3. DIGITAL TRACKERS

We now want to consider digital trackers. The block diagram for digital trackers looks the same as for analog trackers except that the transfer functions are a function of \( z \) and the time responses are a function of \( k \), a “digital” time variable. Thus the block diagram we will use to analyze digital, closed loop trackers is as given in Figure 22.

![Digital Tracker Block Diagram](image)

Figure 22 – Digital Tracker Block Diagram Used in These Analyzes

The same transfer function manipulations apply to the digital tracker. Thus, the closed loop transfer function is

\[
G_{CL}(z) = \frac{K_R G_{OL}(z)(1/K_R)}{1 + K_R G_{OL}(z)(1/K_R)} = \frac{G_{OL}(z)}{1 + G_{OL}(z)}.
\]  

(87)

As with analog trackers, we can have Type 0, 1, 2, etc. servo loops. A Type \( N \) digital tracker will have \( N \) poles at \( z = 1 \). That is, the open loop transfer function can be written as

\[
G_{OL}(z) = \frac{G(z)}{(z-1)^N}
\]  

(88)

where \( G(z) \) has no poles (or zeros) at \( z = 1 \). The same steady state response properties hold. However, the equation for finding steady state error from the \( z \)-transform is different. The appropriate equation is

\[
\Delta x_{ss} = \lim_{k \to \infty} \Delta x(k) = \lim_{z \to 1} \left[ (z-1) \Delta X(z) \right].
\]  

(89)

\( z \)-transforms for some standard inputs are given in Table 2.
Table 2 – $z$-transforms of Some Standard Inputs

<table>
<thead>
<tr>
<th>$f(k)$</th>
<th>$F(z)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Au(k)$ (step)</td>
<td>$\frac{Az}{z-1}$</td>
</tr>
<tr>
<td>$Aku(k)$ (ramp)</td>
<td>$\frac{Az}{(z-1)^2}$</td>
</tr>
<tr>
<td>$\frac{Ak^2}{2}u(k)$ (quadratic)</td>
<td>$\frac{Az(z+1)}{2(z-1)^3}$</td>
</tr>
<tr>
<td>$\frac{Ak^n}{n!}u(k)$ ($n^{th}$ order)</td>
<td>$A\lim_{\alpha \to 0} \frac{(-1)^n}{n!} \frac{\partial^n}{\partial \alpha^n} \left[ \frac{z}{z-e^{-\alpha}} \right]$</td>
</tr>
</tbody>
</table>

We can also create root loci for digital trackers. They have the same construction properties as for analog servos. For example, the root locus for

$$G_{ol}(z) = \frac{K_{ol}(z-b)}{(z-1)(z-a)}$$

would be as shown in Figure 23 for $0 < b < a < 1$. The closed loop tracker associated with this $G_{ol}(z)$ is a Type 1 servo since $G_{ol}(z)$ has one pole at $z=1$.

![Figure 23 – Root Locus for Equation (90) for $0 < b < a$](image)

For digital trackers, the regions of the $z$-plane that are of interest are those bounded by a unit circle in that plane. This is illustrated in Figure 24. In the $s$-domain, the imaginary axis
separates the s-plane into two parts. If all poles of $G(s)$ lie in the left half of the s-plane, the response will decay to zero (the system is asymptotically stable). If any poles lie in the right half, the response increases without bound (the system is unstable). If any poles lie on the imaginary axis, the response could increase without bound or oscillate or remain constant (the system is marginally stable).

In the z-domain, the dividing line is the unit circle of the z-plane. If any poles of $G(z)$ lie outside of the unit circle, the response increases without bound (the system is unstable). If all of the poles lie inside of the unit circle, the response will decay to zero (the system is asymptotically stable). However, if any of the poles lie inside the left half of the unit circle, the response will oscillate from sample to sample. If any poles lie on the unit circle, the response could increase without bound or oscillate or remain constant (the system is marginally stable). As indicated in Figure 24, the point $z = 1$ is analogous to the origin of the s-plane.

As indicated in Figure 24, the point $z = 1$ is analogous to the origin of the s-plane.

Figure 24 – Description of the significant parts of the z-plane

When we create a frequency response for a continuous time system represented by $G(s)$, we compute $|G(s)|$ or $|G(s)|^2$ for $s = j\omega = j2\pi f$. When we create a frequency response for a discrete time (digital) system represented by $G(z)$, we compute $|G(z)|$ or $|G(z)|^2$ for $z = e^{j\theta}$, where $\theta$ varies from 0 to $2\pi$ or $-\pi$ to $\pi$. In other words, as $z$ varies around the unit circle of the z plane. While this is strictly legal, this approach makes it difficult to relate the frequency response to frequency, $f$. To get around this problem, we prefer to compute $|G(z)|$ or $|G(z)|^2$ for $z = e^{j2\pi f T}$ where $f$ varies from 0 to $1/T$ or $-1/(2T)$ to $1/(2T)$. Of course, this means that we must know the sample period, $T$. In trackers we know this because it is the track update period.
3.1 Tracker Transient Response

As with analog trackers we want to discuss the transient response of digital trackers. To that end, we want to discuss how to obtain a state variable representation of a digital system from its z-transform. We will also want to discuss how to design the open loop transfer function to obtain a desired closed loop response.

We define the (one sided) z-transform of some discrete time function, \( x(k) \), as

\[
X(z) = \mathcal{Z}[x(k)] = \sum_{m=0}^{\infty} x(m) z^{-m}.
\] (91)

This carries the tacit assumption that \( x(k) = 0 \quad \forall \ k < 0 \).

To get the z-transform of \( x(k+1) \) we write

\[
X'(z) = \mathcal{Z}[x(k+1)] = \sum_{m=0}^{\infty} x(m+1) z^{-m}.
\] (92)

If we let \( l = m+1 \), or \( m = l-1 \) we get

\[
X'(z) = \sum_{m=0}^{\infty} x(m+1) z^{-m} = \sum_{l=1}^{\infty} x(l) z^{-(l-1)} = z \sum_{l=1}^{\infty} x(l) z^{-l}.
\] (93)

If we add and subtract \( x(0) \) from the last sum we get

\[
X'(z) = z \left[ \sum_{l=1}^{\infty} x(l) z^{-l} + x(0) - x(0) \right] = z \sum_{l=0}^{\infty} x(l) z^{-l} - x(0) = zX(z) - zx(0).
\] (94)

For \( \mathcal{Z}[x(k+2)] \) we get

\[
X''(z) = \mathcal{Z}[x(k+2)] = \sum_{m=0}^{\infty} x(m+2) z^{-m} = \sum_{l=2}^{\infty} x(l) z^{-(l-2)} = z^2 \sum_{l=2}^{\infty} x(l) z^{-l}
\]

\[
= z^2 \left[ \sum_{l=2}^{\infty} x(l) z^{-l} + x(1) z^{-1} + x(0) - x(0) - x(1) z^{-1} \right]
\] (95)

\[
= z^2 \left[ \sum_{l=0}^{\infty} x(l) z^{-l} - \sum_{r=0}^{l-1} x(r) z^{-r} \right] = z^2 X(z) - z^2 \sum_{r=0}^{n-1} x(r) z^{-r}
\]

With some thought we can generalize this to

\[
X^{(n)}(z) = \mathcal{Z}[x(k+n)] = z^n X(z) - z^n \sum_{r=0}^{n-1} x(r) z^{-r}.
\] (96)

We can write the z-transform of \( x(k) \), \( \mathcal{Z}[x(k-1)] \), as

© 2015 M. C. Budge, Jr (merv@thebudges.com) 4
\[ X^{(-1)}(z) = \mathcal{Z}[x(k-1)] = \sum_{m=0}^{\infty} x(m-1) z^{-m} = \sum_{l=1}^{\infty} x(l) z^{-(l+1)} \]

\[ = z^{-1} \sum_{l=1}^{\infty} x(l) z^{-l} = z^{-1} \left[ \sum_{l=0}^{\infty} x(l) z^{-l} + x(-1) \right] = z^{-1} X(z) + zx(-1) \]

Since \( x(k) = 0 \) \( \forall \ k < 0 \) this reduces to

\[ X^{(-1)}(z) = \mathcal{Z}[x(k-1)] = z^{-1} X(z). \]  

(98)

We can generalize this to

\[ X^{(-n)}(z) = \mathcal{Z}[x(k-n)] = z^{-n} X(z). \] 

(99)

Now let us discuss how to go from a \( z \)-transfer function to a difference equation, and to a state variable representation. We start with

\[ G(z) = \frac{Y(z)}{U(z)} = \frac{z^m + b_1 z^{m-1} + b_2 z^{m-2} + \cdots + b_{m-1} z + b_m}{z^n + a_1 z^{n-1} + b_2 z^{n-2} + \cdots + a_{n-1} z + a_n}. \] 

(100)

We next break this into two parts as

\[ W(z) = \frac{1}{U(z)} \frac{z^n + a_1 z^{n-1} + b_2 z^{n-2} + \cdots + a_{n-1} z + a_n}{z^m + b_1 z^{m-1} + b_2 z^{m-2} + \cdots + b_{m-1} z + b_m}. \] 

(101)

and

\[ \frac{Y(z)}{W(z)} = \frac{z^m + b_1 z^{m-1} + b_2 z^{m-2} + \cdots + b_{m-1} z + b_m}{z^n + a_1 z^{n-1} + b_2 z^{n-2} + \cdots + a_{n-1} z + a_n}. \]

(102)

From (101) we get

\[ U(z) = \left( \frac{z^n + a_1 z^{n-1} + b_2 z^{n-2} + \cdots + a_{n-1} z + a_n}{z^m + b_1 z^{m-1} + b_2 z^{m-2} + \cdots + b_{m-1} z + b_m} \right) W(z). \] 

(103)

or

\[ z^n W(z) + a_1 z^{n-1} W(z) + \cdots + a_{n-1} z W(z) + a_n W(z) = U(z). \] 

(104)

From (82) we have

\[ \mathcal{Z}\left[ w(k+n) \right] = z^n W(z) - z^{-1} \sum_{r=0}^{n-1} w(r) z^{-r}. \] 

(105)

As shown in the appendix, we can drop the summation term of (105) because \( w(r) = 0 \) for \( r = 0, 1, \ldots, n-1 \). With this, we get

\[ \mathcal{Z}\left[ w(k+n) \right] = z^n W(z) \] 

(106)

or
\[ \mathcal{Z}^{-1}[z^nW(z)] = w(k+n). \]  

(107)

If we use this to take the inverse z-transform of (104) we get

\[ w(k+n) + a_1w(k+n-1) + a_2w(k+n-2) + \cdots + a_{n-1}w(k+1) + a_nw(k) = u(k). \]  

(108)

From (102) we get

\[ Y(z) = \left(z^m + b_1z^{m-1} + b_2z^{m-2} + \cdots + b_{m-1}z + b_m\right)W(z) \]  

(109)

and

\[ y(k) = w(k+m) + b_1w(k+m-1) + b_2w(k+m-2) + \cdots + b_{m-1}w(k+1) + b_mw(k). \]  

(110)

To derive a state variable representation we define a set of states as

\[
\begin{align*}
  x_1(k) &= w(k) \\
  x_2(k) &= w(k+1) \\
  x_3(k) &= w(k+2) \\
  &\vdots \\
  x_n(k) &= w(k+n-1)
\end{align*}
\]

(111)

By shifting the states by one time step, or stage, we get

\[
\begin{align*}
  x_1(k+1) &= w(k+1) = x_2(k) \\
  x_3(k+1) &= w(k+2) = x_3(k) \\
  x_3(k+1) &= w(k+3) = x_4(k). \\
  &\vdots \\
  x_n(k+1) &= w(k+n)
\end{align*}
\]

(112)

But

\[
\begin{align*}
  w(k+n) &= -a_1w(k+n-1) - a_2w(k+n-2) - \cdots - a_{n-1}w(k+1) - a_nw(k) + u(k) \\
  &= -a_1x_n(k) - a_2x_{n-1}(k) - \cdots - a_{n-1}x_2(k) - a_nx_1(k) + u(k)
\end{align*}
\]

(113)

If we substitute (113) into (112) and write the result in matrix notation we get

\[ X(k+1) = FX(k) + Gu(k) \]  

(114)

where

\[
X(k) = \begin{bmatrix} x_1(k) \\ x_2(k) \\ x_3(k) \\ \vdots \\ x_n(k) \end{bmatrix},
\]

(115)
\[
F = \begin{bmatrix}
0 & 1 & 0 & \cdots & 0 \\
0 & 0 & 1 & \cdots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & 0 & \cdots & 1 \\
-a_n & -a_{n-1} & -a_{n-2} & \cdots & -a_1 \\
\end{bmatrix}, \quad (116)
\]

and

\[
G = \begin{bmatrix}
0 \\
0 \\
0 \\
\vdots \\
1 \\
\end{bmatrix}. \quad (117)
\]

From (110) we get

\[
y(k) = x_{m+1}(k) + b_1x_m(k) + b_2x_{m-1}(k) + \cdots + b_{m-1}x_2(k) + b_mx_1(k) \quad (118)
\]
or

\[
y(k) = H^T X(k) \quad (119)
\]

where

\[
H^T = \begin{bmatrix}
b_m & b_{m-1} & \cdots & b_1 & 1 & 0 & \cdots & 0 \\
\end{bmatrix}. \quad (120)
\]

Since we will need it when we study g-h and g-h-k filters, we want to determine how to go from a state variable representation to a z-transfer function.

If we take the z-transform of (114) and (119) we get

\[
zX(z) = FX(z) + Gu(z) \quad (121)
\]

and

\[
y(z) = H^T X(z). \quad (122)
\]

From (121) we get

\[
(zI - F)X(z) = Gu(z). \quad (123)
\]

Solving for \( X(z) \) yields

\[
X(z) = (zI - F)^{-1} Gu(z). \quad (124)
\]

Substituting (124) in to (122) results in

\[\]

\[1 \text{ Here we made use of (80). We were able to drop the } zX(k)|_{k=0} \text{ term because } X(k)|_{k=0} = 0.\]
\[ y(z) = H^T(zI - F)^{-1} Gu(z). \]  \hspace{1cm} (125)

From (125) we get the z-transfer function as

\[ G_r(z) = \frac{y(z)}{u(z)} = H^T(zI - F)^{-1} G. \]  \hspace{1cm} (126)

Rather than consider low pass filters for the track filter, we will consider specific cases of a g-h filter and a g-h-k filter. These filters have been discussed widely in journal articles and text books and are used in both open and closed loop digital trackers. Most of the literature concerning the design of g-h and g-h-k filters focuses on open loop trackers and uses criteria such as minimizing mean-square error. In this course, we will take a simplistic approach of designing the filter to obtain a desired closed-loop bandwidth and damping coefficient, like we did for the analog tracker.

3.2 g-h Filter

One of the forms of the g-h filter is

\[ x_p(k+1) = x_s(k) + T \dot{x}_s(k) \]
\[ \dot{x}_p(k+1) = \dot{x}_s(k) \]
\[ x_s(k+1) = x_p(k+1) + g(k+1)(y(k+1) - x_p(k+1)) \]  \hspace{1cm} (127)
\[ \dot{x}_s(k+1) = \dot{x}_p(k+1) + \frac{h(k+1)}{T}(y(k+1) - x_p(k+1)) \]

where \( x_p(k) \) and \( \dot{x}_p(k) \) are termed predicted position and rate estimates and \( x_s(k) \) and \( \dot{x}_s(k) \) are smoothed position and rate estimates. \( y(k) \) is the position measurement. The filter gains are \( g(k) \) and \( h(k) \), which, in general, are allowed to vary with time. In our case we assume they are fixed. \( T \) is the update period of the tracker. This, in general, could also be allowed to vary with \( k \). The term \( y(k+1) - x_p(k+1) \) is the error between the measured position and the predicted position. In a closed loop tracker, we assume that \( y(k) \) is not directly measured; instead, the error sensor measures \( \Delta x(k) = y(k) - x_p(k) \) and reports a scaled version as \( e(k) = K_h \Delta x(k) \) (see Figure 22). In a closed loop tracker we are not explicitly interested in \( x_s(k) \) and \( \dot{x}_s(k) \).

---


With the above we rewrite (127) to eliminate \(x_s(k)\) and \(\dot{x}_s(k)\), and replace \(y(k) - x_p(k)\) with \(e(k)\). If we substitute the \(x_s(k+1)\) and \(\dot{x}_s(k+1)\) into the \(x_p(k+1)\) and \(\dot{x}_p(k+1)\) equations, with the appropriate time shift, we get

\[
x_p(k+1) = x_p(k) + ge(k) + T \left( \dot{x}_p(k) + \frac{h}{T} e(k) \right)
\]

\[
\dot{x}_p(k+1) = \dot{x}_p(k) + \frac{h}{T} e(k)
\]

If we write this in matrix form we get

\[
X_p(k+1) = FX_p(k) + Ge(k)
\]

where \(F = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix}\) \(G = \begin{bmatrix} g + h \\ h/T \end{bmatrix}\) \(X_p(k) = \begin{bmatrix} x_p(k) \\ \dot{x}_p(k) \end{bmatrix}\). The filter output is

\[
v_s(k) = H^T X_p(k)
\]

with \(H = \begin{bmatrix} 1 & 0 \end{bmatrix}\).

We now want to find the transfer function of the g-h filter. From the previous section we have

\[
G_p(z) = H^T (zI - F)^{-1} G
\]

We can compute the inverse as

\[
(zI - F)^{-1} = \begin{bmatrix} z & 0 \\ 0 & z \end{bmatrix} - \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix}^{-1} = \begin{bmatrix} z - 1 & -T \\ 0 & z - 1 \end{bmatrix}^{-1} = \begin{bmatrix} 1 & \frac{T}{z - 1} \\ 0 & \frac{1}{z - 1} \end{bmatrix}
\]

Using this in (131) gives

\[
G_p(z) = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} \frac{1}{z - 1} & \frac{T}{(z - 1)^2} \\ 0 & \frac{1}{z - 1} \end{bmatrix} \begin{bmatrix} g + h \\ h/T \end{bmatrix} = \begin{bmatrix} 1 & \frac{T}{(z - 1)^2} \end{bmatrix} \begin{bmatrix} g + h \\ h/T \end{bmatrix}
\]

\[
= \frac{g + h}{z - 1} + \frac{h}{(z - 1)^2} = \frac{(g + h)z - g}{(z - 1)^2}
\]

A block diagram of the tracker is shown in Figure 25.
We note that since \( G_{OL}(z) \) has two poles at \( z = 1 \), the tracker is a Type 2 servo. Which means it should be able to track a step and ramp input with zero steady state error, and a quadratic input with a constant steady state error.

The closed loop transfer function is

\[
G_{CL}(z) = \frac{G_{OL}(z)}{1 + G_{OL}(z)} = \frac{(g + h)z - g}{z^2 + (g + h - 2)z + 1 - g}
\]  

(134)

If we were to replace \( K_R \) with the variable \( K'_R \), the root locus, as a function of \( K'_R / K_R \), would be as shown in Figure 26. \( K'_R \) is the actual scaling between \( \Delta x(k) \) and \( e(k) \) that might occur in the radar. When we consider the radar part of the tracker, we will find out that \( K'_R \) is the variable gain of the discriminator. The circular path of the root locus is notional. The roots may not follow such a circular path, but some other more complicated path.
We will design the tracker so that when $K'_r/K_r = 1$, the closed-loop poles will be complex. Since $K'_r$ is almost always less than $K_r$, the closed-loop poles will remain complex. However, the response will become more sluggish because the poles will approach a pole pair at $z = 1$.

We want to determine $g$ and $h$ to give a desired, quadratic pole, response. Further, since we are familiar with characterizing response in the s-domain, we want to want to specify the loop bandwidth and damping coefficient in that domain and then translate the result to the z-domain. Specifically, we want to specify $\omega_n$ and $\zeta$ and use them to compute $g$ and $h$.

We start by writing an s-domain transfer function as

$$G_{cl}(s) = \frac{N(s)}{D(s)}.$$

(135)

Since the dominator of $G_{cl}(z)$ is quadratic, we want to consider a quadratic denominator for $G_{cl}(s)$. Thus, we write $D(s)$ in the form

$$D(s) = s^2 + 2\zeta\omega_n s + \omega_n^2.$$

(136)

The roots of $D(s)$ are located at

$$s_1, s_2 = -\zeta\omega_n \pm j\omega_n\sqrt{1-\zeta^2}.$$

(137)

To translate these to the z-domain we use the fact that $z = e^{so}$. With this, we get that the poles of $G_{cl}(z)$ are located at

$$z_1, z_2 = e^{-\omega_n z j\omega_n\sqrt{1-\zeta^2}}.$$

(138)

Alternately, we could have used the bilinear transform to find

$$z_1, z_2 = \frac{1+sT/2}{1-sT/2} \cdot \frac{1+s_T/2}{1-s_T/2}.$$

(139)

where $s_1$ and $s_2$ are as in (137). In this case the math is more complicated. Given that $f_c$ is usually less than $f_s/2 = 1/2T$ by a reasonable amount, the transformations of (138) and (139) should result in similar responses.

Given $z_1$ and $z_2$ we can write $G_{cl}(z)$ as

$$G_{cl}(z) = \frac{(g+h)z-g}{z^2+(g+h-2)z+1-g} = \frac{(g+h)z-g}{(z-z_1)(z-z_2)} = \frac{(g+h)z-g}{z^2-(z_1+z_2)z+z_1z_2}.$$

(140)

From this we get
\[ g + h - 2 = - \left( z_1 + z_2 \right) \]
\[ 1 - g = z_1 z_2 \]

which we can solve for \( g \) and \( h \).

### 3.3 \( g \)-h Filter Example

To illustrate this technique we consider an example. Consistent with the analog closed loop tracker example of Section 2.4, we will choose a one-sided bandwidth (cutoff frequency) of \( f_c = 1 \text{ Hz} \) and a damping coefficient of \( \zeta = 1/\sqrt{2} \). This combination yields \( \omega_n = \omega_c = 2\pi f_c = 2\pi \). We will choose a sample period of \( T = 0.05 \text{ s} \), or an update rate of 20 Hz. From (138) we get

\[ z_1, z_2 = \exp \left( \left( 2\pi/\sqrt{2} \pm j2\pi/\sqrt{2} \right) T \right) \]

(142)

Which we use in (141) to solve for \( g \) and \( h \).

A block diagram of the resulting tracker is shown in Figure 27. In the figure, the state variable equation in the track filter block is given by (129) and (130) with the \( F, G, \) and \( H \) defined in those equations. The specific values of \( g \) and \( h \) are computed as discussed above. A script used to implement the digitaltracker is shown in Figure 28.

Since we went to the trouble of deriving the transfer function of the track filter, we could have used the Matlab filter function in place of the state variable representation of Figure 27. However, it is not clear that this would have been any easier to program than the state variable implementation.

Plots for step, ramp and quadratic inputs are contained in Figures 29, 30 and 31. As expected, the tracker follows a step and ramp input with zero steady state error. It follows a quadratic input with a constant steady state error.

---

% Script to implement the g-h tracker
clear;close all
T = 0.05; % Update period
k = [0:60]; % define the stage (time) array
KR = 1; % Set KR = 1

% Select an input type
input = 'q' % s, r, q for step, ramp or quadratic
if input == 'q'
    x = ((k*T).^2)/2;
    titl = 'quadratic input';
elseif input == 'r'
    x = k*T;
    titl = 'ramp input';
else
    x = ones(size(k));
    titl = 'step input';
end

% establish the output arrays
xhat = zeros(size(k));
delx = zeros(size(k));

% Compute g and h
fc = 0.5; % One-sided bandwidth of 1 Hz
wn = 2*pi*fc; % Undamped natural frequency
zeta = 1/sqrt(2); % Damping coefficient
% Compute g and h
alp = zeta*wn;
beta = wn*sqrt(1-zeta^2);
% z1 = -(T*alp-2+j*T*beta)/(T*alp+2+j*T*beta)
% z2 = -(T*alp-2-j*T*beta)/(T*alp+2-j*T*beta)
z1 = exp(-alp*T+j*beta*T);
z2 = exp(-alp*T-j*beta*T);
g = 1 - z1*z2
h = -z1-z2+2-g

% Define the F, G and H matrices
F = [1 T;0 1];
G = [g+h;h/T];
H = [1 0];

% Define the initial state and initial input value
X0 = [0; 0];
e = KR*x(1);

% Run the loop
for kt=1:length(k)
% implement the track filter.  X is the state vector
    X = F*X0 + G*e;
    X0 = X;
    vx = H*X0; % compute the output of the track filter
    xhat(kt) = vx/KR; % compute xhat
    delx(kt) = x(kt)-xhat(kt); % compute the error
    e = KR*delx(kt); % compute e
end

% Plot the output
t = k*T;
fh1 = figure(1);
plot(t,xhat,t,x,'r','LineWidth',2)
ylabel('Position')
xlabel('Time (s)')
legend('xhat','x')
grid on
title(titl)
subplot(212)
plot(t,delx,'LineWidth',2)
ylabel('Error')
xlabel('Time (s)')
grid on
set(fh1,'Position',[435 285 560 368])

Figure 28 – Script for g-h Tracker

Figure 29 – Simulation Output for Step Input – g-h Tracker

Figure 30 – Simulation Output for Ramp Input – g-h Tracker
If we compare the plots of Figures 29, 30 and 31 to similar plots for the previous analog tracker example we note that the response of the g-h tracker seems faster and that the overshoot is larger. This indicates that the digital tracker bandwidth is larger than 1 Hz and the overshoot is larger than we would expect from a quadratic response with $\zeta = 1/\sqrt{2}$. Investigation indicates that this behavior is due to the closed loop zero at $g/(g+h)$. The evidence for this is shown in Figure 32, which is a closed loop frequency response. The blue curve is the response for the tracker that produced the responses of Figures 29, 30 and 31. The red curve is the frequency response of a hypothetical tracker with the closed loop zero removed. As can be seen, the hypothetical tracker has a bandwidth of 1 Hz. However, the closed loop zero pushes the bandwidth of the actual tracker out to about 2.5 Hz. The closed loop zero also causes the increased overshoot in the transient response. We could modify the design to bring the bandwidth of the actual tracker down to 1 Hz. We may also be able to reduce the overshoot by adjusting $g$ and $h$. However, we will not pursue that at this time.
3.4 g-h-k Tracker

One form of the g-h-k filter is (We will change the stage index (time) variable to $n$ to avoid confusion between it and the gain term, $k$.)

\[
\begin{align*}
 x_p(n+1) &= x_s(n) + T\dot{x}_s(n) + \frac{T^2}{2} \ddot{x}_s(n) \\
 \dot{x}_p(n+1) &= \dot{x}_s(n) + T\ddot{x}_s(n) \\
 \ddot{x}_p(n+1) &= \ddot{x}_s(n) \\
 x_s(n+1) &= x_p(n+1) + g(n+1)(y(n+1) - x_p(n+1)) \\
 \dot{x}_s(n+1) &= \dot{x}_p(n+1) + \frac{h(n+1)}{T}(y(n+1) - x_p(n+1)) \\
 \ddot{x}_s(n+1) &= \ddot{x}_p(n+1) + \frac{2k(n+1)}{T^2}(y(n+1) - x_p(n+1))
\end{align*}
\]

(143)

where $x_p(n), \dot{x}_p(n)$ and $\ddot{x}_p(n)$ are termed predicted position, rate and acceleration estimates and $x_s(n), \dot{x}_s(n)$ and $\ddot{x}_s(n)$ are smoothed position, rate and acceleration estimates. $y(n)$ is the position measurement. The filter gains are $g(n)$, $h(n)$ and $k(n)$, which, in general, are allowed to vary with time. In our case we assume they are fixed. $T$, which, in general, could also vary with $n$, is the update period of the tracker. The term $y(n+1) - x_p(n+1)$ is the error between the measured position and the predicted position. In a closed loop tracker, we assume that $y(n)$ is not directly measured; instead, the error sensor measures $\Delta x(n) = y(n) - x_p(n)$ and reports a scaled version as $e(n) = K_n\Delta x(n)$ (see Figure 22). In a closed loop tracker we are not explicitly interested in $x_s(n), \dot{x}_s(n)$ and $\ddot{x}_s(n)$.

With the above we rewrite (143) to eliminate $x_s(n), \dot{x}_s(n)$ and $\ddot{x}_s(n)$, and replace $y(n) - x_p(n)$ with $e(n)$. If we substitute the $x_s(n+1), \dot{x}_s(n+1)$ and $\ddot{x}_s(n+1)$ into the $x_p(n+1), \dot{x}_p(n+1)$ and $\ddot{x}_p(n+1)$ equations, with the appropriate time shift, we get

\[
\begin{align*}
 x_p(n+1) &= x_p(n) + ge(n) + T\left[\dot{x}_p(n) + \frac{h}{T}e(n)\right] + \frac{T^2}{2}\left[\ddot{x}_p(n) + \frac{2k}{T^2}e(n)\right] \\
 \dot{x}_p(n+1) &= \dot{x}_p(n) + \frac{h}{T}e(n) + T\left[\ddot{x}_p(n) + \frac{2h}{T^2}e(n)\right] \\
 \ddot{x}_p(n+1) &= \ddot{x}_p(n) + \frac{2k}{T^2}e(n)
\end{align*}
\]

(144)
If we write this in matrix form we get
\[
X_p(n+1) = FX_p(n) + Ge(n) \quad (145)
\]
where
\[
F = \begin{bmatrix}
1 & T & T^2/2 \\
0 & 1 & T \\
0 & 0 & 1 \\
\end{bmatrix}
\quad
G = \begin{bmatrix}
g + h + k \\
(h + 2k)/T \\
2k/T^2 \\
\end{bmatrix}
\quad
X_p(n) = \begin{bmatrix}
x_p(n) \\
\dot{x}_p(n) \\
\end{bmatrix}
\quad
The filter output is
\[
v_x(n) = H^T X_p(n) \quad (146)
\]
with
\[
H^T = [1 \ 0 \ 0].
\]

We now want to find the transfer function of the g-h-k filter. From the previous section we have
\[
G_p(z) = H^T (zI - F)^{-1} G. \quad (147)
\]

We can compute the inverse as
\[
(zI - F)^{-1} = \begin{bmatrix}
z & 0 & 0 \\
0 & z & 0 \\
0 & 0 & z \\
\end{bmatrix} - \begin{bmatrix}
1 & T & T^2/2 \\
0 & 1 & T \\
0 & 0 & 1 \\
\end{bmatrix} = \begin{bmatrix}
z-1 & -T & -T^2/2 \\
0 & z-1 & -T \\
0 & 0 & z-1 \\
\end{bmatrix}
\]
\[
= \begin{bmatrix}
1 & T & T^2(z+1) \\
z-1 & (z-1)^2 & 2(z-1)^2 \\
0 & 1 & (z-1)^2 \\
0 & 0 & 1 \\
\end{bmatrix}
\quad (148)
\]

Using this in (145) gives
\[
G_p(z) = \begin{bmatrix}
\frac{1}{z-1} & \frac{T}{(z-1)^2} & \frac{T^2(z+1)}{2(z-1)} \\
0 & \frac{1}{z-1} & \frac{T}{(z-1)^2} \\
0 & 0 & \frac{1}{z-1}
\end{bmatrix}
\begin{bmatrix}
g + h + k \\
(h + 2k)/T \\
2k/T^2
\end{bmatrix}
\]

\[
= \frac{1}{z-1} \frac{T}{(z-1)^2} \frac{T^2(z+1)}{2(z-1)} \begin{bmatrix}
g + h + k \\
(h + 2k)/T \\
2k/T^2
\end{bmatrix}
\]

A block diagram of the resulting tracker is shown in Figure 33.

We note that \(G_p(z)\) has three poles at \(z=1\). This means that we have a Type 3 servo, which should be able to track a step, ramp and quadratic input with zero steady state error.

Given that the numerator of \(G_p(z)\) contains a quadratic, there will be two open loop zeros. Depending on the specific values of \(g, h\) and \(k\), they could be real or complex. Notional plots of the root locus for the two cases are shown in Figures 34 and 35. For the case where the two zeros are real, one of the poles at \(z=1\) goes directly to the zero nearest \(z=1\). The other poles travel on the notional circle and return to the real axis to the left of the other zero. One pole then goes to the zero and the other goes to \(-\infty\). For the complex zero case, one of the poles at \(z=1\) goes directly to \(-\infty\) and the other two poles go to the complex zeros. Again, the shape of the locus will most likely not look exactly like Figures 34 and 35. The only sure way to determine the actual shape of the root locus would be to find the poles of \(G_{cl}(z)\).
Figure 34 – Root Locus of g-h-k Tracker with Two, Real Open Loop Zeros

Figure 35 – Root Locus of g-h-k Tracker with Complex Open-loop Zeros

As with the g-h filter we want to find $G_{cl}(z)$ so we can determine $g$, $h$ and $k$ to meet our design requirements. From (87) we have

$$G_{cl}(z) = \frac{G_{ol}(z)}{1 + G_{ol}(z)}.$$  \hspace{1cm} (150)

Substituting from (149) and manipulating gives
\[
G_{cl}(z) = \frac{(g+h+k)z^2 + (k-h-2g)z + g}{(z-1)^3 + (g+h+k)z^2 + (k-h-2g)z + g} = \frac{(g+h+k)z^2 + (k-h-2g)z + g}{(z-1)^3 + (g+h+k)z^2 + (k-h-2g)z + g} = \frac{(g+h+k)z^2 + (k-h-2g)z + g}{z^3 + (g+h+k-3)z^2 + (k-h-2g+3)z + g-1}
\]

Since \(G_{cl}(z)\) has three poles, we need to decide where to place them. We will want two of the poles to be complex to get an underdamped response. As with the g-h filter, we will specify \(\omega_n\) and \(\zeta\) of an s-domain function and map the resulting complex pole pair to the z domain using \(z = e^{st}\). The third pole will be real. Through experimentation, we found that the third pole should be close to \(z = 1\) (i.e. \(s = 0\)). However, it should not be placed at \(z = 1\). If the pole is placed at \(z = 1\), the closed loop transfer function will be marginally stable, which could cause the output to grow without bound for some bounded inputs, such as a step input. Also, if the pole is placed too close to \(z = 1\), the response will be sluggish. In the example considered below, we placed the pole at \(z = e^{st}\) where we set \(s\) to -0.5π.

Once we have the z-domain poles we need to relate them to g, h and k. We can write the denominator of \(G_{cl}(z)\) in terms of the poles as

\[
D(z) = (z-z_1)(z-z_2)(z-z_3) = z^3 - (z_1 + z_2 + z_3) z^2 + (z_1 z_2 + z_1 z_3 + z_2 z_3) z - z_1 z_2 z_3 .
\]

If we relate this to the denominator of \(G_{cl}(z)\) in (151) we get

\[
g + h + k - 3 = -(z_1 + z_2 + z_3), \quad (153)
\]

\[
k - h - 2g + 3 = z_1 z_2 + z_1 z_3 + z_2 z_3 \quad \text{and} \quad (154)
\]

\[
g - 1 = -z_1 z_2 z_3 . \quad (155)
\]

We can solve (155) for g. Once we have g, we can rewrite (153) and (154) as

\[
h + k = -(z_1 + z_2 + z_3) + 3 - g \quad \text{and} \quad (156)
\]

\[
k - h = z_1 z_2 + z_1 z_3 + z_2 z_3 - 3 + 2g . \quad (157)
\]

If we add (156) and (157) we get

\[
k = \frac{1}{2} \left( z_1 z_2 + z_1 z_3 + z_2 z_3 - z_1 - z_2 - z_3 + g \right) . \quad (158)
\]

We can then use (156) or (157) to solve for h. From (156) we get

\[
h = -(z_1 + z_2 + z_3) + 3 - g - k . \quad (159)
\]
3.5 g-h-k Filter Example

As with the g-h tracker example, we choose a one-sided bandwidth (cutoff frequency) of $f_c = 1$ Hz and a damping coefficient of $\zeta = 1/\sqrt{2}$. This combination yields $\omega_n = \omega_c = 2\pi f_c = 2\pi$. We will choose a sample period of $T = 0.05$ s, or an update rate of 20 Hz. With this we get two of the poles of $G_{cz}(z)$ as

$$z_1, z_2 = \exp\left(\left(2\pi/\sqrt{2} \pm j2\pi/\sqrt{2}\right)T\right). \tag{160}$$

To find the third pole we use $s = -0.5\pi$ and get

$$z_3 = \exp\left(-0.5\pi T\right). \tag{161}$$

We will then use (155), (158) and (159) to compute $g$, $h$ and $k$. We will perform these calculations in the simulation script.

The script for the g-h-k tracker will be similar to the script for the g-h tracker except for the definitions of $g$, $h$, $k$, $F$, $G$ and $H^T$. We will leave it as a homework assignment.

Plots that I obtained from my script are shown in Figures 36, 37 and 38 for step, ramp and quadratic inputs. The response time is similar to that for the g-h filter except that this time the steady state error for the quadratic input is headed toward zero, as expected.

![Figure 36 – Simulation Output for Step Input – g-h-k Tracker](image)
Figure 37 – Simulation Output for Ramp Input – g-h-k Tracker

Figure 38 – Simulation Output for Quadratic Input – g-h-k Tracker