4.2.2 Matrix Functions via Eigenvalue Decomposition

The method for computing matrix functions \( f(A) \) given in the previous subsection will work even if the matrix \( A \) has repeated eigenvalues. In this subsection we present another method for computing matrix functions. This method is simple to use, but it applies only to matrices with distinct eigenvalues.

From Chapter 2 we know that the eigenvectors of a matrix with distinct eigenvalues are linearly independent (Fact 2.35). If \( A \) is a diagonal matrix with the eigenvalues of \( A \) as its diagonal elements, and if \( X \) is a matrix whose columns are the corresponding eigenvectors of \( A \) (see (2.30) in Chapter 2), then

\[
AX = X\Lambda \quad \text{or} \quad A = X\Lambda X^{-1}.
\]

(4.33)

For future reference, recall the following notation for a diagonal matrix:

\[
\text{diag}(d_1, d_2, \ldots, d_n) = \begin{bmatrix}
d_1 & 0 & \cdots & 0 \\
0 & d_2 & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & d_n
\end{bmatrix}.
\]

The decomposition of \( A \) in (4.33) can be used to compute the matrix function \( f(A) \) where the scalar function \( f(\tau) \) is defined by the power series

\[
f(\tau) = \sum_{k=0}^{\infty} c_k \tau^k.
\]

By writing out the first few terms of the corresponding matrix power series, it is easy to show that if \( D \) is a diagonal matrix \( D = \text{diag}(d_1, \ldots, d_n) \), then the matrix function \( f(D) \) is also a diagonal matrix given by

\[
f(D) = \text{diag}(f(d_1), \ldots, f(d_n)).
\]

Using this result and the power series definition of \( f(A) \), it is also easy to show that

\[
f(A) = X f(D) X^{-1}
\]

\[
= X \text{ diag}(f(\lambda_1), \ldots, f(\lambda_n)) X^{-1}.
\]

(4.34)

This equation shows that the matrix function \( f(A) \) can be evaluated by first computing the scalar function \( f(\tau) \) at the \( n \) eigenvalues of \( A \) and then multiplying by the eigenvector matrices in (4.34).

As an example of these results, suppose we want to compute \( e^{AT} \) where \( T \) is an unknown sampling period and \( A \) is a matrix with distinct eigenvalues. Using (4.34) we have

\[
e^{AT} = X \text{ diag}(e^{\lambda_1 T}, \ldots, e^{\lambda_n T}) X^{-1}.
\]

(4.35)

It would be useful to express \( e^{AT} \) as a weighted sum of scalar exponentials as follows:

\[
e^{AT} = M_1 e^{\lambda_1 T} + \cdots + M_n e^{\lambda_n T}.
\]

(4.36)

This can be done by partitioning \( X \) into columns and \( X^{-1} \) into rows as follows:

\[
X = [x_1 \cdots x_n], \quad X^{-1} = \begin{bmatrix}
y_1^T \\
\vdots \\
y_n^T
\end{bmatrix}.
\]

(4.37)

By applying the rules of matrix multiplication to the right-hand side of (4.35), it can be shown that the matrices \( M_i \) in (4.36) are given by

\[
M_i = x_i y_i^T.
\]

(4.38)

Example 4.4. In this example we use the eigenvalue decomposition to compute \( e^{AT} \) where \( T \) is arbitrary and

\[
A' = \begin{bmatrix}
0 & 1 \\
0 & -2
\end{bmatrix}.
\]

The eigenvalues and eigenvectors of \( A \) are shown in the matrices

\[
\Lambda = \begin{bmatrix}
0 & 0 \\
0 & -2
\end{bmatrix}, \quad X = \begin{bmatrix}
1 & -0.4472 \\
0 & 0.8944
\end{bmatrix}.
\]

Using (4.35) we have

\[
e^{AT} = X \begin{bmatrix}
e^{0} & 0 \\
0 & e^{-2T}
\end{bmatrix} X^{-1}
\]

\[
= \begin{bmatrix}
1 & -0.4472 \\
0 & 0.8944
\end{bmatrix} \begin{bmatrix}
1 & 0.5000 \\
0 & 1.1180
\end{bmatrix} \begin{bmatrix}
1 & 0 \\
0 & 1
\end{bmatrix} e^{0}
\]

and using (4.37) and (4.38) we get

\[
e^{AT} = \begin{bmatrix}
1 & 0.5 \cdot \frac{1}{1 + e^{-2T}} \\
0 & 0 \cdot \frac{1}{1 + e^{-2T}}
\end{bmatrix} e^{0}
\]

This result checks with Example 4.1.

4.3 DISCRETIZING A SYSTEM WITH TIME DELAY*

As mentioned in Chapter 3, there are many industrial processes that can be modeled by a linear system with a time delay. In addition, the use of digital control always results in a time delay in the control loop. The reason is that it takes a certain amount of time for the compensator to compute its output value given a new input value. This computational delay can be included in the description of the plant for the purpose of design. If the computational delay is very small, it probably does not have to be included in the design model. However, the results of this section can be used to assess the effect of time delay on the closed-loop system and to decide whether or not it is negligible.
A system with time delay has a transfer function of the following form (see Table 3.1 on page 70 and Section 3.6.5):

$$G(s) = e^{-sD}G_1(s).$$

(4.39)

If $G_1(s)$ has a state-space description $(A, b, c)$, then $G(s)$ can be described by the following equations:

$$\dot{x}(t) = Ax(t) + bu(t - D)$$
$$y(t) = cx(t).$$

(4.40)

Note that this equation is not a finite-dimensional state-space description for $G(s)$ because the input is delayed by $D$ seconds. As we show in the following paragraphs, the state vector $x(t_1)$ of a state-space model contains all the information necessary to compute the output for $t \geq t_1$. In other words, the state vector $x(t_1)$ summarizes the effect of past inputs ($t \leq t_1$) on future outputs ($t \geq t_1$). However, a delay in the input means that the vector $x(t_1)$ does not summarize the effect of past inputs.

Consider first the system without time delay. If the value of the state vector is known at time $t_1$, and if the input equals zero for $t \geq t_1$, then the output of the system can be calculated as follows:

$$\dot{x}(t) = Ax(t), \quad x(t_1) \text{ given}$$
$$x(t) = e^{\Delta t}x(t_1), \quad t \geq t_1$$
$$y(t) = cx(t).$$

(4.41)

In the second line of these equations, the state vector for all time greater than $t_1$ can be computed in terms of the state vector at time $t_1$. In other words, the $n$ state variables $x_1(t_1), \ldots, x_n(t_1)$ contain all the information about the future output of the system in response to inputs up to time $t_1$.

Now consider the system with time delay. If the input is zero for $t \geq t_1$, then the output of the system must be computed from

$$\dot{x}(t) = Ax(t) + bu(t - D), \quad x(t_1) \text{ given},$$
$$u(t) \text{ given for } t_1 \leq t \leq t_1 + D$$
$$x(t) = e^{\Delta t}x(t_1 + D), \quad t \geq t_1 + D$$
$$y(t) = cx(t).$$

(4.42)

So $x(t_1)$ does not contain all the information about the system even if the input is set to zero at time $t_1$. In addition to the $n$ state variables $x_1(t_1), \ldots, x_n(t_1)$, the input function $u(t)$ has to be known over the interval $t_1 \leq t \leq t_1 + D$ in order to solve the first equation shown in (4.42) for $x(t)$ in the interval $t_1 \leq t \leq t_1 + D$. The input signal over this interval contains a continuum of values in general, and so an infinite number of state variables are needed to summarize the information about the system at time $t_1$. Thus, a system with time delay is infinite-dimensional.

It is clear that infinite-dimensional systems are much more difficult to deal with than finite-dimensional systems. Nevertheless, we now show that it is possible to obtain a finite-dimensional ZOH equivalent model of a linear system with time delay. The ZOH model will exactly describe the output of such a system in response to a piecewise-constant input. The intuitive reason it is not difficult to handle systems with time delay when the input is piecewise-constant is that such an input signal can be represented by a finite number of values over the time-delay interval. This will be shown in the following development.

We begin with the vector differential equation model of a system with time delay:

$$\dot{x}(t) = Ax(t) + bu(t - D)$$
$$y(t) = cx(t).$$

(4.43)

Assume that this system is driven by an input that is piecewise constant over some sampling interval $T$; that is,

$$u(t) = u[k], \quad kT \leq t < kT + T.$$  

(4.44)

If we integrate the differential equation over one sample period, we obtain, in a similar way as before,

$$x(kT + T) = e^{\Delta T}x(kT) + \int_{kT}^{kT + T} e^{\Delta (\tau - D)}b u(\tau - D) d\tau.$$  

(4.45)

In the previous derivation without time delay, we made use of the fact that the input was constant over the region of integration, and so we could pull it out of the integral. In the present case with time delay, it is not necessarily true that the delayed input is constant over the interval $kT$ to $kT + T$. (It is only true in the special case in which the time delay is an integer multiple of the sampling interval.) This is most easily seen by looking at Fig. 4.5.

We see that if the time delay $D$ is not an integer multiple of the sampling period $T$, then the delayed input signal will take two different values in the interval $kT$ to $kT + T$. Thus, to perform the integration in (4.45), we can split the integral into two parts, where the delayed input signal is constant over each part. The input can then

![Figure 4.5](image-url)

(a) A piecewise-constant signal $u(t)$. (b) The delayed signal $u(t - D)$. In this figure, $D = (1 + \frac{1}{2})T$. From (a), we see that $\alpha$ has the value $u(kT - 2T)$ and $\beta$ has the value $u(kT - T)$. 
be taken outside of each integral. Fig. 4.5 shows the situation when \( D = T + \frac{1}{2}T \). In general, the time delay will equal some integer number (possibly zero) of sampling intervals plus a fractional part of a sampling interval. That is, we can represent any delay \( D \) as \( D = qT + \gamma \) where \( q \geq 0 \) and \( 0 < \gamma \leq T \). In Fig. 4.5 we have \( q = 1 \) and \( \gamma = \frac{1}{2}T \). Note that when \( D \) is equal to an integer multiple of the sampling period, say \( D = mT \), we let \( \gamma = T \) and \( q = m - 1 \). We could also let \( q = m \) and \( \gamma = 0 \), but this is not advisable, as will be explained shortly.

In order to perform the integration in (4.45), we need to know the value of \( u(t - D) \) over the interval \( kT \) to \( kT + T \). In the general case, the result is

\[
u(t - D) = \begin{cases} u(kT - (q + 1)T), & kT \leq t < kT + \gamma \\ u(kT - qT), & kT + \gamma \leq t < kT + T. \end{cases} \tag{4.46}
\]

The reader can verify that this expression is correct for the example shown in Fig. 4.5. We can now substitute the above expression for the delayed input into (4.45) and split the integral into two parts:

\[
x(kT + T) = e^{AT}x(kT) + \int_{kT}^{kT+\gamma} e^{A(kT + \tau - T)}b \, d\tau \, u(kT - qT - T) \\
+ \int_{kT}^{kT+\gamma} e^{A(kT + \tau - T)}b \, d\tau \, u(kT - qT). \tag{4.47}
\]

The integrals in the above equation, which we label \( \Gamma_1 \) and \( \Gamma_0 \), respectively, turn out to be independent of the time index \( k \). Making the substitution \( \gamma = kT + T - \tau \) in the first integral yields

\[
\Gamma_1 = \int_{kT}^{T} e^{A\tau}b \, d\tau. \tag{4.48}
\]

and making the same substitution in the second integral yields

\[
\Gamma_0 = \int_{0}^{T-\gamma} e^{A\tau}b \, d\tau. \tag{4.49}
\]

Note that \( \Gamma_0 + \Gamma_1 = \Gamma \) where \( \Gamma \) is the input vector for the ZOH model without time delay. Substituting \( \Gamma_0 \) and \( \Gamma_1 \) into (4.47) and changing to discrete-time notation yields

\[
x(k + 1) = e^{AT}x(k) + \Gamma_1 u(k - q - 1) + \Gamma_0 u(k - q). \tag{4.50}
\]

We assume temporarily that \( q \geq 1 \). The case when \( q = 0 \) will be handled separately.

Equation (4.50) describes how the variables in the vector \( x(t) \) of the system with time delay behave at sampling instants. It is an exact equation—no approximations were made—but it is not in the form of a state-update equation for a discrete-time system. Recall that the right-hand side of a state-update equation must consist of a linear combination of state variables at time \( k \) and the input at time \( k \). In (4.50), the input term does not appear at time \( k \). We can remedy this situation by defining two new state variables as follows:

\[
z_1[k] = u(k - q - 1) \tag{4.51}
\]

\[
z_2[k] = u(k - q). \tag{4.52}
\]

If these new state variables are substituted into (4.50), then the right-hand side of (4.50) consists of a linear combination of state variables at time \( k \) as desired. However, if \( z_1[k] \) and \( z_2[k] \) are state variables, then we need to write state-update equations for them. Using (4.51) we have

\[
z_1[k + 1] = u(k - q) = z_2[k] \tag{4.53}
\]

\[
z_2[k + 1] = u(k - q + 1). \tag{4.54}
\]

The first equation above is a valid state-update equation because the right-hand side consists of the state variable \( z_2 \) at time \( k \). The second equation in (4.52) is a valid state-update equation only if \( q = 1 \) so that the input term appears at time \( k \). If \( q > 1 \), however, we can define \( z_1[k] = u(k - q) \) and proceed as before. In general, we define as many new state variables as are needed so that the right-hand side of the state-update equation for the last state variable consists of the input at time \( k \).

To summarize the above discussion, equation (4.50) can be written as a valid state-update equation if additional state variables are used to store past values of the input. It turns out that exactly \( q + 1 \) additional state variables are needed:

\[
z_1[k] = u(k - q - 1) \tag{4.55}
\]

\[
z_2[k] = u(k - q) \tag{4.56}
\]

\[
: = : \tag{4.57}
\]

\[
z_{q+1}[k] = u(k - 1). \tag{4.58}
\]

Because these state variables are consecutive shifts of the input sequence, it is easy to write update equations for them. The update equations are

\[
z_1[k + 1] = u(k - q) = z_2[k] \tag{4.59}
\]

\[
z_2[k + 1] = u(k - q + 1) = z_1[k] \tag{4.60}
\]

\[
: = : \tag{4.61}
\]

\[
z_{q}[k + 1] = u(k - 1) = z_{q+1}[k] \tag{4.62}
\]

\[
z_{q+1}[k + 1] = u(k). \tag{4.63}
\]

Notice that the update equation for \( z_{q+1} \) is written in terms of the input at time \( k \), which is a valid update equation. The update equation (4.50) for the vector \( x[k] \) can now be written with the help of the additional state variables \( z_1[k] \) and \( z_2[k] \). Using (4.53) and (4.50), we can write

\[
x[k + 1] = e^{AT}x[k] + \Gamma_1 z_1[k] + \Gamma_0 z_2[k], \quad q \geq 1, \tag{4.64}
\]

which is a valid state-update equation that does not depend explicitly on the input. Note that the above equation is valid only when \( q \geq 1 \) because we have assumed that \( z_2[k] \) is defined as a state variable. If \( q = 0 \) so that the time delay is less than
or equal to the sample period, then only one additional state variable is defined, and then (4.50) can be written as

$$x[k+1] = e^{AT}x[k] + \Gamma_1 z_1[k] + \Gamma_0 u[k], \quad q = 0.$$  \hspace{1cm} (4.56)

A complete state-update equation for a system with time delay can now be obtained by combining (4.55) or (4.56) with (4.54) to obtain

$$
\begin{bmatrix}
x[k+1] \\
z_1[k+1] \\
z_2[k+1] \\
\vdots \\
z_q[k+1] \\
z_{q+1}[k+1]
\end{bmatrix} =
\begin{bmatrix}
\Phi & \Gamma_1 & \Gamma_0 & 0 & \cdots & 0 \\
0 & 0 & 1 & 0 & \cdots & 0 \\
0 & 0 & 0 & 1 & \cdots & 0 \\
\vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & 0 & 0 & \cdots & 1 \\
0 & 0 & 0 & 0 & \cdots & 0
\end{bmatrix}
\begin{bmatrix}
x[k] \\
z_1[k] \\
z_2[k] \\
\vdots \\
z_q[k] \\
z_{q+1}[k]
\end{bmatrix} +
\begin{bmatrix}
0 \\
0 \\
0 \\
\vdots \\
0 \\
1
\end{bmatrix} u[k], \quad q \geq 1,
$$  \hspace{1cm} (4.57)

or

$$
\begin{bmatrix}
x[k+1] \\
z_1[k+1]
\end{bmatrix} =
\begin{bmatrix}
\Phi & \Gamma_1 \\
0 & 0
\end{bmatrix}
\begin{bmatrix}
x[k] \\
z_1[k]
\end{bmatrix} +
\begin{bmatrix}
\Gamma_0 \\
1
\end{bmatrix} u[k], \quad q = 0.  \hspace{1cm} (4.58)
$$

Notice that the above equations contain partitioned matrices. For instance, each zero below the matrix $\Phi$ represents a row vector of $n$ zeros.

For any value of $q$, the output equation is obtained from (4.40) to be

$$y[k] = [c \ 0 \ \cdots \ 0]
\begin{bmatrix}
x[k] \\
z_1[k] \\
\vdots \\
z_{q+1}[k]
\end{bmatrix}.  \hspace{1cm} (4.59)$$

Notice that the state vector contains $n + q + 1$ state variables. The vector $x[k]$ contains $n$ state variables corresponding to the system without time delay, and there are $q + 1$ additional state variables that are used to model the time delay. Recall that the time delay $D$ is represented as $D = qT + \gamma$. Thus, if the time delay is large compared to the sampling interval, then $q$ will be a large number, and many additional state variables will have to be used to store samples of the input sequence.

Let us now consider the special case in which the time delay is an integer multiple of the sampling period; that is, $D = mT$. We have stated previously that in this case, one should choose $\gamma = T$ and $q = m - 1$. With this choice, $\Gamma_0 = 0$ (see (4.49)) and $m$ state variables have to be added to model the time delay. The alternative when $D = mT$ is to choose $q = m$ and $\gamma = 0$. In this case, $\Gamma_1 = 0$ (see (4.48)) and $m + 1$ state variables have to be added to the model. But inspection of (4.57) shows that when $\Gamma_1 = 0$, the state variable $z_1[k]$ is not used in updating the state vector, and this variable can thus be deleted from the equations. Deleting this variable leaves $m$ additional state variables, and the equations reduce to those obtained by choosing $\gamma = T$ and $q = m - 1$.

Before giving an example of the computation of a ZOH equivalent for a system with time delay, we remark that the special case $D = mT$ occurs frequently when modeling digital control systems. The reason has to do with a common way of implementing digital control, which is described next. An input to the compensator is read at time $kT$. The output value is calculated in a time that is less than the sampling period $T$; however, the output value is actually sent to the D/A converter at the next sampling instant $kT + T$. This procedure results in a "computational" time delay of exactly $T$ seconds, as shown in Fig. 4.6. If this procedure is used to control a linear, time-invariant plant described by the state-space model $(A, b, c)$, then a time delay of $\gamma = T$ seconds (with $q = 0$) should be included in the model of the plant. A digital compensator designed using the plant model with time delay will properly account for the computational delay.

**Example 4.5.** Consider the single-time-constant system with time delay that was introduced in Section 3.6.5. Let the time constant be 1 second and the time delay be 0.66 seconds. A differential equation for this system of the type shown in (4.43) is

$$\dot{x}(t) = x(t) + u(t - 0.66)$$
$$y(t) = x(t).$$

Note for this system the matrix $A = 1$ and $b = 1$. Suppose that we want to use a sampling rate of $T = 0.2$ seconds. In this case, the delay can be decomposed as $0.66 = 3(0.2) + 0.06$ so that $q = 3$ and $\gamma = 0.06$. We compute $\Phi = e^{AT} = e^{0.2} = 1.2214$ and use (4.48) and (4.49) to compute $\Gamma_1$ and $\Gamma_0$ as follows:

$$\Gamma_1 = \int_{0.2-0.06}^{0.2} e^{sT} ds = e^{0.2} |_{t=0.2} - e^{0.2} |_{t=0.14} = 0.0711$$
and

$$\Gamma_0 = \int_{0}^{0.14} e^{sT} ds = e^{0.14} |_{t=0} - e^{0.14} |_{t=0.14} = 0.1503.$$

Recall that we have to define $q + 1 = 4$ additional state variables $z_1[k], \ldots, z_4[k]$. Then the ZOH model from (4.57) is

**Figure 4.6** Timing sequence of a single input/output pair of samples of a digital compensator using a typical implementation procedure. The output of the compensator is delayed by exactly one sample period.
\[
\begin{bmatrix}
    x[k + 1] \\
    z_1[k + 1] \\
    z_2[k + 1] \\
    z_3[k + 1] \\
    z_4[k + 1]
\end{bmatrix} =
\begin{bmatrix}
    1.2214 & 0.0711 & 0.1503 & 0 & 0 \\
    0 & 0 & 1 & 0 & 0 \\
    0 & 0 & 0 & 1 & 0 \\
    0 & 0 & 0 & 0 & 1 \\
    0 & 0 & 0 & 0 & 0
\end{bmatrix}
\begin{bmatrix}
    x[k] \\
    z_1[k] \\
    z_2[k] \\
    z_3[k] \\
    z_4[k]
\end{bmatrix} +
\begin{bmatrix}
    0 \\
    u[k]
\end{bmatrix}
\]

\[y[k] = [1 \ 0 \ 0 \ 0 \ 0] \begin{bmatrix}
    x[k] \\
    z_1[k] \\
    z_2[k] \\
    z_3[k] \\
    z_4[k]
\end{bmatrix}.
\]

Example 4.6. Consider a type-1 servo system (e.g., a dc motor) described by poles at \(s = 0\) and \(s = -10\). Suppose that the sampling period is \(T = 0.01\) seconds and that a digital compensator will introduce a time delay of one period. We can incorporate this time delay into the ZOH model of the plant as follows:

\[x(t) = \begin{bmatrix} 0 & 1 \\ 0 & -10 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t - 0.01)\]

\[y(t) = [1 \ 0] x(t).
\]

A ZOH equivalent model can be obtained by computing \(\Phi, \Gamma_0,\) and \(\Gamma_1\) and substituting these into (4.58). Example 4.1 derives the ZOH equivalent model for this system without time delay. The matrix \(\Phi\) will be the same in both examples, and so we evaluate the expression for \(\Phi\) in Example 4.1 with \(T = 0.01\) and \(p = 10\):

\[\Phi = \begin{bmatrix} 1 & 0.0000 \\ 0 & 0.0000 \end{bmatrix}.
\]

Recall that \(\Gamma_0\) defined in (4.49) equals \(\mathbf{0}\), and because \(\gamma = T\), we have from (4.48) that \(\Gamma_1 = \Gamma\), where \(\Gamma\) is given in Example 4.1:

\[\Gamma = \begin{bmatrix} 0.0000 \\ 0.0000 \end{bmatrix}.
\]

We can now use (4.58) to obtain the ZOH equivalent model:

\[
\begin{bmatrix}
    x[k + 1] \\
    z_1[k + 1] \\
    z_2[k + 1] \\
    z_3[k + 1] \\
    z_4[k + 1]
\end{bmatrix} =
\begin{bmatrix}
    1 & 0.0000 & 0 & 0 & 0 \\
    0 & 0.0000 & 0.0000 & 0 & 0 \\
    0 & 0 & 0 & 0 & 0 \\
    0 & 0 & 0 & 0 & 0 \\
    0 & 0 & 0 & 0 & 0
\end{bmatrix}
\begin{bmatrix}
    x[k] \\
    z_1[k] \\
    z_2[k] \\
    z_3[k] \\
    z_4[k]
\end{bmatrix} +
\begin{bmatrix}
    0 \\
    u[k]
\end{bmatrix}
\]

\[y[k] = [1 \ 0 \ 0 \ 0 \ 0] \begin{bmatrix}
    x[k] \\
    z_1[k] \\
    z_2[k] \\
    z_3[k] \\
    z_4[k]
\end{bmatrix}.
\]

This model can be used to assess the effect of time delay on a closed-loop control system. This is done in the next chapter.

### 4.4 THE ZOH POLE-MAPPING FORMULA

When the plant is replaced by its ZOH equivalent, the design model is a discrete-time system. The performance specifications for the design model are thus discrete-time specifications. However, the actual control system operates in continuous time, and it is natural to use continuous-time performance specifications. A convenient way to achieve performance specifications is to specify the pole locations that the closed-loop system should have. For continuous-time specifications, we would place closed-loop poles in the \(s\) plane. Because the design model is a discrete-time system, however, the desired closed-loop \(s\) plane poles must be mapped into an equivalent set of desired \(z\) plane poles. In this section we show that the poles of the ZOH design model are related in a simple way to the poles of the continuous-time plant. This relationship will be used in the next chapter to map desired \(s\) plane pole locations into the \(z\) plane.

It turns out that there is no simple relationship between the zeros of a continuous-time system and its ZOH equivalent [4]. However, there is a formula that relates the respective pole locations. Recall that the system matrices for the state-space descriptions of \(G(s)\) and \(G_z(z)\) are related by the formula \(\Phi = e^{AT}\) (see (4.12)). Also recall from Chapter 3 that the poles of a system are given by the eigenvalues of its system matrix. So the question of how the poles map under the ZOH transformation reduces to the question of how the eigenvalues of a matrix are transformed by the matrix exponential. This question can be answered by the following remarkable fact about functions of matrices and their eigenvalues:

**Fact.** Let \(B = f(A)\), where \(A\) and \(B\) are matrices. If \(\lambda_1\) is an eigenvalue of \(A\) then \(f(\lambda_1)\) is an eigenvalue of \(B\).

The proof of this fact is based on the Cayley-Hamilton Theorem and can be established by comparing (4.21), (4.25) and (4.28).

Because \(\Phi\) and \(A\) are related by \(\Phi = e^{AT}\), the eigenvalues of \(\Phi\) and \(A\) are related by the same exponential function. That is, from the definition of \(\Phi\) and the fact just established, a given \(s\) plane pole location \(s_0\) maps into the \(z\) plane by the ZOH mapping to the location

\[z_0 = e^{s_0 T}.
\]

Note that the exponential function maps the left half of the \(s\) plane into the interior of the unit circle in the \(z\) plane. This means that stable analog systems map to stable discrete-time systems.

It is sometimes useful to invert (4.60); that is, to solve for \(s_0\) in terms of \(z_0\). This is accomplished by taking logarithms of both sides of (4.60) to obtain

\[s_0 = \frac{1}{T} \ln(z_0).
\]

When \(z_0\) is a complex number, the above formula requires the logarithm of a complex number. The result can be expressed in terms of the logarithm of a real number as follows:

\[s_0 = \frac{1}{T} [\ln|z_0| + j \arg(z_0)],
\]

where \(|z_0|\) is the magnitude of the complex number \(z_0\), and \(\arg(z_0)\) is the argument.