4. Design of Discrete Control Systems

In previous chapters, we learned the fundamental concepts of discrete time systems (ch. 1), the mathematical tools to describe those systems (ch. 2), and how to apply those tools to analyse a system's behaviour (stability and frequency response) (ch. 3). In this last chapter, we will finally learn how to **control**. In other words, we will learn how to modify the system's behaviour to achieve a more desirable response.

This process can be considered in two steps: 1) 'design' the controller's pulse transfer function to achieve the desired system characteristics, 2) 'implement' (or 'realize') the design. This means putting the design into a form suitable for hardware or software implementation, which of course must work in the time domain, rather than the z domain.

4.1 Equivalent Continuous Time Design

Most control design tasks have the same general form:

- There is some system ('plant' or 'process') with transfer function ***G(s)***, which may have undesirable characteristics (instability or oscillation).
- We want to introduce a controller ***C(s)*** to alter these characteristics. As described in previous chapters, we may use an open-loop control system…

$$H(s) = C(s)G(s)B(s)$$

... or a closed-loop control system....

$$H(s) = \frac{C(s)G(s)}{1 + B(s)C(s)G(s)}$$

- So we must choose ***C(s)*** (and perhaps ***B(s)***, but not in this course) to ensure that ***H(s)*** has better response characteristics than the uncontrolled system, ***G(s)***.

To design a digital controller it is more appropriate to work in the z-domain (i.e. find ***C(z)*** instead of the continuous time controller ***C(s)***). In previous chapters, we learned how to transform ***G(s)*** into ***G(z)***, so that we can work in the z-domain. Some people are more familiar and comfortable with working in the s-domain. They may use an alternative approach, known as **equivalent continuous time design**. In this approach, we first design ***C(s)*** to have the desired characteristics, then apply the z-transform.

It must always be remembered, however, that if a discrete time controller is to be used, the presence of a zero order hold in the system must be taken into account.

Furthermore, it must be assured that the design can be implemented, i.e., y(n) must not depend on y(n+1) because we don't know y(n+1) at the time when we need to calculate y(n). In the z domain, we can easily avoid this problem by seeking a form that only contains negative powers of *z* (remember that z^{-1} represents a delay of one sample). E.g. For the linear difference equation linking input (error) signal ***e(k)*** and output (control) signal ***u(k)***

$$a_0\,u(k) + a_1\,u(k-1) + \ldots a_m\,u(k-m) = b_0\,e(k) + b_1\,e(k-1) + \ldots b_n\,e(k-n)$$

we can obtain the ***z*** domain transfer function

$$C(z) = \frac{U(z)}{E(z)} = \frac{b_0 + b_l z^{-l} + b_2 z^{-2} + b_3 z^{-3} \ldots + b_n z^{-n}}{a_0 + a_1 z^{-1} + a_2 z^{-2} + a_3 z^{-3} \ldots + a_m z^{-m}}$$

where $a_0 .. a_m$, $b_0 .. b_n$ are the equation coefficients and the difference equation is of order ***m*** or ***n***, whichever is greater.

Because of the practical concerns mentioned above, and others, it is often more straightforward to design discrete time controllers in the z domain rather than using equivalent continuous time design (once you're comfortable with the mathematics).

In sections 4.3 and 4.4 we will see some examples of how to design a specific controller for a known system. First we will learn how to actually implement a chosen controller, to bring further practical concerns to our attention.

4.2 Realization and Implementation

Digital control systems can be realized either in software as a computer program or in hardware using digital circuitry. For a given transfer function or linear difference equation, the **realization diagram** of the necessary arithmetic operations, i.e., the physical layout for a digital controller, is determined. The same realization diagram can be useful for both hardware and software implementations. The basic operations include a unit delay, adder/subtractor, constant multiplier, branching operation, and signal multiplier.

4.2.1 Direct Form Realization

The realization can take different forms, either direct, cascade, or parallel realization forms. We will start this chapter by looking at the two simpler direct realization forms. Assuming the transfer function of interest to be of the form

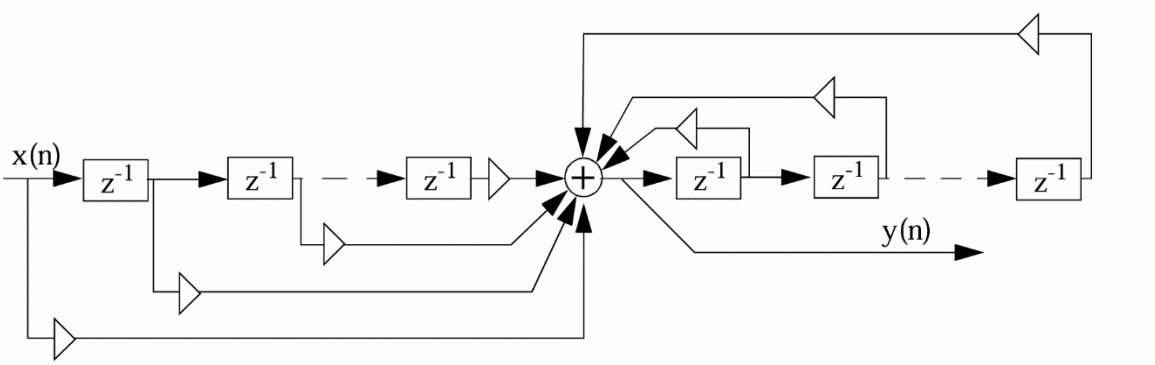
$$H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{i=0}^{n} b_i z^{-i}}{1 + \sum_{i=1}^{m} a_i z^{-i}},$$

the corresponding difference equation can be written as

$$y(n) = \sum_{i=0}^{n} b_i x(n-i) - \sum_{i=1}^{m} a_i y(n-i).$$

Note that this form requires us to first ensure that a_0 (as featured on the previous page) is 1. This is common practice as it avoids having to apply an extra gain to the output y(n) (see figure below).

Such a difference equation can be directly realized using the **direct form 1** method as shown below.



The **direct form 1** implementation can require up to ***2k*** delay operations to provide the successively time shifted values of ***x(n)*** and ***y(n)*** for a ***k***th order system.

Each delay operation (z^{-1} block) represents an additional register (unit of memory) required to implement the system. It is therefore desirable to implement the system in a way that minimises the number of z^{-1} blocks, so we use less memory and/or fewer hardware components.

The number of necessary delay elements can be reduced to ***k*** by using a ***direct form 2*** implementation. Writing

$$Y(z) = H(z)X(z) = \frac{N(z)X(z)}{D(z)},$$

we define a new variable

$$W(z) = \frac{X(z)}{D(z)}$$

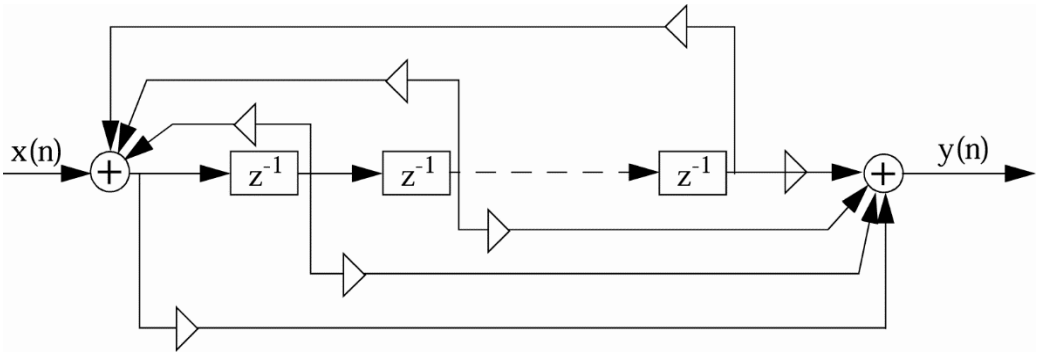
so that ***Y(z)*** and ***X(z)*** can be expressed as

$$Y(z) = N(z)W(z)$$

The inverse transforms can then be written as

$$w(n) = x(n) - \sum_{i=1}^{k} a_i w(n-i) \qquad \text{and} \qquad y(n) = \sum_{i=0}^{k} b_i w(n-i),$$

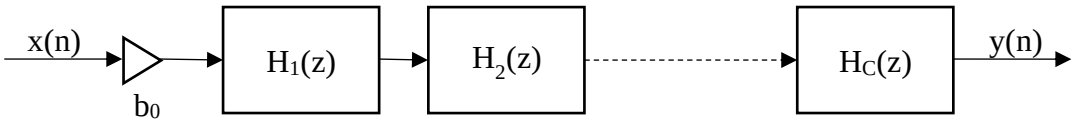
The layout of the **direct form 2** realization or 'canonic form' is shown below.



4.2.2 Cascade Form Realization

The use of a direct form realization can have adverse effects on the performance of the system. Storing numbers in digital form requires the number to be quantized (rounded), so that it can be stored using a certain number of bits. It has been shown that the direct implementation of a higher order linear difference equation can lead to an accumulation of quantization errors, worsening the controller's performance.

This effect can be significantly reduced by implementing high order difference equations in terms of the product or sum of lower order difference equations. Therefore **cascade** and **parallel** implementations are more suitable for the implementation of higher order transfer functions. To implement these forms, the transfer function is decomposed into first and second order elements, which are either connected in series (cascade form) or parallel, as shown below.



Cascade Form Implementation

Each block $H_i(z)$ contains a direct form implementation of a lower order transfer function. Hence, to implement the cascade form we must first decompose ***H(z)*** into the product of several simpler transfer functions

$$H(z) = b_0 H_1(z) H_2(z) \cdots H_C(z) = b_0 \prod_{i=1}^{C} H_i(z) .$$

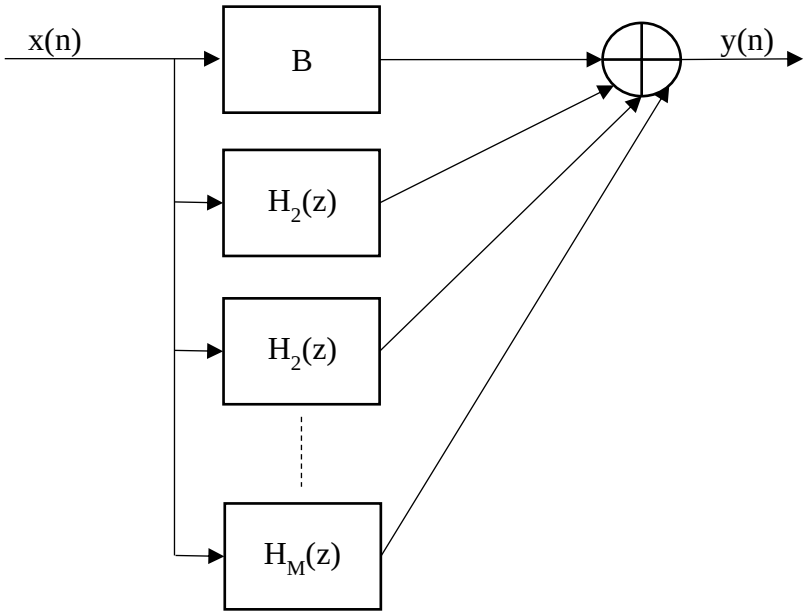
Mostly, the individual transfer functions are chosen as first and second order sections,

$$H_i(z) = \frac{b_{i0} + b_{i1} z^{-1}}{1 + a_i z^{-1}}$$

$$H_j(z) = \frac{b_{j0} + b_{j1} z^{-1} + b_{j2} z^{-2}}{1 + a_{j1} z^{-1} + a_{j2} z^{-2}}$$

with the overall gain b_0 as a constant multiplier. For known poles and zeros of the overall transfer function, real poles and zeros can be used to form first order sections. Pairs of complex conjugate poles (or zeros) can be multiplied together to produce second order sections with real coefficients. Obviously, a pair of real *poles* might be grouped with a pair of complex conjugate *zeros* to obtain a second order section, or vice versa. The first or second order transfer function sections can then be realized in direct form as shown above.

4.2.3 Parallel Form Realization



Parallel Form Implementation

The parallel form realization can be obtained by decomposing H(z) into the sum of first and second order transfer functions and a constant term expressed as

$$H(z) = B + H_1(z) + H_2(z) + \ldots + H_C(z) = B + \sum_{i=1}^{C} H_i(z).$$

Due to the constant term B the first and second order sections can be chosen in the simpler forms

$$H_i(z) = \frac{b_{i0}}{1 + a_{i1} z^{-1}}$$

$$H_j(z) = \frac{b_{j0} + b_{j1} z^{-1}}{1 + a_{j1} z^{-1} + a_{j2} z^{-2}}$$

The denominator polynomials can be obtained by grouping pairs of complex conjugate poles to obtain real constants and using single real poles. To determine the numerator polynomials a partial fraction expansion similar to chapters 2.5 and 3.3 must be carried out.

4.3 Discrete PID Controller Design

4.3.1 Equivalent Continuous Time Design

In section 4.1, we learned that 'equivalent continuous time design' refers to the process of designing our discrete time controller ***C(z)*** by first designing ***C(s)*** then transforming it to the z-domain.

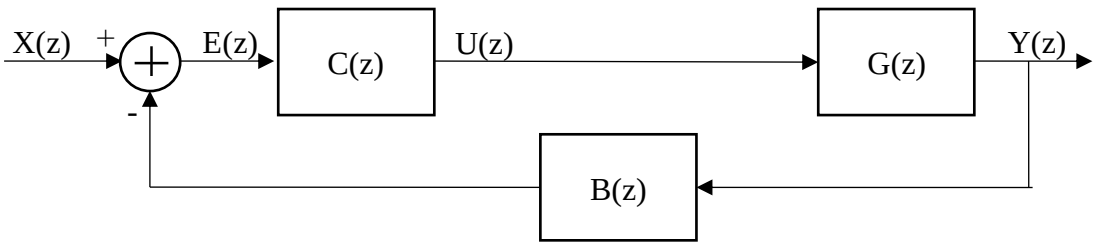
Typically, calculating this transformation exactly is not achievable due to the loss of information between continuous- and discrete-time representations of signals. Several approximate methods exist for producing equivalent ***z*** transfer functions to known continuous time transfer functions, including differential approximations using finite differences, bilinear transformations, and **pole/zero mapping** (also known as the Matched Poles/Zeroes Method). All of these methods are approximate, but the last method, pole/zero mapping, gives the least error and for larger transfer functions is just as easy as the other methods.

The pole/zero mapping method simply consists of noting the positions of the zeroes and poles in ***C(s)***, using the relationship $z = e^{sT}$ to map these positions to the z-plane, and multiplying these poles and zeros together to form the z domain transfer function, ***C(z)***. ***C(z)*** will not behave exactly as ***C(s)*** would, but it will exhibit the same stability characteristics; as we saw in chapter 3, these characteristics are determined by the pole locations. The time domain and frequency domain response of such a matched transfer function is then also close to that of the original system.

4.3.2 Direct Discrete Time PID Design0

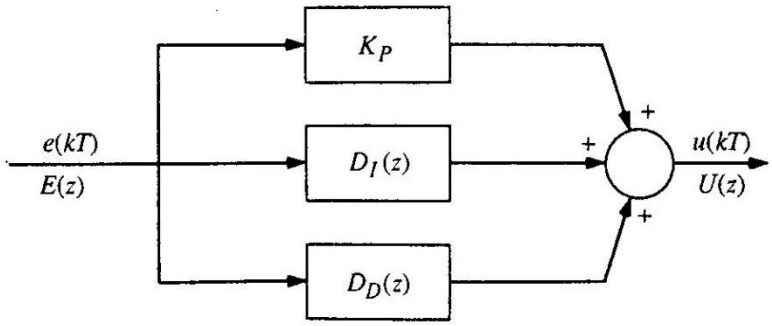
As noted in section 4.1, the alternative to equivalent continuous time controller design is a direct discrete time controller design using a discrete time model (***G(z)***) of the system to be controlled. The zero order hold in this case is considered to be part of the controlled system and a transfer function must first be derived for this combined component (see section 3.1). The direct discrete time controller design can thus be made of a realizable form immediately and requires no more complex calculations for pole assignment than a continuous time design. Discrete time design is for these reasons often chosen in preference where it is easy to find ***G(z)***.

PID Controllers (Proportional-Integral-Derivative Controllers) are a widely used general form of controller, consisting of three branches (see figure on next page). Varying the gains of these branches allows the designer to achieve a desirable compromise between the steady-state error, speed of response, and mean error. Using the labels drawn below, we will now explore the structure of a PID controller, ***C(z)***.



The controller is considered to be broken down into three separate sections, and any possible zero offset is neglected. In the following sections, we will generate the difference equations and corresponding z transforms for three types of control, proportional C_p, integral C_i, and derivative C_d.

$$C = C_p + C_i + C_d$$



The controllers may be defined as

$$c_p = K_p e \quad c_i = K_i \int e dt \quad c_d = K_d \frac{de}{dt}$$

with constants K_p, K_i, K_d, and error term e.

4.3.3 Proportional Term

The proportional term can trivially be written as

$$u_p(n) = K_p e(n)$$

with the constant z domain transfer function

$$C_p(z) = \frac{U_p(z)}{E(z)} = K_p$$

4.3.4 Integral Term

The integral is the area under the graph of ***e*** vs ***t***, starting at ***t = 0***. If the time step is sufficiently small, we can approximate the area under the graph using "backward-rectangular integration", adding up the incremental areas $A(n) = e(n-1)T$ for each sampling interval.

Hence the integral part is approximated as

$$u_i(n) = K_i \sum_{k=0}^{n-1} e(k)T$$

which leads to the state update equation

$$u_i(n) = u_i(n-1) + K_i e(n-1)T$$

with the ***z*** domain transfer function

$$C_i(z) = \frac{U_i(z)}{E(z)} = K_i \frac{T}{z-1}.$$

Please note that the choice of a different approximation of the integral, e.g., forward-rectangular, would lead to a different state equation and transfer function.

4.3.5 Derivative Term

The derivative term is approximated in first order as the gradient of the line between two points. Hence, the gradient is given by $\frac{de}{dt} = \frac{e(n) - e(n-1)}{T}$, and the derivative term as

$$u_d = K_d \frac{e(n) - e(n-1)}{T}.$$

with the ***z*** domain transfer function

$$/C_d(z) = \frac{U_d(z)}{E(z)} = K_d \frac{z-1}{Tz}$$

4.3.6 Combined Term Equation

The z transform for the combined PID controller is thus

$$C(z) = \frac{U(z)}{E(z)} = C_P(z) + C_i(z) + C_d(z)$$

$$= K_p + K_i \frac{T}{z-1} + K_d \frac{z-1}{Tz}$$

$$= \frac{K_p z(z-1) + K_i Tz + K_d / T(z-1)^2}{z(z-1)}$$

$$= \frac{\left(K_p + \frac{K_d}{T}\right)z^2 + \left(-K_p + K_i T - \frac{2K_d}{T}\right)z + \frac{K_d}{T}}{z^2 - z}$$

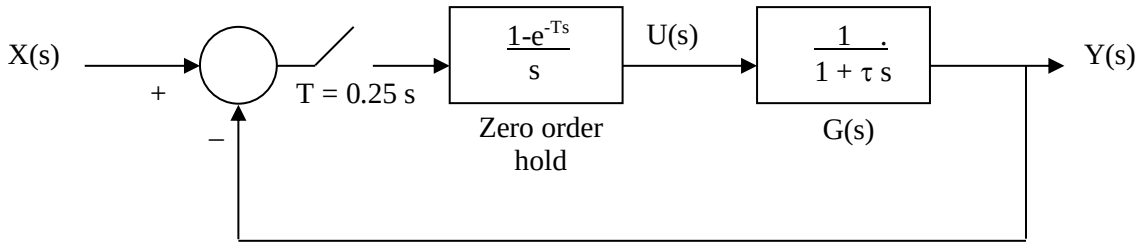
$$= \frac{\left(K_p + \frac{K_d}{T}\right) + \left(-K_p + K_i T - \frac{2K_d}{T}\right)z^{-1} + \frac{K_d}{T} z^{-2}}{1 - z^{-1}}$$

4.4 Digital Control Applications

[After Golten & Verwer]

For a fast enough sampling rate ($\omega_S = 2\pi/T \geq 30\omega_G$), the design of the controller, e.g., a PID controller, can be done in most cases as for an analogue controller, using the familiar Laplace domain notation and description. In this short section we will look at some of the special effects encountered when using digital control and how the controller can be designed directly in the digital domain. This treatment will be rather brief, and for real applications advanced methods like state-space description for multiple input-output systems should be considered from the reading list provided.

4.4.1 First order sampled-data system with proportional control



Looking at the case where the plant is a first order system

$$G(s) = \frac{Y(s)}{U(s)} = \frac{1}{1+\tau s} = \frac{1/\tau}{s+1/\tau}$$

we have shown in chapter 3.1 that the impulse transfer function of the zero order hold and first order system combined is given by

$$G'(z) = \frac{1-z^{-1}}{s} Z\left[\frac{1/\tau}{(s+1/\tau)}\right] = \frac{1-\beta}{z-\beta}.$$

For ***T*** = 0.25 seconds and a time constant τ of 1 second

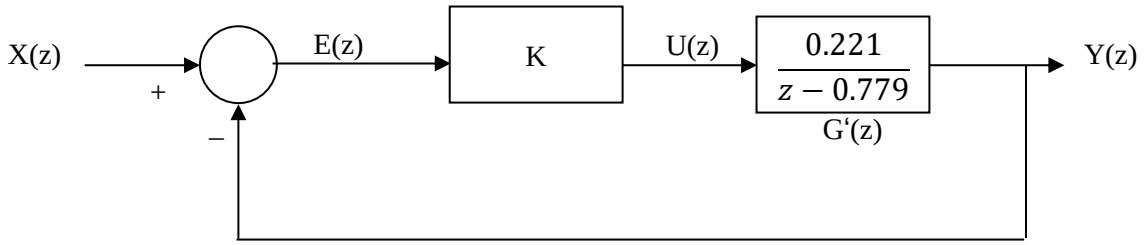
$$\beta = e^{-T/\tau} = e^{-0.25} = 0.779$$

and the open loop impulse transfer function becomes

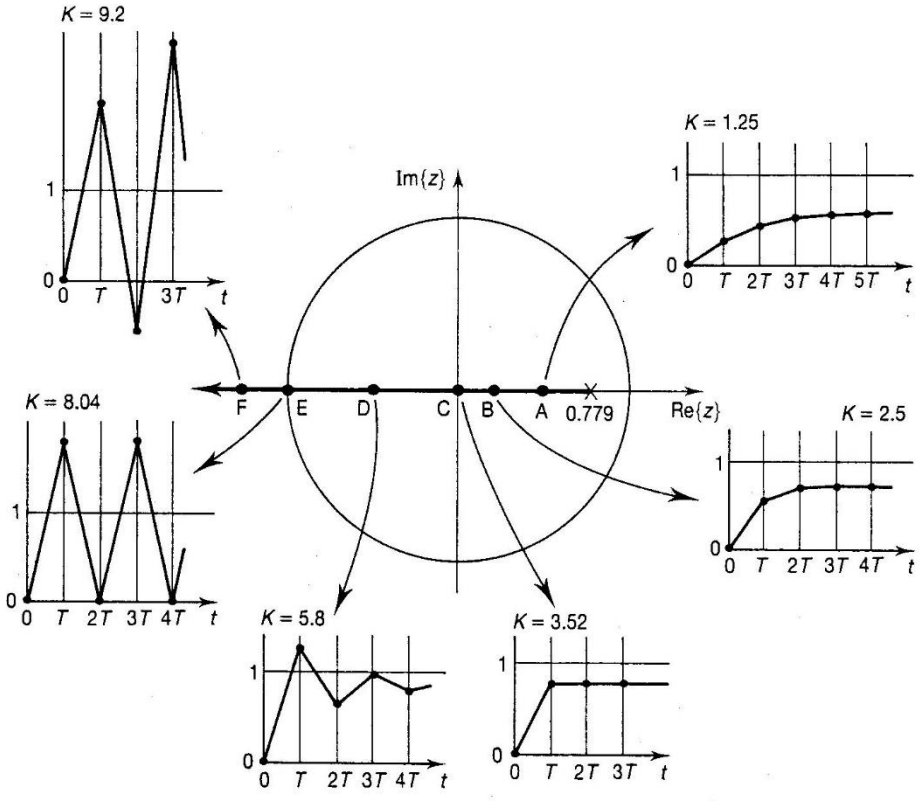
$$G'(z) = \frac{0.221}{z-0.779}.$$

Adding a digital proportional controller (***C'(z)*=*K***), the closed loop transfer function can be calculated as

$$H(z) = \frac{Y(z)}{U(z)} = \frac{C'(z)G'(z)}{1 + C'(z)G'(z)} = \frac{0.221K}{z - 0.779 + 0.221K}$$



The system has a single pole, which can be placed anywhere on the real axis in the root locus diagram by varying the gain ***K***.



For small gain values the pole is on the positive real axis, showing a (desired) reduction in offset and settling time. For ***K*** = 3.52 the pole lies on the origin, and the system behaves like a pure delay of one sample, which we call **deadbeat control**. This exact settling of the transient response looks very attractive, but it can be very dependent on an exact plant model and might necessitate an excessive control effort. If the gain is set just slightly too high, the pole will lie on the negative real axis. From our understanding of the z-plane (section 3.2), we know that this implies that an oscillation at the Nyquist frequency has been introduced (like for point D above), and this oscillation is likely to be undesireable in practical systems. Often a more conservative control approach with pole placement on the positive real axis is preferable.

As the gain increases beyond ***K*** = 3.52, the decay of the Nyquist frequency oscillation becomes slower. At ***K*** = 8.04 the oscillation has constant amplitude (no decay; marginal stability) . For ***K*** > 8.04, the system is unstable.

This instability would not occur in the equivalent continuous time system under proportional control (see for yourself by considering the closed loop transfer function ***KG(s)/(1+KG(s))***, with ***G(s)*** defined as before). The instability is one of the possible effects of digital control that need to be taken into account.

Please note that with only proportional control, there will always be an offset in the final value. This can be shown using the final value theorem (section 2.3), assuming ***x(n)*** is a unit step input. Our transform table tells us that, for a unit step input…

$$X(z) = \frac{z}{z-1}$$

The final value theorem tells us

$$\lim_{n\to\infty}\left[y(n)\right] = \lim_{z\to 1}\left[\frac{z-1}{z}Y(z)\right]$$

$$= \lim_{z\to 1}\left[\frac{z-1}{z}X(z)H(z)\right]$$

$$= \lim_{z\to 1}\left[\frac{z-1}{z}\frac{z}{z-1}H(z)\right] = \lim_{z\to 1}\left[H(z)\right]$$

Note that the above simplification of the final value theorem is always applicable for a unit step input.

Continuing for our system…

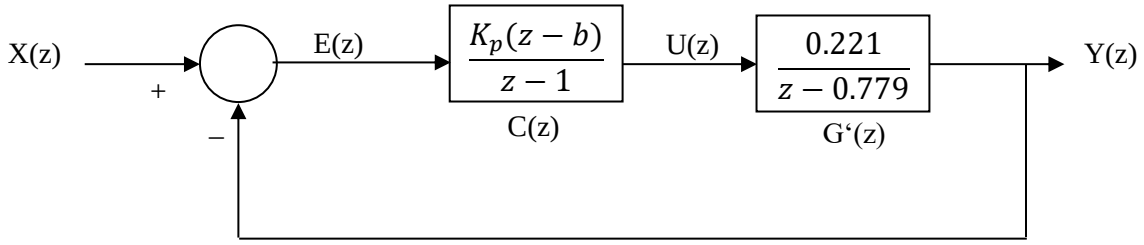
$$\lim_{n\to\infty}\left[y(n)\right] = \lim_{z\to 1}\left[\frac{0.221K}{z-0.779+0.221K}\right] = \frac{0.221K}{0.221+0.221K} = \frac{K}{K+1}$$

This ‘final value’ is clearly offset from 1 for all values of K.

4.4.2 First-order sampled-data system with proportional and integral (PI) control

For the first order system studied above, we will now consider the use a combined proportional and integral (PI) control algorithm, instead of just proportional control. The z domain transfer function of the PI controller was derived in chapter 4.3 as

$$C(z) = \frac{U(z)}{E(z)} = K_p + K_i \frac{T}{z-1} = \frac{K_p(z-1) + K_i T}{z-1} = \frac{K_p(z-b)}{z-1}, \text{ where } b = 1 - \frac{K_i}{K_p} T$$



The overall open loop impulse transfer function is now

$$C'(z)G'(z) = \frac{0.221K_p(z-b)}{(z-1)(z-0.779)}.$$

An interesting choice of parameters is to find the value of K_i that makes b = 0.779. In this case, z-b in the numerator will cancel out z-0.779 in the denominator. The open loop transfer function becomes…

$$C'(z)G'(z) = \frac{0.221K_p}{z-1}.$$

The closed loop transfer function can then be calculated as

$$H(z) = \frac{Y(z)}{U(z)} = \frac{C'(z)G'(z)}{1 + C'(z)G'(z)} = \frac{0.221K_p}{z - 1 + 0.221K_p}$$

Now the system has only one pole, on the real axis. As with the proportional control example in 4.4.1, we can vary K_p to choose the pole's position on the real axis. But our careful selection of K_i has yielded an additional benefit: the offset of the simple proportional controller has been eliminated. This is shown by re-examining the Final Value Theorem:

$$\lim_{z \to 1} H(z) = \lim_{z \to 1} \frac{0.221K}{z - 1 + 0.221K} = \frac{0.221K}{0.221K} = 1$$

4.4.3 Direct Discrete Controller Design

Digital controllers allow the direct design without referencing back to an underlying analogue design. The problem is then posed directly as finding a control algorithm ***C'(z)*** that gives a desired closed loop performance ***F(z)***.

$$H(z) = \frac{Y(z)}{U(z)} = \frac{C'(z)G'(z)}{1 + C'(z)G'(z)} = F(z)\,.$$

From this C'(z) can be found as

$$C'(z) = \frac{1}{G'(z)} \frac{F(z)}{1 - F(z)}\,.$$

Several constraints and performance requirements need to be fulfilled and are discussed in more detail in [Golten & Verwer, Ch. 9.5].

We will treat this using an example to find a suitable controller (sample interval 10 seconds) for a plant with first order dynamics (time constant 45 seconds) and a transport delay of 5 seconds. The process can be modelled as

$$G'(z) = \frac{0.105(z + 0.895)}{z(z - 0.8)}\,.$$

We could try and achieve a closed loop transfer function as

$$F(z) = \frac{0.5}{z - 0.5}$$

with a stable pole at ***z = 0.5*** and the numerator as ***0.5*** to avoid steady-state offset.

Solving for ***C'(z)*** we get

$$C'(z) = \frac{1}{G'(z)} \frac{F(z)}{1 - F(z)} = \frac{4.76z(z - 0.8)}{(z - 1)(z + 0.895)}\,.$$

The transfer function of the whole system will now be exactly as prescribed. However, the controller has a (stable) ringing pole at ***z = -0.895***, so the signal from the controller to the plant (actuator) will oscillate at the Nyquist frequency. This oscillatory actuation is likely to result in adverse effects such as noisy operation and accelerated wear. Hence, it is not a desirable solution in practice. This example demonstrates that direct discrete design is not always as straightforward as we might expect, and why common forms of controller such as the PID are often chosen in favour of the direct approach.

We may repeat the direct controller design without cancelling the problem plant zero, thus avoiding the introduction of the ringing pole. In other words, we can seek a closed loop transfer function as

$$F(z) = \frac{b(z+0.895)}{z(z-0.5)}$$

incorporating the plant zero at ***z = -0.895***. The constant ***b*** can be found from the steady-state requirement ***F(1) = 1*** as ***b = 0.264*** (remember our simplified form of the final value theorem, section 4.4.1).

$$F(z) = \frac{0.264(z+0.895)}{z(z-0.5)}$$

Solving for ***C'(z)*** we now obtain

$$C'(z) = \frac{1}{G'(z)}\frac{F(z)}{1-F(z)} = \frac{z(z-0.8)}{0.105(z+0.895)}\frac{0.264(z+0.895)}{(z-1)(z+0.236)} = \frac{2.514z(z-0.8)}{(z-1)(z+0.236)}$$

Although we still have a pole on the negative real axis, the fact that it's much closer to zero implies that the associated decay is much faster. In practice, the effects of the Nyquist frequency oscillation will be much reduced.