Simple technique for root locus plotting

M. Cywiak
*Centro de Investigaciones en Óptica, A.C.
Apartado Postal 37150 León, Guanajuato, México*

M. Castro
*Centro de Investigación Científica y de Educación Superior de Ensenada
Apartado Postal 2732, 22880 Ensenada, B.C., México*

Recibido el 20 de junio de 2001; aceptado el 20 de febrero de 2002

We present a simple technique for calculating and plotting the root locus of a linear control system. Although several programs are commercially available for this purpose, and remarks on this subject can be found in dedicated texts, a complete description of the program code is desirable to provide the designer with versatility and deeper insight. This report describes our technique and includes complete program codes. In this technique, the roots of the system's characteristic function are found by providing a single, unique guess value to be used in a first derivative search algorithm in order to calculate the unknown set of roots. Similarly to other techniques, the root locus is found by calculating the system's set of roots as the gain varies in the range of interest. The resolution obtained depends on the sampling rate of the gain parameter. The technique is suitable for any programming language and there is basically no limit on the root order of the system. To illustrate the technique a four-pole order system is analyzed. Then, to improve the response of the system a lead compensation network is added.. The usefulness of the technique is illustrated by comparing the time response of the compensated network over the uncompensated one.

Keywords: Linear control system; root locus; stability

Presentamos una técnica sencilla para calcular y trazar el lugar de las raíces de un sistema de control lineal. Aunque existen diversos programas comerciales para este propósito y pueden encontrarse algunos tópicos sobre este tema en textos dedicados, es deseable tener una descripción completa del código del programa, permitiéndole al diseñador mayor versatilidad y profundizar más en el sistema bajo su estudio. Este reporte describe nuestra técnica e incluye códigos completos de la programación. En esta técnica las raíces de la función característica se calculan proporcionando un único valor de prueba que es utilizado en un algoritmo de búsqueda de derivada de primer orden. De manera similar a otras técnicas, el lugar de las raíces es construido mediante el cálculo del conjunto de las raíces del sistema en función del parámetro de ganancia. La técnica puede ser utilizada en cualquier lenguaje de programación y básicamente no hay límite en el orden del polinomio característico. Para ilustrar la técnica analizamos un sistema lineal de cuarto orden al cual posteriormente se le añade una red de compensación de adelanto de fase para mejorar la respuesta en el tiempo. La utilidad de la técnica es ilustrada comparando la respuesta en tiempo del sistema compensado con respecto al sistema original.

Descriptores: Sistema de control lineal; lugar de las raíces; estabilidad.

PACS: 84.30.B; 02.30.Y

1. Introduction

The use of feedback in system controlling has become an important tool in several areas as for example in automation of industrial processes. Commonly applications can be found in controlling of communication networks [1]. Additionally, the controlling of hydraulic drives represents an important tool for metal forming machines, injection molding, and in the construction of testing devices [2]. Perhaps, the most commonly known application can be found in automatic aircraft piloting [3]. Higher control system technologies can be found in the manufacturing of semiconductor devices [4].

Independently of the area of the application, the main concern in system controlling is to calculate accurately the performance of the system under design for stability, time response and reliability.

Calculating the stability of a feedback system has deserved a great deal of attention. Several techniques have been proposed on this subject. Among them, the most used are the Nyquist stability criterion, the Nichols chart and the root-locus-plotting. Vast literature on these subjects can be found, for example in Refs. 5 and 6.

With the advent of the personal computer and the availability of faster processors, a suitable technique to be used in a personal computer is the root-locus-plotting. This technique gives accurate results and provides the designer with an ample knowledge of the behavior of the system under inspection. Then, several parameters of the system (gain, damping factor, time response, etc) can be calculated accurately. This allows for designing, not only stable systems, but also, by trial and error, it is possible to propose improvements over the original design. In contrast, the other mentioned techniques ordinarily give local, relative or moderate information [5,6].

There are several commercially available programs in control design and for root-locus-plotting, among them, RLOCUS function of MATLAB® is the most frequently used. However, the designer depends on inherent built in functions of the particular program or tool in use and no information is available of the internals of these calculations. Thus, an open technique is desired to provide more versatility to the designer. In what follows we show how the proposed technique works.

2. Feedback transfer function considerations

Let us consider the basic feedback block diagram shown in Fig. 1. This system can be used to derive the feedback relations common to all types of feedback theory [7]. In Fig. 1, $C\left(s\right)$ represents the Laplace transform of the time dependent input $c\left(t\right)$, and $R(s)$ the Laplace transform of the output $r\left(t\right)$. $G(s)$ and $H(s)$ are the forward and feedback transfer functions respectively. The parameter s is the complex variable in the Laplace's space.

Thus, as it is well known, the closed loop transfer function can be written as

$$\frac{C\left(s\right)}{R\left(s\right)}=\frac{G\left(s\right)}{1+G\left(s\right)H\left(s\right)}. \qquad (1)$$

Equation (1), represents the closed loop transfer function as the ratio of the Laplace-transformed integrodifferential equation describing the output function to the Laplace-transform of the integrodifferential equation describing the input function, with all initial conditions equal to zero [7]. Some examples are given in Sec. 3.

The equation that results of equating the denominator in Eq. (1) to zero is known as the characteristic equation. It is given by

$$1+G\left(s\right)H\left(s\right)=0. \qquad (2)$$

The characteristic equation, [Eq. (2)], is used to plot the system root-locus as described in the next sections.

3. Examples of transfer functions

Some typical examples found in electrical networks, are shown in Fig. 2. The corresponding transfer function is expressed for each network. As indicated above, the initial conditions are taken equal to zero. The terms phase-lead and phase-lag indicates that one is considering steady-state sinusoidal excitation and response functions. For example, Fig. 2A represents a phase-lag network because the phase angle associated with the phasor transfer function is negative when the substitution $s=i\omega$ is made, where $i=\sqrt{-1}$, $\omega=2\pi f$, and $f$is the frequency.

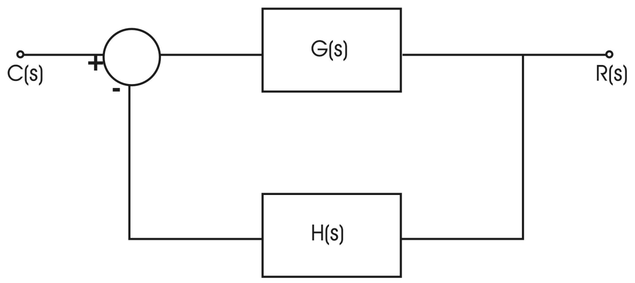


FIGURE 1. Basic feedback system. $C\left(s\right)$, is the Laplace transform of the time dependent input $c\left(t\right)$. $R(s)$ is the Laplace transform of the output $r\left(t\right)$. $G(s)$, and $H(s)$ are the forward and feedback transfer functions respectively. The parameter s is the Laplace's complex variable.

As a second example let us consider a network for controlling the speed of a d-c motor. The devices involved in this system are composed by an isolation amplifier, a tachometer a power amplifier an amplidyne and a d-c motor. A brief description for each mechanical device follows.

The tachometer is a fixed field d-c generator and used as a rotational speed sensor. The transfer function for the tachometer is given by

$$\frac{E_{out}\left(s\right)}{\omega\left(s\right)}=K_{T},$$

where K_{T} is the tachometer constant with dimensions of Volts/radian [7].

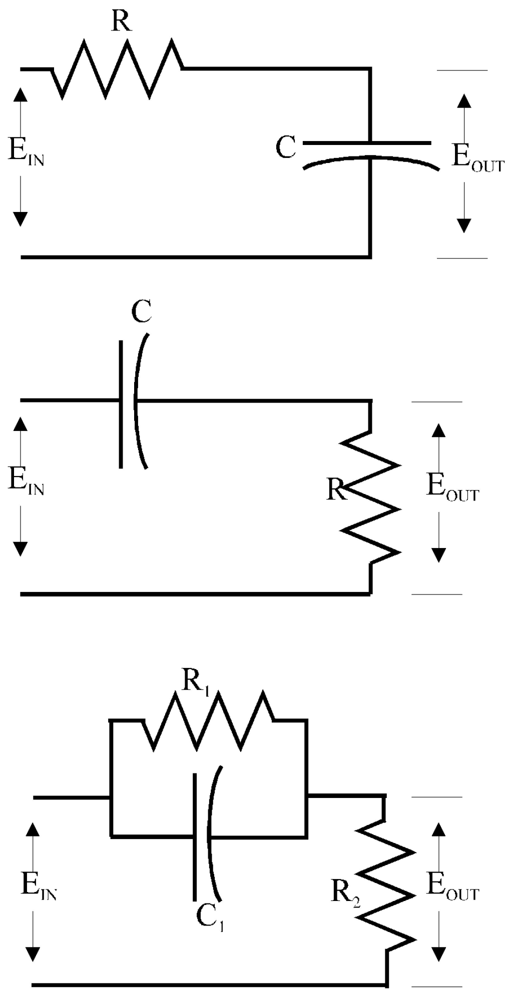


FIGURE 2. Basic electrical networks and transfer functions. A) Integrating or phase-lag network. B) Differentiating or phase-lead network. C) Phase-lead network with fixed DC attenuation.

The d-c motor uses electrical power to generate mechanical torque. The transfer function for the d-c motor is given by

$$\frac{\omega(s)}{E_{in}(s)} = \frac{K_M}{1+\tau_M s} = \frac{K_M/\tau_M}{(s+1/\tau_M)},$$

where $K_M = 1/K_b$, $\tau_M = JR_A/K_T K_b$. K_b is the back emf, or generator constant, measured in Volts per radian per second, J is the inertia of the motor shaft and the load measured in slug-feet^2, R_A is the resistance of the shunt field and K_T is the motor-torque constant measured in pound-feet per Ampere, [7].

The amplidyne is a very rugged electromechanical power amplifier capable of the controlled transformation of a few Watts into thousands of Watts. The amplidyne is essentially a d-c generator with an extra set of brushes, thus it consists of a control field winding and a quadrature field or QQ brushes used in much the same way as the elements of a d-c shunt generator. The transfer function for the amplidyne is given by

$$\frac{E_{out}(s)}{E_{in}(s)} = \frac{K_A}{(1+\tau_{Cc}s)(1+\tau_Q s)} = \frac{K_A/\tau_C\tau_Q}{(s+1/\tau_C)(s+1/\tau_Q)},$$

where K_A is the amplidyne constant and is dimensionless, and τ_C, and τ_Q are the control and quadrature field time constants [7].

The components described above can be combined to design a driven d-c motor controlled system as shown in Fig. 3. Using the transfer function for each component, it can be shown that for this system

$$G(s)H(s) = \left(\frac{s+1/R_1C_1}{s+\dfrac{R_1+R_2}{R_1R_2C_1}}\right)^2 \times \frac{K_1K_2(K_M/\tau_M)K_A\tau_C\tau_Q}{(s+1/\tau_C)(s+1/\tau_Q)(s+1/\tau_M)}.$$

The product $G(s)H(s)$ is known as the open loop transfer function.

In this application it is considered that the gain K_1 is to be determined for stable working conditions. All other parameters in the above equation are considered as constants, except of course, the state variable s.

Denoting the gain K_1 by the variable k, the above equation can be written as

$$G(s)H(s) = k\left(\frac{s+1/R_1C_1}{s+\dfrac{R_1+R_2}{R_1R_2C_1}}\right)^2 \times \frac{K_2(K_M/\tau_M)K_A\tau_C\tau_Q}{(s+1/\tau_C)(s+1/\tau_Q)(s+1/\tau_M)}.$$

This equation will be used in the next section to attain some general remarks on the analytical expression of the closed loop transfer function of a linear feedback system.

4. General remarks on linear system transfer functions

The results of the above example allows us to summarize some general remarks on the analytical expression of the overall transfer function. This approach is similar to the one found in dedicated texts, [6,7].

It can be seen that a feedback network consists basically of a forward and a feedback path.

Each path in the network is composed by individual blocks. Each block represents a set of electrical networks or/and servo-mechanical-electronic devices each having an associated rational transfer function which is written as the quotient of two polynomials with constant coefficients. Moreover, the rational function is explicitly expressed as products of well determined poles an zeroes which characterizes the physical behavior of the associated component or device at least in a functional linear region.

If N blocks, $B_1, B_2, \cdots, B_N$ are located in a particular path, the overall transfer function for the path is given by the product of the transfer function of each block as $(B_1)(B_2)\cdots(B_N)$.

A global multiplicand parameter, denoted as k, represents the unknown to be determined for stable working conditions. This parameter is located in a specific block and it is usually in the form of a gain.. When parameters other than k are unknown, they are varied (one at a time), in this case the corresponding root locus plot is defined as a generalized root loci.

From the above discussion, it is apparent that the open loop equation $G(s)H(s)$ can be expressed as

$$G(s)H(s) = k\frac{(s-z_1)(s-z_2)\ldots(s-z_m)}{(s-p_1)(s-p_2)\ldots(s-p_n)} \quad , \qquad (3)$$

where $m < n$, that is, the number of poles is greater than the number of zeros. Compare with the example equation at the end of Sec. 3.

The number of poles in the origin defines the type of system under consideration. If 0,1,2,3, and so on, poles are located at the origin a type 0,1,2,3, and so on, system is defined. The open-loop gain factor, k, is usually considered positive. Equation (3) is the general analytical form that can be studied by the root locus technique and this is the only case that is considered in this work.

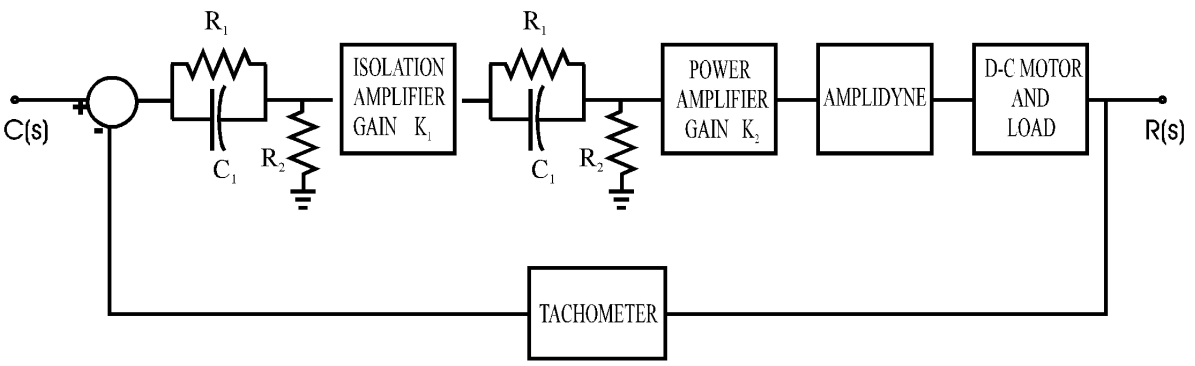


FIGURE 3. Block diagram for d-c motor controlling.

Fortunately the vast majority of linear servo-systems can be modeled by equation (3). When the above conditions are not fulfilled it is necessary to use partial fraction expansion techniques to intend to describe the system as in Eq. (3).

By using Eq. (3), the characteristic equation, can be written as

$$1+k\frac{(s-z_1)\,(s-z_2)\ldots(s-z_m)}{(s-p_1)\,(s-p_2)\ldots(s-p_n)}=0. \tag{4}$$

From Eq. (4), the root-locus of the system is given by the set of roots of the equation

$$(s-p_1)\,(s-p_2)\ldots(s-p_n) \\ +k\,(s-z_1)\,(s-z_2)\ldots(s-z_m)=0, \tag{5}$$

where the value of the gain is sampled over the range of interest.

As $m < n$, Eq. (5) results to be a polynomial of order n. Let $R1, R2, ..., R_n$ denote the n roots of Eq. (5), then, the root-locus of the system is the plot of the real part against the imaginary part for each of the roots as the gain varies form an initial to a final value.

Without lose of generality we will consider the most common case in which the gain k, varies for positive values, although this is not a restriction for the present technique.

5. Root locus construction, an example

As mentioned above, the technique can be used on any programming system or language.

The example discussed in Sec. 3 could be used to describe our root locus technique. Nevertheless, we consider advisable to study a more complex system to enhance the description. For this purpose, let us consider a unitary feedback control system. Typically, unitary feedback systems are suggested as the starting point in system design [5].

The example selected for this description consists of a four-pole, type-1 (one pole at the origin) system. Additionally, for completeness, two arbitrary zeroes are also included. The poles are located at 0, -1, -4 and –6 and the zeroes at $-2-i20$and at $-2+i20$. Although the example is arbitrarily selected, several servo-systems can easily be studied with this model by only changing the numerical values assigned to the poles and zeroes

The programming platform is Mathcad® Plus 6.0 and was selected due to its powerful graphical environment.

The characteristic equation for this system can then be written as

$$(s-p_1)\,(s-p_2)\,(s-p_3)\,(s-p_4) \\ +k\,(s-z_1)\,(s-z_2)=0 \tag{6}$$

where

$$p_1=0;\quad p_2=-1;\quad p_3=-4;\quad p_4=-6; \\ z_1=-2-i20;\quad z_2=-2+i20. \tag{7}$$

It can be seen that Eq. (6) is a polynomial of fourth degree on the variable s and that the restriction $m < n$, is fulfilled.

Explicitly, Eq. (6) can be written as

$$s^4+11s^3+(34+k)s^2+(24+4k)s+404k=0.$$

We may express this polynomial as

$$a_4(k)s^4+a_3(k)s^3+a_2(k)s^2+a_1(k)s+a_0(k)=0,$$

where

$$a_4(k)=1,\quad a_3(k)=11,\quad a_2(k)=34+k, \\ a_1(k)=24+4k\quad \text{and}\quad a_0(k)=404k.$$

In order to illustrate the technique we divide the description in the following steps:

Step a) In the first step of the technique the characteristic function is constructed. In Mathcad® the next statements are used for this particular example:

$$a4(k) := 1$$
$$a3(k) := 11$$
$$a2(k) := 34 + k$$
$$a1(k) := 24 + 4k$$
$$a0(k) := 404k$$
$$f1(s,k) := a4(k)\cdot s^4 + a3(k)\cdot s^3 + a2(k)\cdot s^2 + a1(k)\cdot s + a0(k)$$

The above directives are statements only for Mathcad®, the notation must be written accordingly for the programming language selected. Obviously for each value of k a different polynomial must be considered. For example, if $k = 0$, the polynomial to be considered is

$$s^4 + 11s^3 + 34s^2 + 24s + 0 = 0.$$

The dependence on k in the above statements will allow to find a set of roots for each particular value of k as described in the next step.

The above statements are valid only for this example and must be rewritten for the particular system under study. Notice however, that it is not necessary to calculate by hand each coefficient of the polynomial. For this task, Mathcad®provides the built in function "Polynomial Coefficients".

In the above statements the characteristic function (denoted as function one) is introduced only for programming manipulation.

Step b) In the second step the number of points for displaying the root-locus are chosen. Obviously, the higher the number of samples the better the resolution of the plot but the larger the processing time. As it will be apparent in the simulations shown below, 800 samples give reasonable good resolution for the selected example. In Mathcad®, the next statements accomplish for this task

$$N = 800;\ n = 0..N; \quad k_n = \frac{n}{10400}.$$

The above statements allow varying the gain parameter from 0 to 0.077 (800/10400) in 800 equally spaced steps. Each value is stored in a linear array with 800 entries. This variation is suitable for the example as it can be seen in Fig. 2A. The range of interest needs to be adjusted for each particular case by trial and error by plotting the root-locus several times.

Step c) The third step is to provide an initial guess value for initiating the process. A common way to do this is by considering the three terms of higher degree in the polynomial as a second-degree polynomial. In the example, the polynomial

$$a_4(k)s^4 + a_3(k)s^3 + a_2(k)s^2 + a_1(k)s + a_0(k) = 0 \quad (8)$$

is considered as

$$a_4(k)s^2 + a_3(k)s + a_2(k) = 0, \quad (9)$$

for a particular, arbitrary value of k. One of the roots of Eq. (9) is used as the guess value.

Although the guess value provided is not critical, it is possible that some values, especially when the guess value results real, may cause the calculations to fall into an infinite loop. This error is easy to detect by introducing a break point in the loop (described in step d) after a reasonable number of iterations. Mathcad®gives an out of bounds message when this error occurs. Experience shows that this error can easily be fixed by adding a pure imaginary number between one and two to the guess value. Note that a unique guess value is proposed for the overall process.

Step d) The next step is to write the loop where the roots will actually be calculated. The next statements in Mathcad® will calculate the first root of the characteristic equation

```
$root1(s,k) :=$
    $m \leftarrow 0$
    $s_m \leftarrow s$
    while 1
        $y \leftarrow s_m$
        $s_{m+1} \leftarrow y - \frac{f1(y,k)}{\frac{d}{dy}f1(y,k)}$
        break if $|f1(s_m,k)| < 10^{-8}$
        $m \leftarrow m + 1$
        break if $m > 200$
    $s_{m+1}$
```

In the above code one root of the characteristic equation (denoted first function) is calculated. The loop begins by setting the first entry of an array (entry zero) to the guess value and it is provided at the calling of the loop (next statement bellow). The loop approaches the actual root according to a first derivative search. Every value estimated in the loop is introduced into the function. When the absolute value of function-one computed in the proposed value results less than some predetermined error, in this case 10^{-8}, the loop finishes and returns the estimated root within this very small error. If 200 iterations are not sufficient to find a root within this error, a break statement finishes the loop and an "out of bounds" message is obtained. Note that Mathcad®provides a built in derivative function.

In this case, the loop described above is executed 800 times by the next statement,

$$R1_n := root1(s1, k_n),$$

where $s1$ is the guess value and k_n is the n-th sampled gain.

After complete execution, (no "out of bounds" message generated), a set of 800 roots within the specified precision is obtained and stored in an array. Each root in the set corresponds to one of the elements in the gain array constructed in step b). As the characteristic equation for this example is a polynomial of degree four, this set of roots is denoted as the first-set of roots. Three more sets remain to be found.

In order to proceed further, the second-set of roots will be calculated. For this, a second function is defined by using the next statement

$$f2(s, k, R1) := \frac{f1(s, k)}{s - R1}.$$

Notice that the second function has a degree less than the first function.

Notice that the first function has dependence only on s and on the gain while the second function shows dependence also on the first-set of roots previously found. This will allow to obtain the second-set of roots for each value stored in the first-set of roots.

The loop statements for calculating the second-set of roots is then

```
$root2(s,k,R) :=$
    $m \leftarrow 0$
    $s_m \leftarrow s$
    while 1
        $y \leftarrow s_m$
        $s_{m+1} \leftarrow y - \frac{f2(y,k,R)}{\frac{d}{dy} f2(y,k,R)}$
        break if $|f2(s_m,k,R)| \le 10^{-8}$
        $m \leftarrow m+1$
    $s_{m+1}$
```

As in the first loop the guess value is provided and stored in the first entry of an array. Additionally, the corresponding gain and value stored in the gain-array and in the first-set of roots are provided for each iteration.

The statement for executing 800 times the above code to calculate the second-set of roots is

$$R2_n := root2(s1, k_n, R1_n).$$

At the end of the loop, the second-set of roots within the proposed precision is stored in the array. As described, each calculated root corresponds to a gain stored in the gain array and at the same time to a root stored in the first-set array.

The technique follows the same sequence up to the fourth set of roots. For completeness, the statements for calculating the third and fourth set of roots follow:

$$f3(s, k, R1, R2) := \frac{f2(s, k, R1)}{s - R2},$$

```
$root3(s,k,R1,R2) :=$
    $m \leftarrow 0$
    $s_m \leftarrow s$
    while 1
        $y \leftarrow s_m$
        $s_{m+1} \leftarrow y - \frac{f3(y,k,R1,R2)}{\frac{d}{dy} f3(y,k,R1,R2)}$
        break if $|f3(s_m,k,R1,R2)| \le 10^{-8}$
        $m \leftarrow m+1$
    $s_{m+1}$
```

$$R3_n := root3(s1, k_n, R1_n, R2_n).$$

The code for finding the fourth set is

$$f4(s, k, R1, R2, R3) := \frac{f3(s, k, R1, R2)}{s - R3},$$

```
$root4(s,k,R1,R2,R3) :=$
    $m \leftarrow 0$
    $s_m \leftarrow s$
    while 1
        $y \leftarrow s_m$
        $s_{m+1} \leftarrow y - \frac{f4(y,k,R1,R2,R3)}{\frac{d}{dy} f4(y,k,R1,R2,R3)}$
        break if $|f4(s_m,k,R1,R2,R3)| \le 10^{-7}$
        $m \leftarrow m+1$
    $s_{m+1}$
```

$$R4_n := root4(s1, k_n, R1_n, R2_n, R3_n).$$

6. Root-locus plotting for the example

Figure 4A shows the root-locus obtained for the four-pole example of the above section. The dashed line shows the desired damping ratio. In this case we have chosen the commonly used value 0.5. The intersection of the dashed line with the root-locus corresponds to one of the roots. By exploiting the graphical capabilities of Mathcad®, which allows expanding portions of the plot, it is possible to estimate graphically the corresponding root an then to find it precisely by choosing the corresponding entry in the calculated arrays. The intersection of the root-locus with the damping ratio line results $-0.363 + i0.612$.

Once a root is found, the gain and the remaining roots are obtained by consulting in the arrays the corresponding values. For this case, $k = 0.033$. The corresponding four roots are

$$R1 = -4.57; \quad R2 = -5.704;$$

$$R3 = -0.363 + i0.612;$$

$$R4 = -0.363 - i0.612.$$

For reliability, each calculated root is introduced into the characteristic equation and its absolute value calculated for each corresponding gain value. The maximum error found results in less than 10^{-10}, representing an excellent precision for most practical applications.

Once the complete set of roots and gain are known the system performance can be tested. For example, for a unitary-step input, the overall transfer function can readily be expanded in partial fractions and the time response can be plotted by calculating the inverse Laplace transform. Figure 4B shows the time response for the four-pole system discussed above. It can be seen, from Fig. 4B, that the maximum overshoot occurs at approximately five seconds and the system settles after approximately 14 seconds.

Let us illustrate how the present technique can be used to improve the performance of the system. For this purpose, let us consider that it is desired to speed up the time response of the system just discussed. Several techniques to accomplish this are possible. Among them, let us choose, a simple lead compensation network introduced in the forward loop of the system. Let, the lead network consist of a pole located at –16.0 and the zero at –1.6 (trial an error is necessary to find suitable values). Additional amplification is included to decrease the error signal, for impedance matching, and to meet with the gain value that will be obtained after performing the calculations, Fig. 5. The characteristic equation for the compensated system can be written as

$$(s - p_1)(s - p_2)(s - p_3)(s - p_4)(s - p_5) + k(s - z_1)(s - z_2)(s - z_3) = 0, \quad (10)$$

where $p_1, p_2, p_3, p_4, z_1, z_2$, corresponds to the original system, given in Eq. (7), and p_5, z_3, correspond to the additional pole and zero introduced by the compensating network (-16.0 and -1.6, respectively).

It will be noticed that Eq. (10) corresponds to a fifth degree polynomial. The coefficients of the polynomial can readily be found by using the "polynomial coefficient", built in tool provided by Mathcad®. The fifth degree polynomial for this characteristic function is given by

$$s^5 + 27s^4 + (210 + k)s^3 + (568 + 5.6k)s^2 + (384 + 410.4k)s + 646.4k = 0,$$

and the corresponding program statement is given by

$$f1(s,k) := a5(k)\cdot s^5 + a4(k)\cdot s^4 + a3(k)\cdot s^3 + a2(k)\cdot s^2 + a1(k)\cdot s + a0(k).$$

In order to plot the root-locus for the compensated system, the code for calculating the fifth set of roots must be added to the overall previous code. For completeness we provide the code for obtaining the fifth set of roots:

$$f5(s,k,R1,R2,R3,R4) := \frac{f4(s,k,R1,R2,R3)}{s - R4}$$

```
$root5(s,k,R1,R2,R3,R4) :=$
    $m \leftarrow 0$
    $s_m \leftarrow s$
    while 1
        $y \leftarrow s_m$
        $s_{m+1} \leftarrow y - \frac{f5(y,k,R1,R2,R3,R4)}{\frac{d}{dy} f5(y,k,R1,R2,R3,R4)}$
        break if $|f5(s_m,k,R1,R2,R3,R4)| \le 10^{-7}$
        $m \leftarrow m + 1$
        break if $m > 200$
    $s_{m+1}$
```

$$R5_n := root5(s1, k_n, R1_n, R2_n, R3_n, R4_n).$$

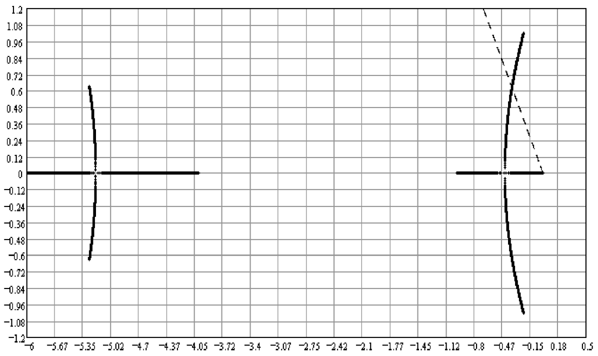


FIGURE 4A. Root-locus plot for the example four-order pole system. The poles are located at 0, -1., -4 and –6 and the zeroes at $-2 - i20$and at $-2 + i20$.

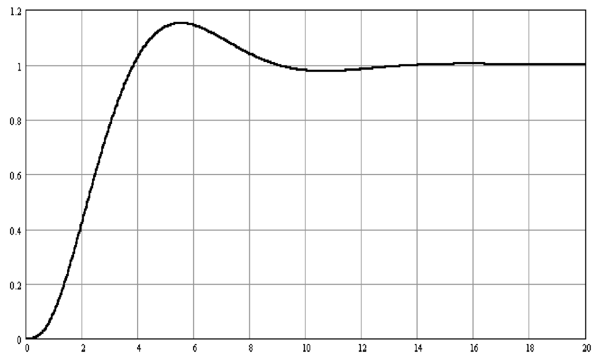


FIGURE 4B. System time response to the unitary step input.

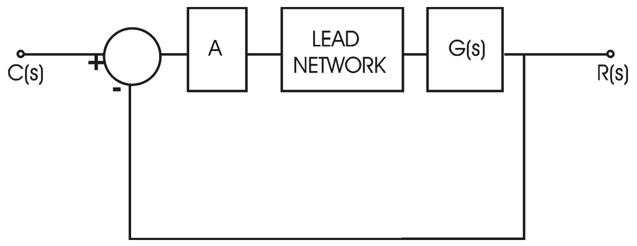


FIGURE 5. Compensated system for the example. It includes an amplification module, A,a lead network for compensation and unitary Feedback. The other parameters are as in Fig. 1.

Figure 6A shows the root locus for the compensated system and Fig. 6B shows the corresponding time response to the unitary-step input.

For a 0.5 damping- factor we obtain the following roots,

$$R1 = -2.39; \quad R2 = -0.70 + i1.18;$$

$$R3 = -0.70 - i1.18;$$

$$R4 = -7.46; \quad R5 = -15.74.$$

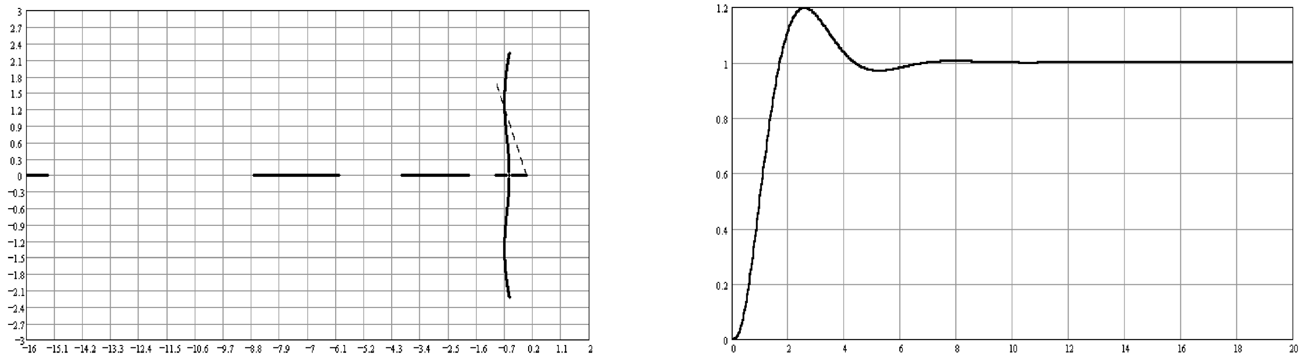
The corresponding gain value, 0.825 results in a gain increase of approximately 25 times as compared with the gain of the original system (0.033). This represents an additional improvement as increasing the gain decreases the error signal.

As it can be seen in Fig. 6B, the system responds faster. The maximum overshoot now occurs at approximately 2.5 seconds and the system settles at approximately 8.0 seconds, 6.0 seconds faster than the original. Then, an improvement in time response and gain is obtained over the original system.

Finally, we point out that the simulations were performed in a personal computer with a 700 Mhz. processor. The maximum processing time was about 1.5 minutes. This running time is comparable with the time of execution of commercially available programs and gives the designer completely free access to the internal code which is not possible with built in programs.

7. Conclusions

A technique for calculating and plotting the root-locus of a linear control system was presented. As in other techniques, the characteristic equation of the system is analyzed as a polynomial, which is a function of the gain parameter of the system. The root locus is found by varying and sampling the gain parameter in the range of interest. For each value of the gain, a complete set of roots is obtained by providing a unique guess value and performing a first derivative search for calculating the actual values. For reliability, each root so computed is substituted in the characteristic equation and an error value is recorded. The maximum error results in less than 10^{-10} and can easily be lowered if desired.



(A) (B)

FIGURE 6. A) Root-locus plot for the compensated system. The poles are located at 0, -1, -4, -6, -16 and the zeroes at $-2 - i20$, $-2 + i20$ and -1.6. B) System time response to the unitary step input.

To illustrate the simplicity and usefulness of the technique, a particular four-pole order system was selected. The root-locus of the system was plotted and the characteristic roots and gain were found for a desired damping ratio. The time response of the system to a unitary-step was obtained. A lead compensation network was proposed for improving the time response and by trial and error suitable parameters were found. The root-locus of the compensated system was compared to the uncompensated one to show the improvement in the time response thus, showing the usefulness of having an open code root locus visualization technique.

The technique can be implemented in any programming language and shows to be reliable and time efficient, offering a versatile tool for the interested designer.

Acknowledgements

We thank to Manuel Servín for useful discussion.

1. L. G. Bushnell, *IEEE Control Systems* **21** (2001) 22.
2. E. Detiek and E. Kiker, *Experimental techniques* **25** (2001) 35.
3. E. O. Doebelin, *Measurement Systems Application and Design*, (McGraw-Hill Book Co, 1966), p. 345.
4. J. Y. Choi and H. M. Do, *IEEE Trans. On Semiconductor Manufacturing* **14** (2001) 1.
5. P. H. Lewis and C. Yang, *Basic Control Systems Engineering*, (Prentice-Hall Inc, 1997), p. 248.
6. W. J. Palm III, *Control Systems Engineering*, (John Wiley & Sons Inc. 1986), p. 484.
7. R. E. Lueg, *Basic Electronics*, (International Textbook Company 1963), p. 284.