

and software such as MATLAB, which can directly solve the characteristic equation and give the location of the poles, rather than their number in the left and right half planes.

We shall further discuss stability from the viewpoint of *state-space* methods (as opposed to the classical frequency and Laplace domain methods) in Chapter 3. Before the advent of state-space methods, it was customary to employ *graphical methods* in the Laplace or the frequency domain for determining whether a closed-loop system was stable. The Bode plot is a graphical method that we have already seen. However, we are yet to discuss Bode plots from the stability viewpoint. Other graphical methods are the *root-locus*, the *Nyquist* plot, and the *Nichols* plot. Design using graphical methods is an instructive process for single-input, single-output, linear, time-invariant systems. Also, sometimes the graphical methods help in visualizing the results of a *multi-variable state-space* design (Chapters 5–7), and in that sense they have an important place in the modern control system design.

2.9 Root-Locus Method

To understand the root-locus method of stability analysis, reconsider the single-input, single-output feedback control system of Figure 2.32. The closed-loop transfer function, $Y(s)/Y_d(s)$, given by Eq. (2.139) is said to have a *return difference function*, $1 + G(s)H(s)$, which is the same as the *characteristic polynomial* of the *closed-loop system*. The *return difference* is a property of the feedback loop, and comes from the fact that if $Y_d(s) = 0$, then $Y(s) = G(s)U(s)$ and $U(s) = -H(s)Y(s)$; combining these two equations we can write $Y(s) = -G(s)H(s)Y(s)$, i.e. $Y(s)$ has *returned* to itself, or $[1 + G(s)H(s)]Y(s) = 0$. (The function $G(s)H(s)$ is called the *return ratio*.) The root-locus method determine stability simply by investigating whether the return difference becomes zero for any value of s in the *right* half of the Laplace domain (i.e. values of s with positive real parts). If $1 + G(s)H(s) = 0$ for some s in the right-half plane, it implies that there must be a closed-loop pole with *positive real part*, thus (according to stability criteria 2) the closed-loop system must be *unstable*. By drawing a *locus* of the each of the *roots* of the *return difference function*, $1 + G(s)H(s)$, as a *design parameter* is varied, we can find those values of the design parameter for which the system is stable (i.e. for which the loci do not enter the right half s -plane). How far away the locus closest to the imaginary axis is from crossing over into the right half s -plane also indicates how far away the system is from being unstable – in other words, the *stability margin* of the closed-loop system. A pole on the imaginary axis indicates *zero stability margin*, which we called the case of stable but not asymptotically stable system in stability criteria 3. By the same criteria, more than one pole at the *origin* indicates instability. If we see the loci of one (or more) poles crossing into the right-half s -plane, or the loci of two (or more) poles *simultaneously* approaching the origin, we should realize that the closed-loop system is heading towards instability.

Constructing a root-locus plot by hand is difficult, and standard classical controls textbooks, such as D’Azzo and Houpis [2], contain information about doing so. However, by using either the intrinsic MATLAB command *roots* repeatedly for each value of the

design parameter, or the Control System Toolbox (CST) command *rlocus* the root-locus plot is very easily constructed. In the CST command *rlocus(sys)*, *sys* is the LTI object of the open-loop transfer function, $G(s)H(s)$. The *rlocus* program utilizes the MATLAB *roots* command repeatedly to plot the roots of $1 + KG(s)H(s)$ as a *design parameter*, K , is varied automatically. The design parameter, K , is thus a *scaling factor* for the controller transfer function, $H(s)$, and the root-locus shows what happens to the closed-loop poles as the controller transfer function is *scaled* to $KH(s)$. The user can also specify a vector *k* containing all the values of K for which the roots are to be computed by entering *rlocus(sys,k)* at the MATLAB prompt.

Example 2.19

For the closed-loop system of Figure 2.32 with plant, $G(s) = (2s^2 + 5s + 1)/(s^2 + 2s + 3)$, and controller, $H(s) = 1$, the root-locus is plotted in Figure 2.34 using the MATLAB command *rlocus* when the design parameter, K , is varied from -0.4 to 0.4 as follows:

```
>>num=[2 5 1]; den=[1 2 3]; k=-0.4:0.02:0.4; GH=tf(num,den);
    p= rlocus(GH,k); <enter>

>>plot(p(1,:), 'o'); hold on; plot(p(2,:), 'x') <enter>
```

It is clear from Figure 2.34 that both the poles of the closed-loop system have real part *zero* when $K = -0.4$, and then move towards the left as K is increased.

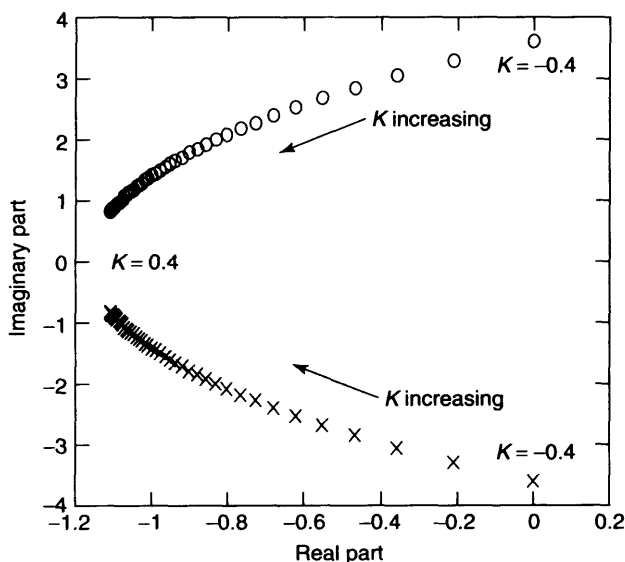


Figure 2.34 Root-locus plot of the closed-loop system of Example 2.19, as the design parameter, K , is varied from -0.4 to 0.4

For $K = -0.4$, the two loci are on the verge of crossing into the right half plane. Hence the closed-loop system is *stable* for $K > -0.4$ and *unstable* for $K < -0.4$. The value of K for which both the poles are on the imaginary axis (i.e. real part = 0) can be calculated by hand for this second order system by finding the characteristic equation as follows:

$$1 + KG(s)H(s) = 1 + K(2s^2 + 5s + 1)/(s^2 + 2s + 3) = 0 \quad (2.147)$$

or

$$(1 + 2K)s^2 + (2 + 5K)s + (3 + K) = 0 \quad (2.148)$$

Note that Eq. (2.148) is the second order characteristic equation, which can be expressed in terms of the natural frequency, ω_n , and damping-ratio, ζ , as Eq. (2.82). The poles with zero real part correspond to a damping-ratio, $\zeta = 0$. Comparing Eq. (2.131) with Eq. (2.82) and putting $\zeta = 0$, we get $(2 + 5K) = 0$, or $K = -0.4$, for which $\omega_n = [(3 + K)/(1 + 2K)]^{1/2} = 3.606$ rad/s, and the corresponding pole locations are $p_{1,2} = \pm i\omega_n = \pm 3.606i$, which are the same as the pole locations in Figure 2.34 for $K = -0.4$.

We can find the values of the design parameter K for *specific pole locations* along the root-loci using the CST command *rlocfind* as follows:

```
>>[k,poles]=rlocfind(sys,p) <enter>
```

where p is the specified vector containing the pole locations for which values of K (returned in the vector k) are desired. You can get the value of gain, K , the pole location, associated damping, natural frequency, and maximum overshoot of step response by clicking at any point on the root-locus generated by the command *rlocus(sys)*. Similarly, the command *rlocfind(sys)* lets you move a cross-hair along the root-loci with the mouse to a desired pole location, and then returns the corresponding value of K when you click the left mouse button. To do this you must first plot a root-locus, and then use *rlocfind* on the same plot as follows:

```
>>rlocus(sys); hold on; rlocfind(sys) <enter>
```

The CST command *rlocus* is of limited utility in the stability analysis of closed-loop system, because it merely scales the controller transfer function, $H(s)$, by a factor K . For the root-locus plot of more general systems, we should use the MATLAB command *roots*, as shown in Example 2.20.

Example 2.20

Consider a closed-loop system shown in Figure 2.35. Note that this system is different from the one shown in Figure 2.32, because the controller, $H(s)$, is in the *feedback path* whereas the controller was placed in the *forward path*, in series (or

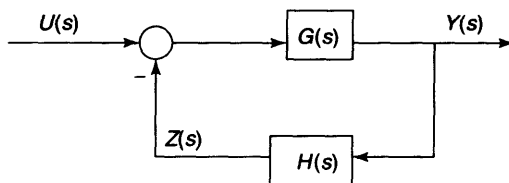


Figure 2.35 A closed-loop system with plant, $G(s)$, and feedback controller, $H(s)$

cascade) with the plant, $G(s)$, in Figure 2.32. You can show that both the systems have the *same* closed-loop transfer function, $Y(s)/Y_d(s)$, if $Z(s) = H(s)Y_d(s)$. The closed-loop transfer function, $Y(s)/U(s)$, for the system in Figure 2.35 is derived in the following steps:

$$Y(s) = G(s)[U(s) - Z(s)] \quad (2.149)$$

where

$$Z(s) = H(s)Y(s) \quad (2.150)$$

Substituting Eq. (2.150) into Eq. (2.149), we get the following expression for $Y(s)$:

$$Y(s) = G(s)U(s) - G(s)H(s)Y(s) \quad (2.151)$$

or

$$Y(s)/U(s) = G(s)/[1 + G(s)H(s)] \quad (2.152)$$

Note that the characteristic equation is still given by $1 + G(s)H(s) = 0$. For the present example, let us take $G(s) = (2s^2 + 5s + 1)/(s^2 - 2s + 3)$ and $H(s) = 1/(Ks + 1)$. This choice of $H(s)$ allows introducing a pole of $G(s)H(s)$ at a location that can be varied by changing the design parameter, K . (Introduction of poles and zeros in the open-loop transfer function, $G(s)H(s)$, through a suitable $H(s)$ to achieved desired closed-loop performance is called *compensation*, and will be studied in Section 2.12.) Let us plot the root-locus of the closed-loop system as the controller design parameter, K , is varied. Then the closed-loop characteristic equation is given by

$$Ks^3 + (3 - 2K)s^2 + (3 + 3K)s + 4 = 0 \quad (2.153)$$

Since $H(s)$ is not merely scaled by K (which was the case in Example 2.19), we cannot use the MATLAB CST command `rlocus` to get the root-locus. Instead, the intrinsic MATLAB command `roots` is used repeatedly to plot the root-locus as K is varied from 1.0 to 1.3 as follows:

```
>>i=1; for k=1:0.02:1.3; r=roots([k 3-2*k 3+3*k 4]); R(i,:)=r'; i=i+1;
end; plot(R,'x') <enter>
```