will be stable or even work very well. We can use Matlab to determine the transfer function as follows:

```
GG = Gtilde -Ftilde*Kb;
HH = Htilde - Ftilde*(Ka+Kb*Ke);
CC = Kb;
DD = Ka + Kb*Ke
[num,den] = ss2tf(GG,HH,CC,DD);
Gc = tf(num,den,Ts);
```

Recall that for the minimum order observer we must have the known state in the first position. Thus we will always have $C = C_y$ with a 1 in the first position and all other positions 0. The order of the states may need to be rearranged, as usual when using a minimum order observer.

Figure 43 displays the results for our usual fourth order system with all of the system poles at 0.6 and the observer poles at 0.1, 0.11, and 0.12. The output (and known state) is the first state (x_1). The observer controller transfer function is

$$G_{oc}(z) = \frac{-0.1786z^3 + 0.02658z^2 + 0.3013z - 0.2486}{z^3 - 0.5158z^2 + 0.9837z + 1.05}$$

In this instance, the results from the true minimum order obseerver/state feedback controller and the transfer function implementations are very nearly the same. Figure 44 displays the results when the system poles are all moved to 0.1. In this case clearly configuration B is superior and nearly identical to the true minimum observer/state feedback system.

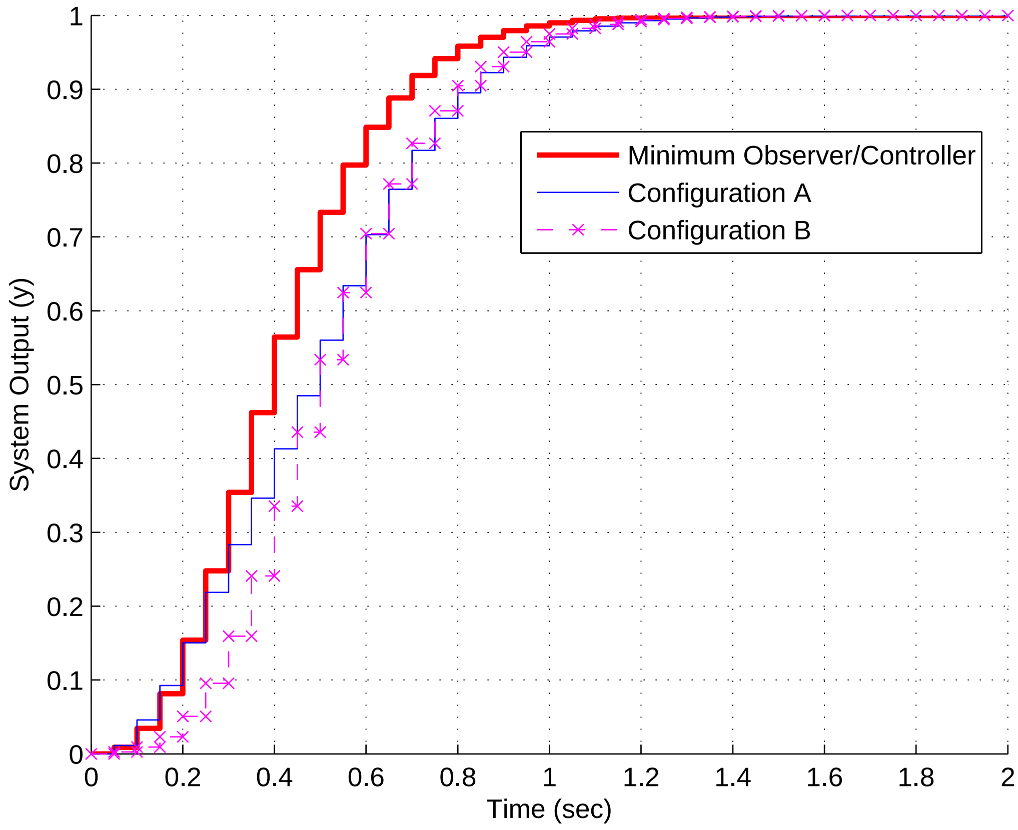

Figure 43: Closed loop step response of fourth order system with true minimum order observer controller and with the observer controller transfer function. The output is the first state (x_1). All of the system poles are at 0.6 and the observer poles are at 0.1, 0.11, and 0.12.

Figure 45 displays the results for the system where the third state (x_2) is the known/output state, all of the system poles are at 0.6, and the observer poles are at 0.1, 0.11, and 0.12. The observer controller transfer function is

$$G_{oc}(z) = \frac{0.1465z^3 - 0.01845z^2 - 0.2748z + 0.2296}{z^3 - 1.292z^2 - 0.9488z - 0.09644}$$

In this instance the reordering of the state variables is necessary and we still have $C = C_y = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}$. In this instance nearly all three configurations perform identically. Finally, Figure 46 displays the results when all of the system poles are moved to 0.1. Again, the results from configuration B more closely match those of the complete minimum order observer/state variable feedback system, but with much simpler implementation.

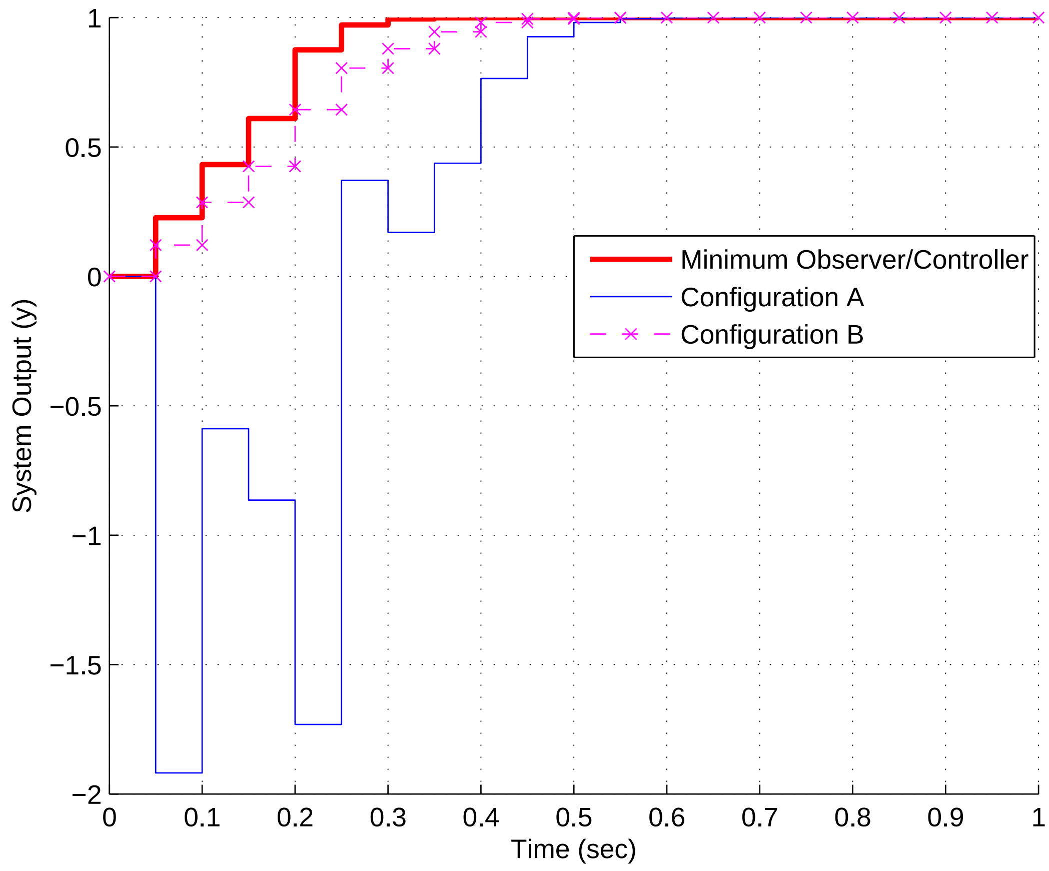

Figure 44: Closed loop step response of fourth order system with true minimum order observer controller and with the observer controller transfer function. The output is the first state (x_1). All of the system poles are at 0.1 and the observer poles are at 0.1, 0.11, and 0.12.

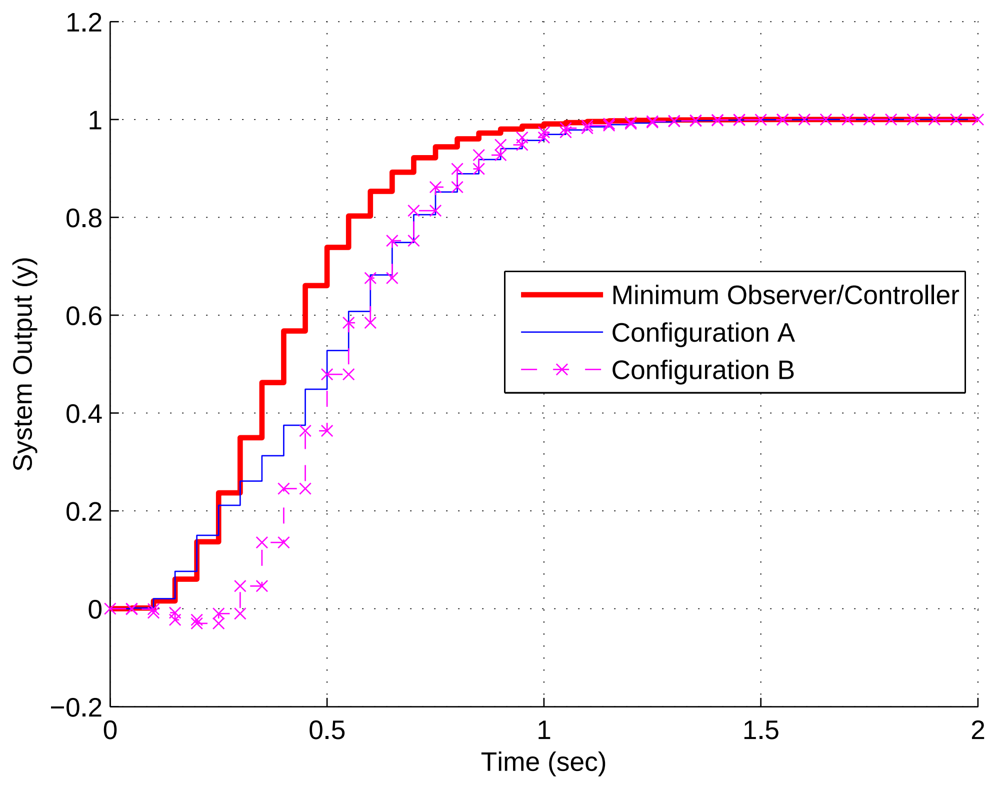

Figure 45: Closed loop step response of fourth order system with true minimum order observer controller and with the observer controller transfer function. The output is the third state (x_2). All of the system poles are at 0.1 and the observer poles are at 0.1, 0.11, and 0.12.

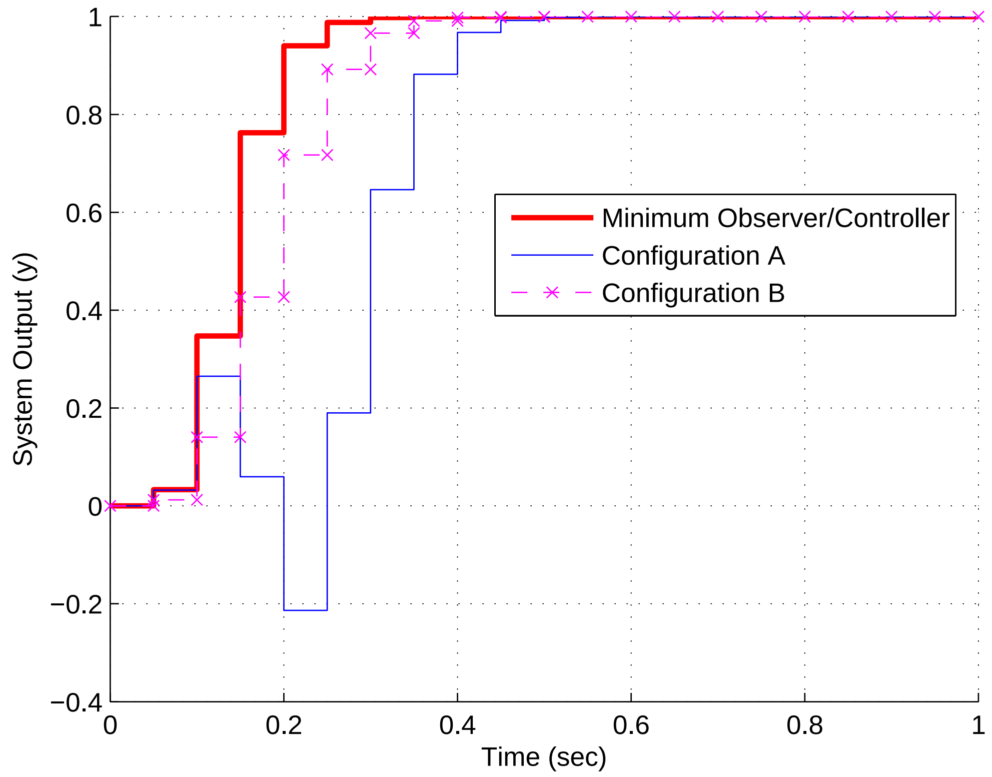

Figure 46: Closed loop step response of fourth order system with true minimum order observer controller and with the observer controller transfer function. The output is the third state (x_2). All of the system poles are at 0.1 and the observer poles are at 0.1, 0.11, and 0.12.

11 Linear Quadratic Optimal Control

In this final chapter, we will focus on solving a linear quadratic tracking problem. Assume we have a known input signal $r(k)$ that we would like the output of our system to follow. We assume that the system operates over a finite number of time steps, from $k = 0$ to $k = N - 1$, and that our system can be modeled as a linear time-invariant state variable system,

$$\begin{aligned} x(k+1) &= Gx(k) + Hu(k) \\ y(k) &= Cx(k) \end{aligned}$$

For this type of problem there are usually three things we are interested in:

1. How closely does $x(k)$ match $r(k)$ over $k = 0, \dots, N-1$?
2. How close is $X(N)$ to $R(N)$, i.e. does our system end at the same place the reference input ends?
3. How much control effort is required?

One way to formulate this problem is as follows:

$$\begin{aligned} \min_{u(k)} J &= \frac{1}{2}\left[y(N) - r(N)\right]^T Q_f \left[y(N) - r(N)\right] \\ &+ \frac{1}{2}\sum_{k=0}^{k=N-1} \left\{\left[y(k) - r(k)\right]^T Q \left[y(k) - r(k)\right] + u(k)^T R u(k)\right\} \end{aligned}$$

Subject to the constraints

$$\begin{aligned} x(k+1) &= Gx(k) + Hu(k) \\ y(k) &= Cx(k) \end{aligned}$$

We assume $x(0) = x_0$ and the signal we want our system to follow, $r(k)$, are known. Finally, we assume $Q = Q^T$, $Q_f = Q_f^T$, $R = R^T$, and R^{-1} exists. These three matrices impose penalties on

- the control effort (R)
- not matching the reference signal at the final time (Q_f)
- not matching the reference signal during the transients (Q)

For this formulation, the system is *linear* and the penalties are *quadratic*. This problem is said to have *soft terminal constraints* since we do not force $y(N) = r(N)$. We can try to make them equal by imposing a penalty, but it is not absolutely required that they be equal. If we require that $y(N) = R(N)$ as part of the problem, then we have a problem with *hard terminal constraints*, and it needs to be solved a bit differently than the method we will follow.

11.1 Euler Lagrange Equations

With our background in matrix calculus and Lagrange multipliers, let's return to our original problem. We need to adjoin the constraints to our original cost function. Hence we need to now find the control sequence $u(k)$ to minimize $\bar{J}$, where $\bar{J}$ is the same cost function as J but has the constraints adjoined. Specifically, we can write $\bar{J}$ as

$$\begin{aligned}\bar{J} &= \frac{1}{2}\left[y(N)-r(N)\right]^T Q_f \left[y(N)-r(N)\right] \\ &+\frac{1}{2}\sum_{k=0}^{k=N-1}\Big\{ \left[y(k)-r(k)\right]^T Q\left[y(k)-r(k)\right] + u(k)^T R u(k) \\ &\qquad +\lambda(k+1)^T\left[Gx(k)+Hu(k)-x(k+1)\right]\Big\}\end{aligned}$$

Notice that in this case the Lagrange multiplier is a time varying vector. At this point is may seem odd to write it as $\lambda(k+1)$ instead of $\lambda(k)$, but it will turn out to be a bit simpler in the future. Rather than having the two variables $y(k)$ and $x(k)$, let's rewrite our expression for $\bar{J}$ in terms of $x(k)$,

$$\begin{aligned}\bar{J} &= \frac{1}{2}\left[Cx(N)(N)-r(N)\right]^T Q_f \left[Cx(N)-r(N)\right] \\ &+\frac{1}{2}\sum_{k=0}^{k=N-1}\Big\{ \left[Cx(k)-r(k)\right]^T Q\left[Cx(k)-r(k)\right] + u(k)^T R u(k) \\ &\qquad +\lambda(k+1)^T\left[Gx(k)+Hu(k)-x(k+1)\right]\Big\}\end{aligned}$$

Next we define the discrete-time *Hamiltonian* as

$$\mathcal{H} = \frac{1}{2}\Big\{\left[Cx(k)-r(k)\right]^T Q\left[Cx(k)-r(k)\right] + u(k)^T R u(k)\Big\} + \lambda(k+1)^T\left[Gx(k)+Hu(k)\right]$$

and define

$$\Phi\left(x(N)\right) = \frac{1}{2}\left[Cx(N)-r(N)\right]^T Q_f\left[Cx(N)-r(N)\right]$$

In terms of these new variables we have

$$\begin{aligned}\bar{J} &= \Phi\left(x(N)\right) + \sum_{k=0}^{N-1}\Big\{\mathcal{H}(k) - \lambda(k+1)^T x(k+1)\Big\} \\ &= \Phi\left(x(N)\right) + \sum_{k=0}^{N-1}\mathcal{H}(k) - \sum_{k=1}^{N}\lambda(k)^T x(k) + \underbrace{\lambda(0)^T x(0) - \lambda(0)^T x(0)}_{adding\ zero} \\ &= \Phi\left(x(N)\right) + \sum_{k=0}^{N-1}\Big[\mathcal{H}(k) - \lambda(k)^T x(k)\Big] - \lambda(N)^T x(N) + \lambda^T(0)x(0)\end{aligned}$$

Now let's consider differential changes in $\bar{J}$ due to differential changes in $u(k)$ and $x(k)$,

$$\begin{aligned}d\bar{J} &= \frac{d\Phi\left(x(N)\right)}{dx(N)}dx(N) - \lambda^T(N)dx(N) + \lambda^T(0)dx(0) \\ &= \quad + \sum_{k=0}^{k=N-1}\left\{\left[\frac{\partial\mathcal{H}(k)}{\partial x(k)} - \lambda^T(k)\right]dx(k) + \frac{\partial\mathcal{H}(k)}{\partial u(k)}du(k)\right\}\end{aligned}$$

At an optimum we need $d\bar{J} = 0$. To avoid having to determine the $dx(k)$ caused by the incremental changes in $du(k)$, we will choose the Lagrange multipliers as

$$\lambda^T(k) = \frac{\partial \mathcal{H}(k)}{\partial x(k)}$$

Next, since we don't know what $du(k)$ is, we will choose

$$\frac{\partial \mathcal{H}(k)}{\partial u(k)} = 0$$

We know the initial condition $x(0)$ is a constant, so $dx(0) = 0$. Finally we are left with the boundary condition

$$\lambda^T(N) = \frac{d\Phi\left(x(N)\right)}{dx(N)}$$

In summary, to find a control sequence $u(k)$ that produces a minimum value of J, we must solve the following discrete-time *Euler-Lagrange (EL)* equations

$$\begin{aligned}
\frac{\partial \mathcal{H}(k)}{\partial x(k)} &= \lambda^T(k) \\
\frac{\partial \mathcal{H}(k)}{\partial u(k)} &= 0 \\
\lambda^T(N) &= \frac{d\Phi\left(x(N)\right)}{dx(N)} \\
x(0) &= x_0 \quad \text{a known initial condition}
\end{aligned}$$

where the discrete-time *Hamiltonian* is

$$\mathcal{H} = \frac{1}{2}\Big\{\left[Cx(k) - r(k)\right]^T Q\left[Cx(k) - r(k)\right] + u(k)^T R u(k)\Big\} + \lambda(k+1)^T\left[Gx(k) + Hu(k)\right]$$

and

$$\Phi\left(x(N)\right) = \frac{1}{2}\left[Cx(N) - r(N)\right]^T Q_f \left[Cx(N) - r(N)\right]$$

This is known as a *two point boundary value problem*, since we have a boundary value at the initial time ($x(0)$) and a boundary value at the end time ($\lambda(N)$). In general these are very difficult to solve analytically, and a computer solution is required. For our particular problem we will be able to general a recursive relationship that must be solved to determine the solution.

11.2 Derivation of Recursive Relationships

At this point, we need to use the Euler-Lagrange equations to determine the solution to our problem. First we need to expand out our expression for $\mathcal{H}(k)$,

$$\begin{aligned}
\mathcal{H} &= \frac{1}{2}\Big\{\left[Cx(k) - r(k)\right]^T Q\left[Cx(k) - r(k)\right] + u(k)^T R u(k)\Big\} + \lambda(k+1)^T\left[Gx(k) + Hu(k)\right] \\
&= \frac{1}{2}x^T(k)C^T Q C x(k) - x^T(k)C^T Q r(k) + \lambda^T(k+1)Gx(k) + \frac{1}{2}u^T(k)Ru(k) \\
&\quad + \lambda^T(k+1)Hu(k) + r^T(k)Qr(k)
\end{aligned}$$

Then

$$\frac{\partial \mathcal{H}(k)}{\partial x(k)} = x^T(k)C^TQC - r^T(k)QC + \lambda^T(k+1)G = \lambda^T(k)$$

or

$$\lambda(k) = C^TQCx(k) - C^TQr(k) + G^T\lambda(k+1)$$

At this point we have a recursive relationship between the λ's. We then have the boundary condition

$$\begin{aligned}
\lambda(N)^T &= \frac{d\Phi\left(x(N)\right)}{dx(N)} \\
&= \frac{d}{dx(N)}\left\{\frac{1}{2}\left[x(N)^TC^T - r(N)^T\right]Q_f\left[Cx(N) - r(N)\right]\right\} \\
&= \frac{d}{dx(N)}\left[\frac{1}{2}x(N)^TC^TQ_fCx(N) - r(N)^TQ_fCx(N) + \frac{1}{2}Q_fr(N)\right] \\
&= x^T(N)C^TQ_fC - r(N)^TQ_fC
\end{aligned}$$

or

$$\lambda(N) = C^TQ_fCx(N) - C^TQ_fr(N)$$

Our last condition is given by the remaining Euler-Lagrange equation,

$$0 = \frac{\partial \mathcal{H}(k)}{\partial u(k)} = u^T(k)R + \lambda^T(k+1)H$$

or

$$Ru(k) + H^T\lambda(k+1) = 0$$

which leads to

$$u(k) = -R^{-1}H^T\lambda(k+1)$$

At this point we do not have particularly useful results, so we are going to have to do some algebra and apply a trick or two. We will derive some equations that seem to be getting worse than when we started, but then they will simplify and lead to useful recurrence relationships. We can write the state equations as

$$x(k+1) = Gx(k) + Hu(k) = Gx(k) + H\left\{R^{-1}H^T\lambda(k+1)\right\} = Gx(k) - HR^{-1}H^T\lambda(k+1)$$

Next, let's assume $\lambda(k) = S(k)x(k) - v(k)$ for some functions $S(k)$ and $v(k)$ to be determined. We then have

$$x(k+1) = Gx(k) - HR^{-1}H^T\left\{S(k+1)x(k+1) - v(k+1)\right\}$$

or

$$\left[I + HR^{-1}H^TS(k+1)\right]x(k+1) = Gx(k) + HR^{-1}H^Tv(k+1)$$

If we define

$$\Delta(k) = \left[I + HR^{-1}H^TS(k+1)\right]^{-1}$$

then we have

$$x(k+1) = \Delta(k)Gx(k) + \Delta(k)HR^{-1}H^Tv(k+1)$$

We also have

$$\lambda(k) = S(k)x(k) - v(k) = C^TQCx(k) - C^TQr(k) + G^T\left\{S(k+1)x(k+1) - v(k+1)\right\}$$

or

$$S(k)x(k) - v(k) = C^TQCx(k) + G^TS(k+1)\left\{\Delta(k)Gx(k) + \Delta(k)HR^{-1}H^Tv(k+1)\right\} - G^Tv(k+1) - C^TQr(k)$$

Rearranging this expression we get

$$\begin{aligned}&\left\{-S(k) + G^TS(k+1)\Delta(k)G + C^TQC\right\}x(k)\\ &\qquad + \left\{v(k) + G^TS(k+1)\Delta(k)HR^{-1}H^Tv(k+1) - G^Tv(k+1) - C^TQr(k)\right\} = 0\end{aligned}$$

This expression must be true for all $x(k)$, which means each of the terms in brackets must be zero, so we have

$$\begin{aligned}S(k) &= G^TS(k+1)\Delta(k)G + C^TQC\\ v(k) &= \left[-G^TS(k+1)\Delta(k)HR^{-1}H^T + G^T\right]v(k+1) + C^TQr(k)\\ \Delta(k) &= \left[I + HR^{-1}H^TS(k+1)\right]^{-1}\end{aligned}$$

These are the recursive equations we need. If we know $S(k+1)$ and $v(k+1)$ we can determine $S(k)$ and $v(k)$. However, we need a starting condition. To get the starting condition, we use

$$\Lambda(N) = S(N)x(N) - v(N) = C^TQ_fCx(N) - C^TQ_fr(N)$$

so we have the starting conditions

$$\begin{aligned}S(N) &= C^TQ_fC\\ V(N) &= C^TQ_fr(N)\end{aligned}$$

Finally, we want to put our results in a control law we can use. We have

$$\begin{aligned}u(k) &= -R^{-1}H^T\lambda(k+1)\\ Ru(k) &= -H^T\lambda(k+1)\\ &= -H^T\left\{S(k+1)x(k+1) - v(k+1)\right\}\\ &= -H^T\left\{S(k+1)\left[Gx(k) + Hu(k)\right] - v(k+1)\right\}\\ &= -H^TS(k+1)Gx(k) - H^TS(k+1)Hu(k) + H^Tv(k+1)\\ \left[R + H^TS(k+1)H\right]u(k) &= -H^TS(k+1)Gx(k) + H^Tv(k+1)\\ u(k) &= -\left[R + H^TS(k+1)H\right]^{-1}H^TS(k+1)Gx(k)\\ &\qquad + \left[R + H^TS(k+1)H\right]^{-1}H^Tv(k+1)\\ &= -K(k)x(k) + u_f(k)\end{aligned}$$

The *time-varying feedback matrix* $K(k)$ is given by

$$K(k) = \left[R + H^TS(k+1)H\right]^{-1}H^TS(k+1)G$$

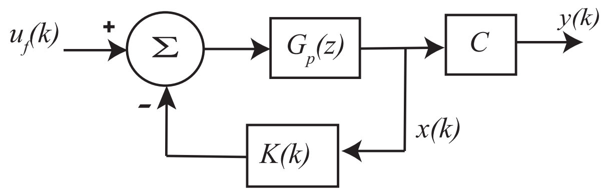

Figure 47: Structure for use with the LQR optimal system. Note that the signal we are trying to track is known in advance.

and the *feedforward control* $u_f(k)$ is given by

$$u_f(k) = \left[R + H^T S(k+1)H\right]^{-1} H^T v(k+1)$$

The basic structure for this system is shown in Figure 11.2.

Before giving some examples in the next section, we should summarize. The signal we are trying to track, $r(k)$ is known in advance from $k = 0$ to $k = N$. From this information, and knowledge of C and Q_f, we can compute (in advance) the values for $S(N)$ and $V(N)$. From these values, we can use the recursive relationships we have derived to compute $s(k)$ and $v(k)$ for all required k. Finally we can compute $K(k)$ and $u_f(k)$ and implement the control law. Normally you guess some values for R, Q, and Q_f and then simulate the system to get the desired response using an intelligent guess and check sort of method. *At this point you should be aware that there is on very serious potential problem with what we have done. This method assumes our model of the plant is very accurate and there is no noise or external disturbances in the system.*

11.3 Examples

In this section we present some results of the optimal tracker algorithm for the discrete-time system modeled by the equation

$$\begin{aligned}
\begin{bmatrix} x_1(k) \\ v_1(k) \\ x_2(k) \\ v_2(k) \end{bmatrix} &= \begin{bmatrix} 0.4964 & 0.0391 & 0.5153 & 0.0095 \\ -15.5298 & 0.4361 & 14.7498 & 0.5022 \\ 0.4452 & 0.0082 & 0.3068 & 0.0358 \\ 12.7738 & 0.4326 & -21.0039 & 0.2572 \end{bmatrix} \begin{bmatrix} x_1(k) \\ v_1(k) \\ x_2(k) \\ v_2(k) \end{bmatrix} + \begin{bmatrix} 4.0181 \\ 142.8037 \\ 0.3955 \\ 29.8022 \end{bmatrix} u(k) \\
y(k) &= \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1(k) \\ v_1(k) \\ x_2(k) \\ v_2(k) \end{bmatrix}
\end{aligned}$$

The sample interval is $T_s = 0.05$ seconds and we are trying to track the input signal from $t = 0$ until $t = 2.0$ seconds. Again, we assume this input signal is known in advance.

Figure 48 shows the values of each state as well as the control signal (u) and feedforward signal (u_f) for an input of $r(k) = 1.0$ for $Q = 0.001$, $Q_f = 0.001$ and $R = 0.1$. Figure 49 shows the

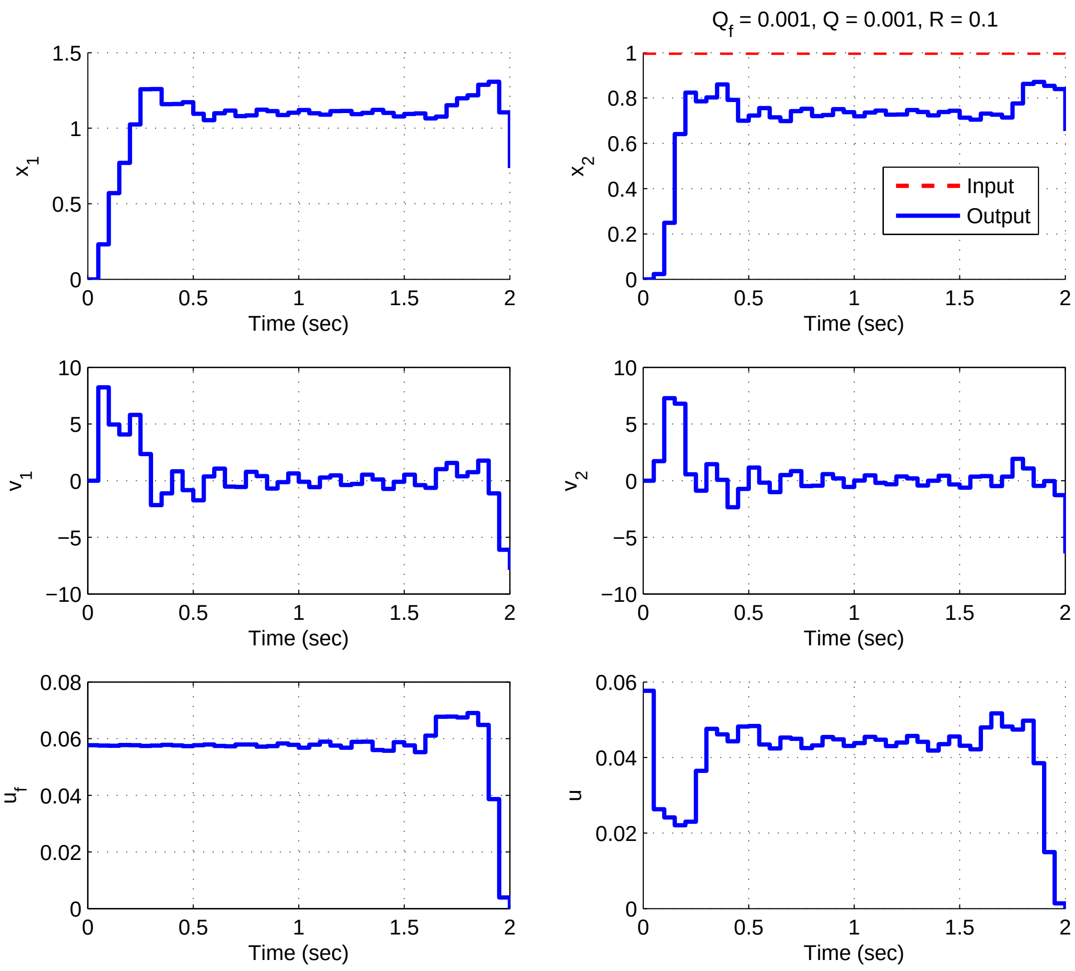

Figure 48: Pathetic tracking of a unit step signal.

corresponding values of the feedback gains $(K(k))$. Clearly we are not tracking the input signal very well, but then again we weren't trying yet either!

We next adjust the weights so the output of the system matches the reference signal at the end of the interval. If we only want to match at the end of the interval, we will usually just adjust the value of Q_f. Figure 50 shows the values of each state as well as the control signal (u) and feedforward signal (u_f) for our input of $r(k) = 1.0$ for $Q = 0.001$, $Q_f = 0.01$ and $R = 0.1$. Figure 51 shows the corresponding values of the feedback gains $(K(k))$. Clearly we are doing a much better job of matching the value of the reference signal at the end of the interval. Note that both the feedforward control signal (u_f) and the feedback gains are fairly constant until near the end of the interval.

Next we try and track the reference signal across the whole interval. We will start by adjusting the value of Q, and returning Q_f to its original value. Figure 52 shows the values of each state

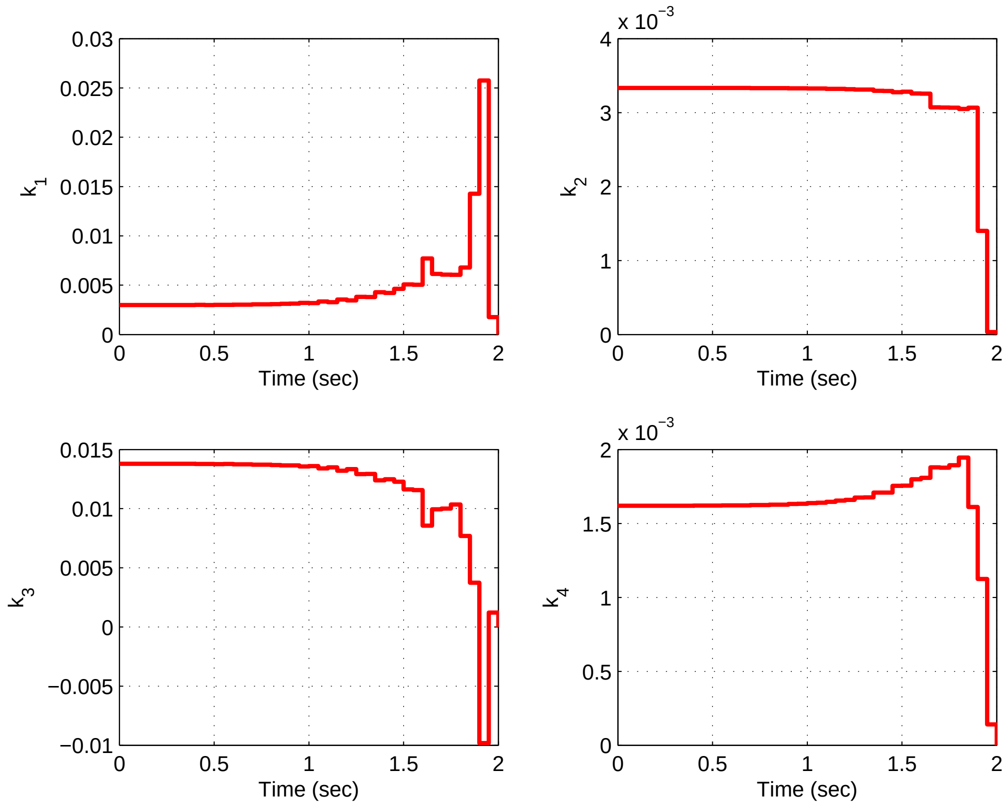

Figure 49: Time-varying feedback gains for the pathetic tracking of a unit step signal.

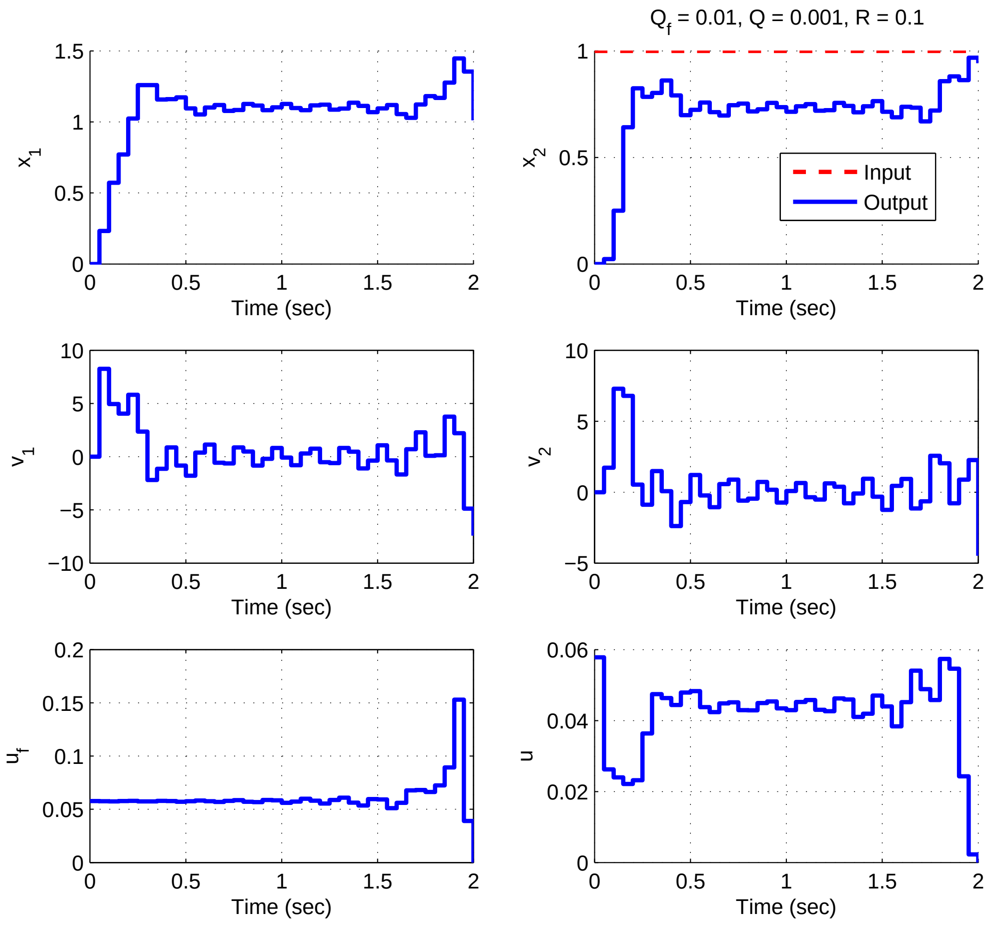

Figure 50: Matching the value of the reference signal at the end of the interval.

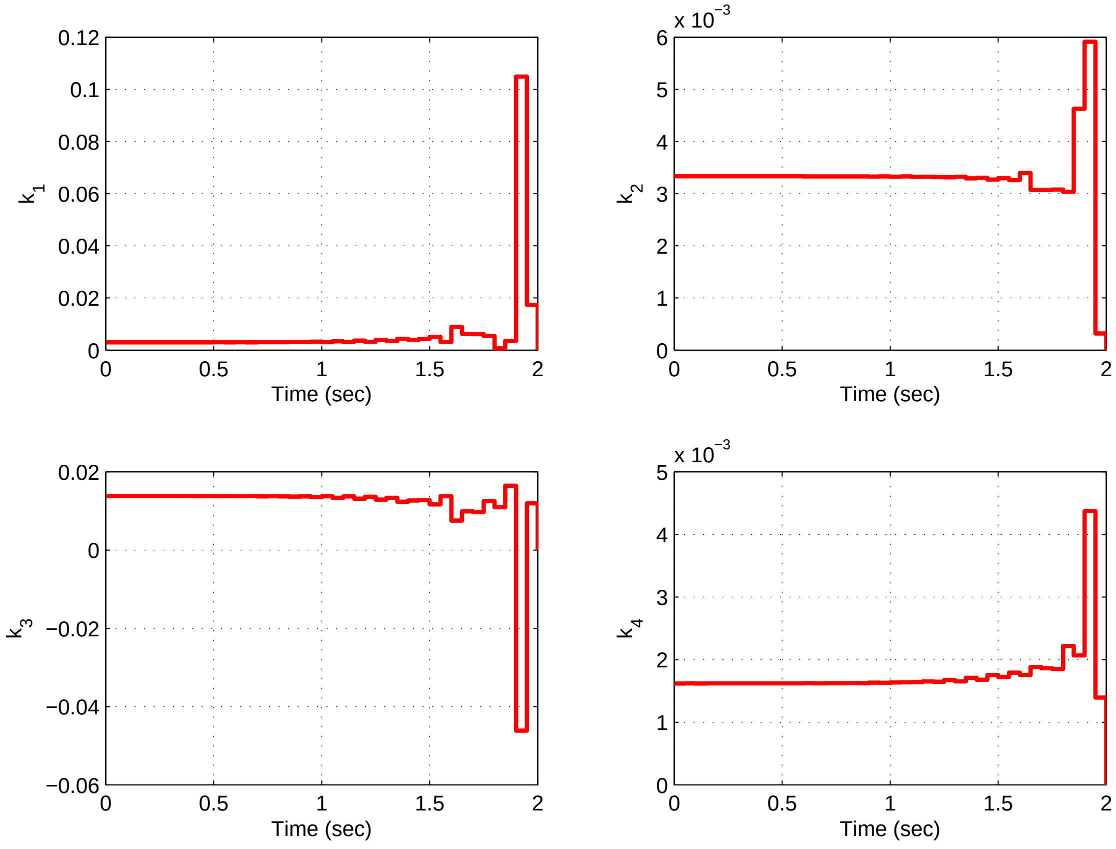

Figure 51: Time-varying feedback gains for matching the value of the reference signal at the end of the interval.

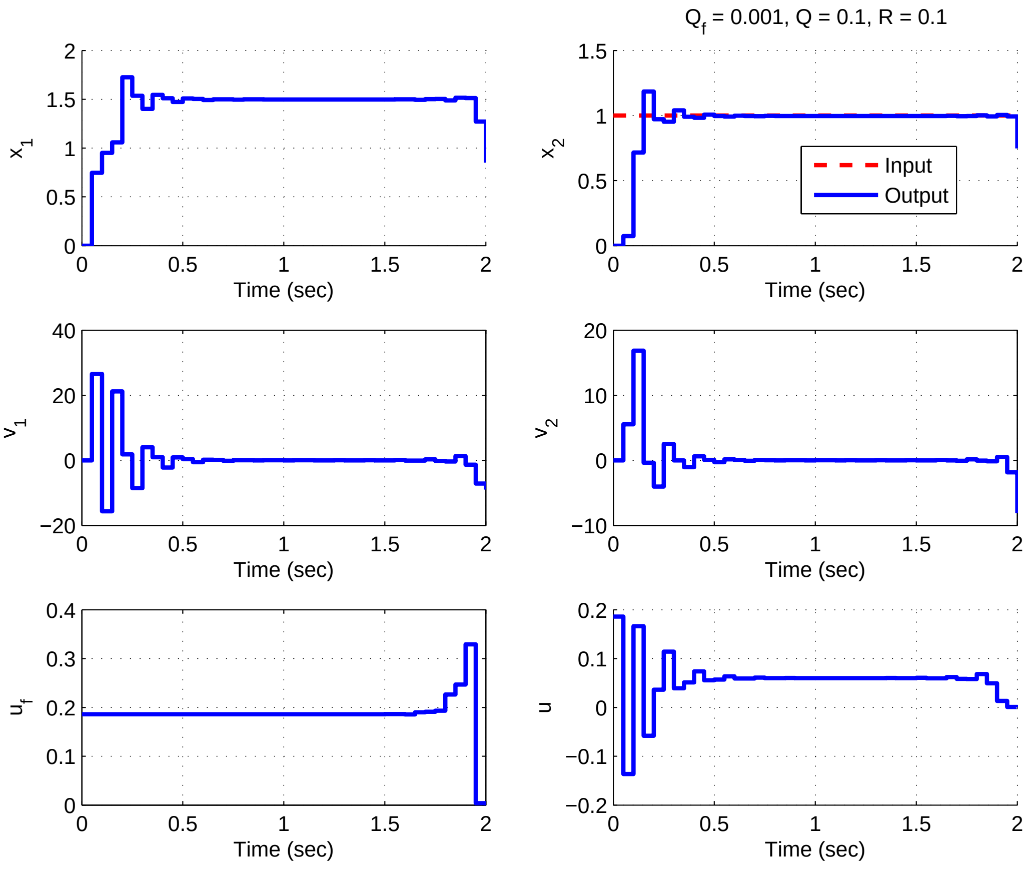

Figure 52: Tracking of a unit step signal across much of the interval.

and the control signals, and Figure 53 shows the corresponding values of the feedback gains for $Q = 0.1$, $Q_f = 0.001$ and $R = 0.1$. Figure 53 shows the corresponding values of the feedback gains $(K(k))$. Clearly we are doing a much better job of matching the value of the reference signal throughout the interval, however we are off a bit at the end of the interval. Note that both the feedforward control signal (u_f) and the feedback gains are fairly constant until near the end of the interval.

Figures 54 and 55 shows the results after modifying Q_f so we track the reference signal across the interval and match it fairly well at the end of the interval. Figures 56 shows the results when we use a constant feedback gain matrix. This is much easier to implement than a time-varying feedback gain matrix, and the performance for this application is nearly the same except at the end of the interval. In fact, if you were to use the *dlqry* command in Matlab with the same weights we used,

```
dlqry(H,G,C,D,Q,R)
```

Matlab produces the constant feedback gains 0.1096, 0.0083, -0.0384, and -0.0043, which are nearly the same (within numerical accuracy) feedback gains we obtained. Note that if you know

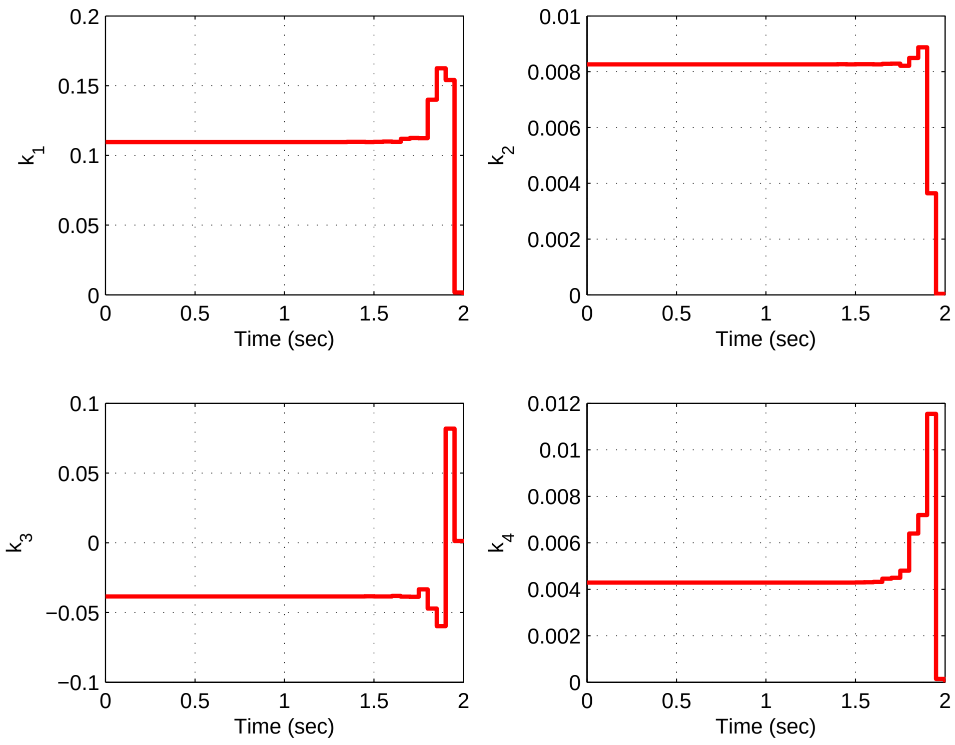

Figure 53: Time-varying feedback gains for tracking a unit step signal across much of the interval.

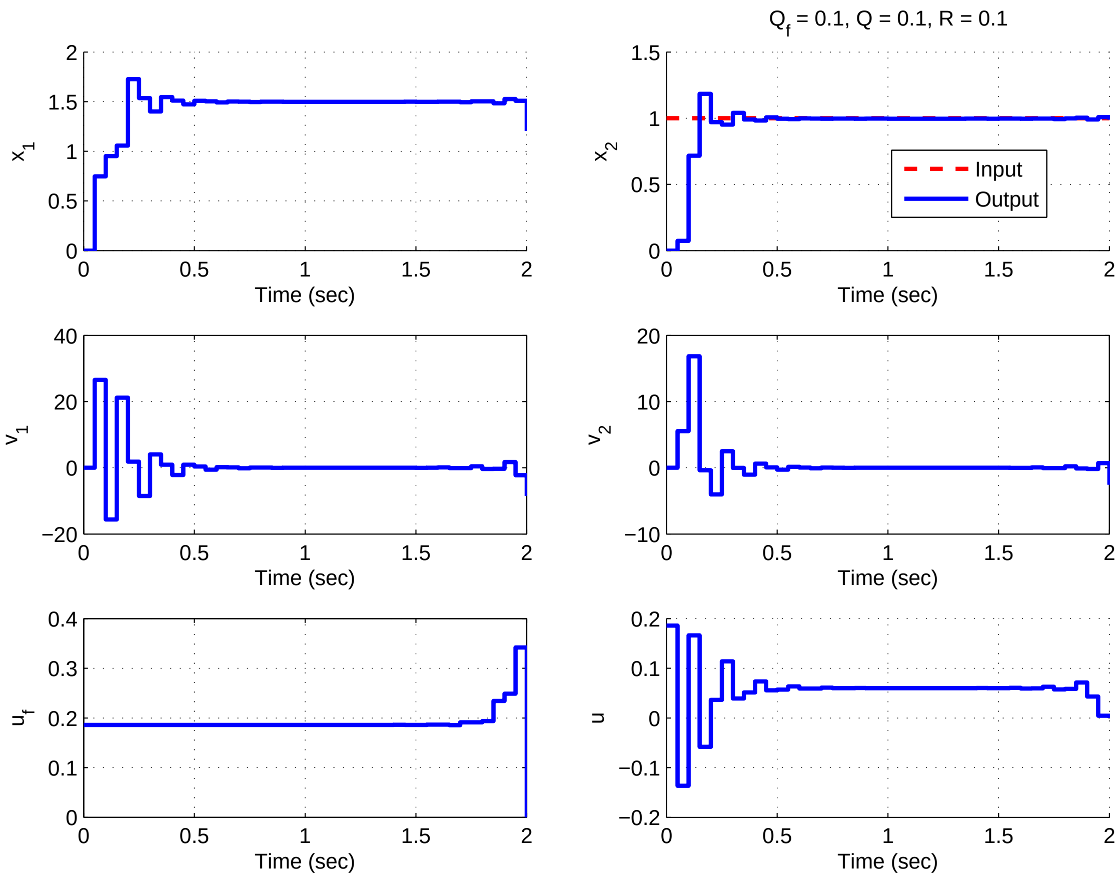

Figure 54: Tracking of a unit step signal across the interval and matching at the end of the interval.

you only want to use the constant feedback gains, you can make some simplifications to our derivation. In addition, Matlab does not allow for the final weight Q_f that our derivation does.

Finally, if we look at a sinusoidal input, as shown in Figure 57 (using constant feedback gains). What is most interesting and important to note about this results is to notice that both the feedforward control signal (u_f) and the control signal (u) *lead* the signal we want to track. Nearly all real systems will have a delay in them and not be able to accurately track a reference input without a delay. We are able to do this only because we know the input signal $(r(k))$ in advance.

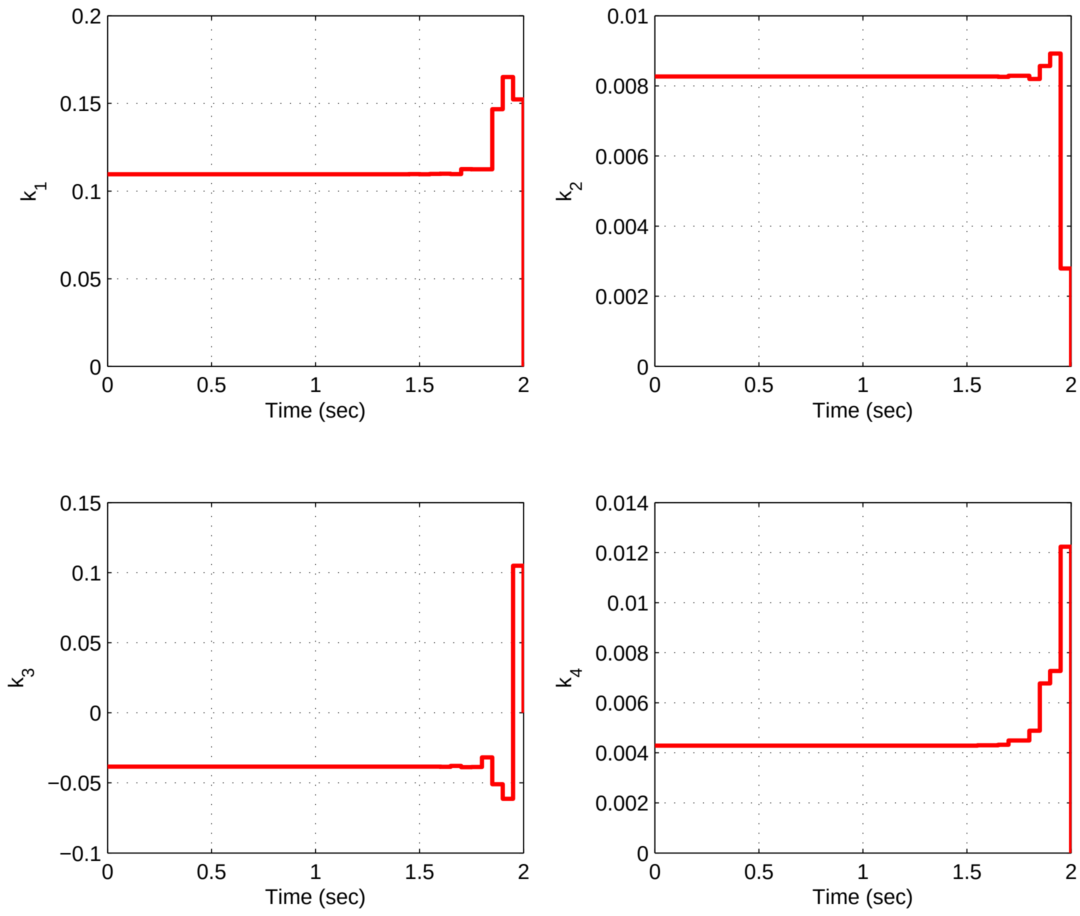

Figure 55: Time-varying feedback gains for tracking a unit step signal across the interval and matching at the end of the interval.

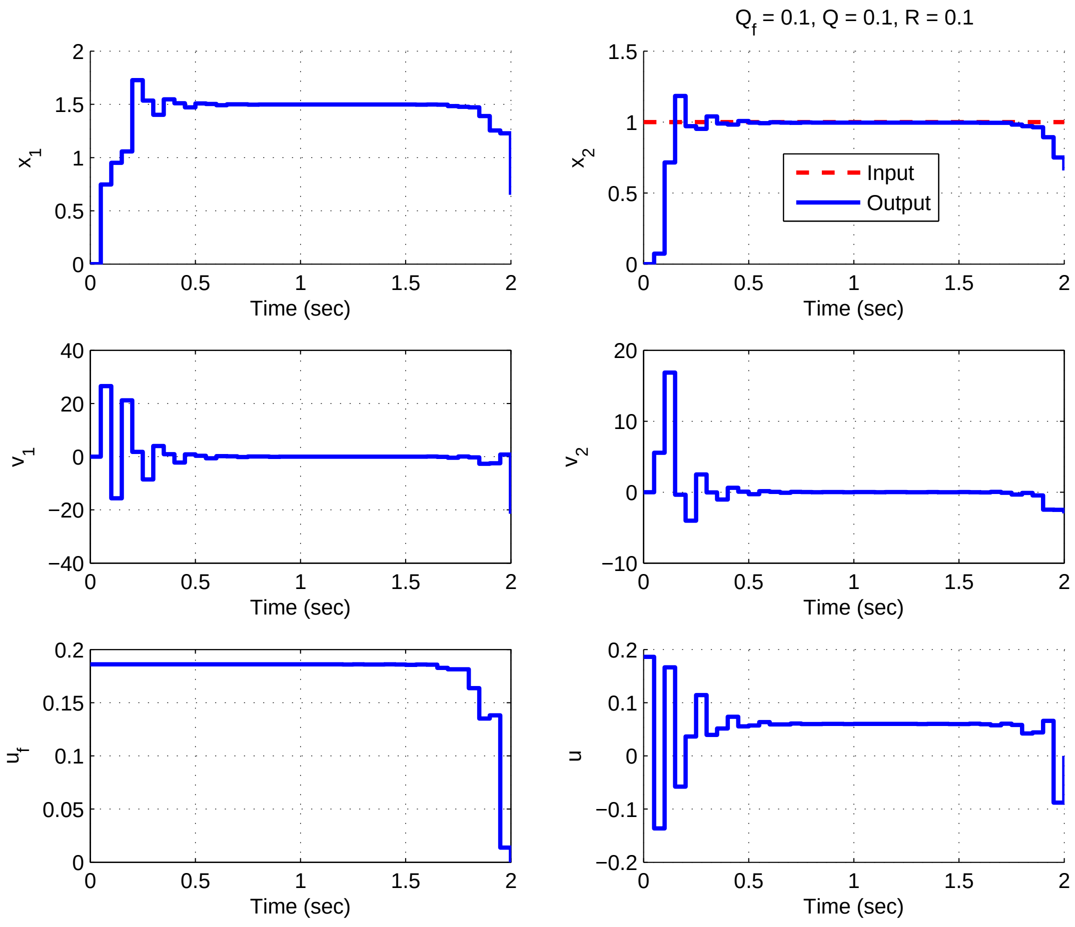

Figure 56: Same system as in Figure 54 only we are using the constant gains instead of time-varying gains.

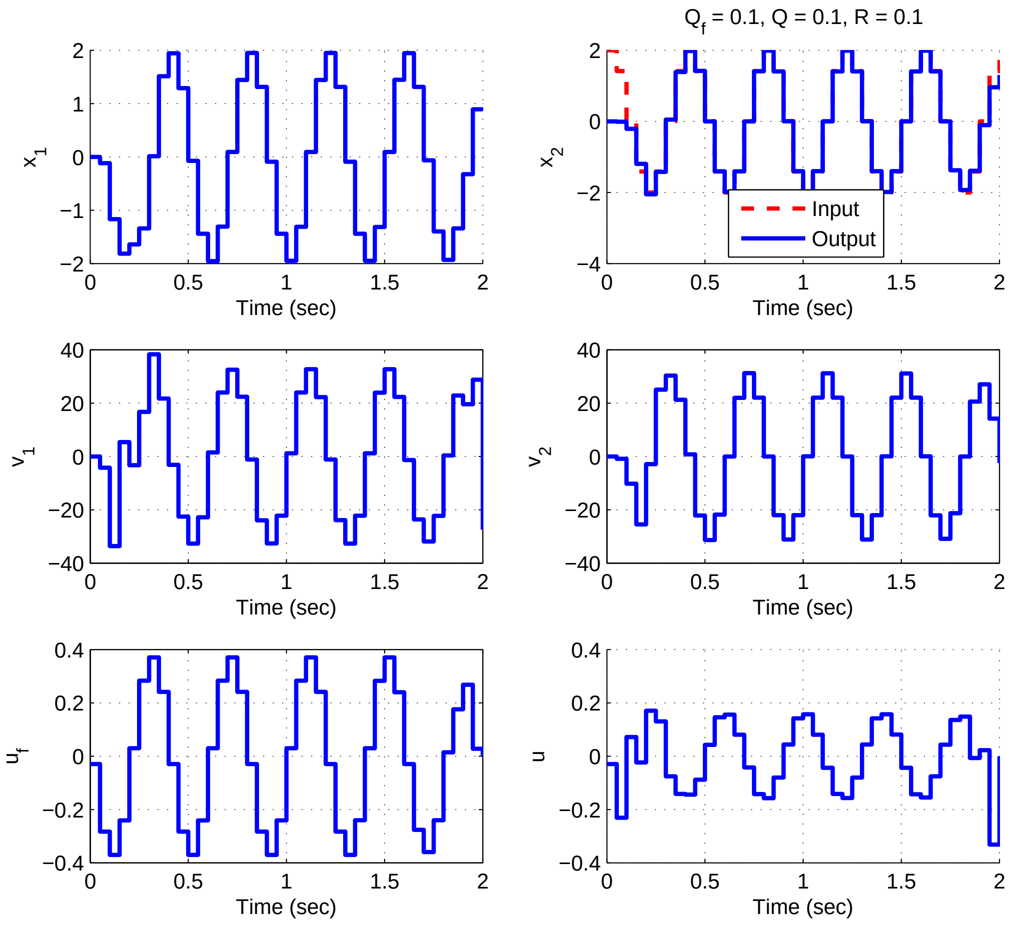

Figure 57: Tracking a sinusoid with constant gains. Note that both the feedforward control and control signal *lead* the known input.