Root Locus Algorithms for Programmable Pocket Calculators

E. R. Wechsler
Communications Systems Research Section

Two algorithms are described which allow the plotting of individual points on a root locus diagram with or without time delay. The development was performed during the design of a continuous phase shifter used in the Baseband Antenna Combiner for the Deep Space Network (DSN). The algorithms, which are expected to be useful in similar DSN efforts, are simple enough to be implemented on a programmable pocket calculator. The coordinates of the open-loop zeros and poles, the gain constant K, and the time delay T are the data inputs.

I. Introduction

The root locus method (Ref. 1) allows the designer to obtain the poles of a closed-loop linear system when the zeros and poles of the open loop are known. The equation describing the root locus is

$$K \frac{\prod_{i=1}^{z}(s - s_i)}{\prod_{i=z+1}^{z+p}(s - s_i)} e^{-Ts} = -1 \qquad (1)$$

where

$s = \sigma + j\omega$

$s_i = \sigma_i + j\omega_i$ { are zeros for $1 \leqslant i \leqslant z$; are poles for $z < i \leqslant z + p$

$T \geqslant 0$ is the time delay

K is a real parameter (usually a gain constant)

There are basically three methods for plotting root loci by digital computation:

(1) The s plane is scanned until points of the locus are found with the desired accuracy (Ref. 2). This method requires lots of computation and of memory.

(2) The plotting starts from a pole and advances along the locus in fixed increments. If any parameter is modified, the search has to start again because the locus changes its shape (Refs. 3, 4).

(3) The plotting is performed by solving the polynomial:

$$Q(s) + KP(s) = 0$$

This approach was implemented on a pocket calculator (Ref. 5). It does not apply to systems with time delay and it requires a preliminary expansion of the polynomial by hand or with an additional program.

The root locus algorithms described here were developed during the design of a continuous phase shifter used in the Baseband Antenna Combiner of the DSN. The algorithms

should be helpful to engineers involved in the design of linear feedback systems.

II. Calculations

Using the complex Ln function we get from (1):

$$\mathrm{Ln}\left[K\frac{\prod_{i=1}^{z}(s-s_i)}{\prod_{i=z+1}^{z+p}(s-s_i)}e^{-Ts}\right]-\mathrm{Ln}\,(-1)\;=\;0 \qquad (2)$$

which can be expanded to:

$$\ln|K|+\sum_{i=1}^{z+p}\delta_i\ln|s-s_i|-\sigma T+j\left[\sum_{i=1}^{z+p}\delta_i\arg(s-s_i)-\omega T-n\pi\right]$$

$$=0 \qquad (3)$$

$$n\;=\;\begin{cases}\pm1,\pm3,\pm5,\cdots\text{ for } K>0\\ 0,\pm2,\pm4,\cdots\text{ for } K<0\end{cases}$$

$$\delta_i\;=\;\begin{cases}1\text{ for zeros } 1\leqslant i\leqslant z\\ -1\text{ for poles } z<i\leqslant z+p\end{cases}$$

Two real functions are defined, based on the real and the imaginary parts of (3):

$$R(\sigma,\omega)=\ln|K|+\sum_{i=1}^{z+p}\delta_i\ln\left[(\sigma-\sigma_i)^2+(\omega-\omega_i)^2\right]^{1/2}-\sigma T \qquad (4a)$$

$$Q(\sigma,\omega)\;=\;\sum_{i=1}^{z+p}\delta_i\arg\left[(\sigma-\sigma_i)+j(\omega-\omega_i)\right]-\omega T-n\pi \qquad (4b)$$

n such that $-\pi<Q\leqslant\pi$

A point on the root locus will satisfy the equations:

$$R(\sigma,\omega)\;=\;0\quad\text{and}\quad Q(\sigma,\omega)\;=\;0 \qquad (5)$$

Two different methods were used for solving Eqs. (5):

The Newton-Raphson iteration (Ref. 6) starts from a point of coordinates σ and ω and produces the correction terms $\Delta\sigma$ and $\Delta\omega$ from the following equations:

$$R(\sigma,\omega)+R_\sigma\,\Delta\sigma+R_\omega\,\Delta\omega\;=\;0 \qquad (6a)$$

$$Q(\sigma,\omega)+Q_\sigma\,\Delta\sigma+Q_\omega\,\Delta\omega\;=\;0 \qquad (6b)$$

Since $R(\sigma,\omega)$ and $Q(\sigma,\omega)$ are the real and imaginary parts of the analytic function (2) they satisfy the Cauchy-Riemann conditions

$$Q_\sigma\;=\;-R_\omega \qquad (7a)$$

$$Q_\omega\;=\;R_\sigma \qquad (7b)$$

Substituting (7) in (6) and solving the system we get:

$$\Delta\sigma\;=\;\frac{-R\cdot R_\sigma+Q\cdot R_\omega}{R_\sigma^2+R_\omega^2} \qquad (8a)$$

$$\Delta\omega\;=\;\frac{-R\cdot R_\omega-Q\cdot R_\sigma}{R_\sigma^2+R_\omega^2} \qquad (8b)$$

R_σ and R_ω are derived from (4a):

$$R_\sigma\;=\;\sum_{i=1}^{z+p}\delta_i\frac{\sigma-\sigma_i}{(\sigma-\sigma_i)^2+(\omega-\omega_i)^2}-T \qquad (9a)$$

$$R_\omega\;=\;\sum_{i=1}^{z+p}\delta_i\frac{\omega-\omega_i}{(\sigma-\sigma_i)^2+(\omega-\omega_i)^2} \qquad (9b)$$

The coordinates of the new iteration are:

$$\sigma'\;=\;\sigma+\Delta\sigma\quad\text{and}\quad\omega'\;=\;\omega+\Delta\omega \qquad (10)$$

The iteration process continues until:

$$\Delta\sigma^2+\Delta\omega^2\leqslant P^2$$

where P is the desired precision.

This algorithm converges very fast but it presents difficulties when the starting point is far from the solution because it can diverge and give errors or jump to a solution on a branch other than the one of interest.

A gradient method (Ref. 7) was developed which does not have the disadvantages of the previous algorithm but is slower to converge. Consider the gradients of $R(\sigma, \omega)$ and $Q(\sigma, \omega)$:

$$\text{grad } R = \mathbf{u}R_\sigma + \mathbf{v}R_\omega \tag{11a}$$

$$\text{grad } Q = \mathbf{u}Q_\sigma + \mathbf{v}Q_\omega \tag{11b}$$

u and **v** are unit vectors along the respective σ and ω axis.

Substituting (7) in (11b):

$$\text{grad } Q = -\mathbf{u}R_\omega + \mathbf{v}R_\sigma \tag{12}$$

We notice that:

$$|\text{grad } R| = |\text{grad } Q| \tag{13a}$$

$$\arg(\text{grad } Q) - \arg(\text{grad } R) = \frac{\pi}{2} \tag{13b}$$

Because the two gradients are orthogonal, moving along grad R will produce a maximum rate of change for R and a zero rate of change for Q and vice versa. In order to make R decrease in absolute value at the fastest rate we have to move in the direction pointed by the vector:

$$\mathbf{a} = -\text{Sgn}(R) \cdot \text{grad } R \qquad \text{Sgn}(R) = 1 \text{ for } R \geqslant 0 \qquad \text{Sgn}(R) = -1 \text{ for } R < 0$$

Likewise for Q we have to move along:

$$\mathbf{b} = -\text{Sgn}(Q) \cdot \text{grad } Q$$

Therefore moving along the search vector $\mathbf{c} = \mathbf{a} + \mathbf{b}$ will reduce both R and Q in absolute value.

$$\mathbf{c} = -\text{Sgn}(R) \cdot \text{grad } R - \text{Sgn}(Q) \cdot \text{grad } Q \tag{14}$$

Depending on the polarities of R and Q there are four possible values for the angle θ between **c** and grad R (Fig. 1).

$$\theta = \arg(\mathbf{c}) - \arg(\text{grad } R) \tag{15}$$

$$\left.\begin{array}{ll} R \geqslant 0 \;;\; Q \geqslant 0 & \theta = -135^\circ \\ R \geqslant 0 \;;\; Q < 0 & \theta = 135^\circ \\ R < 0 \;;\; Q \geqslant 0 & \theta = -45^\circ \\ R < 0 \;;\; Q < 0 & \theta = 45^\circ \end{array}\right\} \tag{16}$$

We start from a point of coordinates σ and ω and we calculate R, Q, R_σ, R_ω from (4a), (4b), (9a), (9b) respectively.

We find arg (grad R) from R_σ and R_ω using the rectangular to polar conversion. Knowing the polarities of R and Q we find θ from table (16). We get the direction of search vector **c** from (15):

$$\arg(\mathbf{c}) = \arg(\text{grad } R) + \theta$$

During one pass, the magnitude of the step is a constant D. The recommended value for D during the first pass is a few percent of full scale.

The search vector has the modulus D and the argument arg (**c**). Using the polar to rectangular conversion we get the coordinate corrections $\Delta\sigma$ and $\Delta\omega$. The search moves now to a new point of coordinates:

$$\sigma' = \sigma + \Delta\sigma \qquad \omega' = \omega + \Delta\omega$$

The search pass will consist initially of steps moving at a constant angle θ relative to the local grad R. As soon as the search crosses one of the curves $R = 0$ or $Q = 0$, it starts a zigzag pattern along the crossed curve because of the change in polarity of the corresponding function (Fig. 2). When the search reaches the vicinity of the solution ($R = 0$; $Q = 0$) the zigzag pattern is interrupted and a 180-deg change of direction occurs.

There are only two possible cases as shown in Fig. 2. The history of the last three steps is enough for detecting this 180-deg change in direction between the last step and any of the previous two steps.

This part of the algorithm is implemented through the use of a three-register stack into which the angles θ are fed. After each step we look for:

$$|\theta_N - \theta_{N-1}| = 180^\circ \quad \text{or} \quad |\theta_N - \theta_{N-2}| = 180^\circ$$

If the answer is true the step size D is reduced and the search process continues with a new pass. The search ends when the magnitude of the increment D becomes smaller than the required precision.

III. Conclusion

Both algorithms were implemented on a Casio FX-602P programmable pocket calculator. The first program uses 36 memory registers and 281 program steps, while the second one uses 41 memory registers and 351 program steps. They allow the design of systems with up to eight singularities ($z + p = 8$).

References

1. D'Azzo, J. J., and Houpis, C. H., *Linear Control System Analysis and Design*, McGraw-Hill, pp. 202-241, 1975.
2. Krall, A. M., and Fornaro, R., "An Algorithm for Generating Root Locus Diagrams," *Communications of the ACM*, Vol. 10, No. 3, pp. 186-188.
3. Williamson, S. E., "Accurate Root Locus Plotting Including the Effects of Pure Time Delay," *Proc. IEE (GB)*, Vol. 116, No. 7, pp. 1269-71.
4. Ash, R. H., and Ash, G. R., "Numerical Computation of Root Loci Using the Newton-Raphson Technique," *IEEE Trans.*, AC-13, pp. 576-58, 1968.
5. Harden, R. C., and Simons, F. O., "Root Locus Algorithms and Routines Adapted to Hand-Held HP-67 Computers," 12th Annual Southeastern Symposium on System Theory, Publ, *IEEE*, pp. 285-9, 1980.
6. Scarborough, J. B., *Numerical Mathematical Analysis*, 2nd ed., Oxford University Press, pp. 203-204, 1955.
7. Korn, G. A., and Korn, T. M., *Electronic Analog and Hybrid Computers*, 2nd ed., McGraw-Hill, pp. 328-335, 1972.

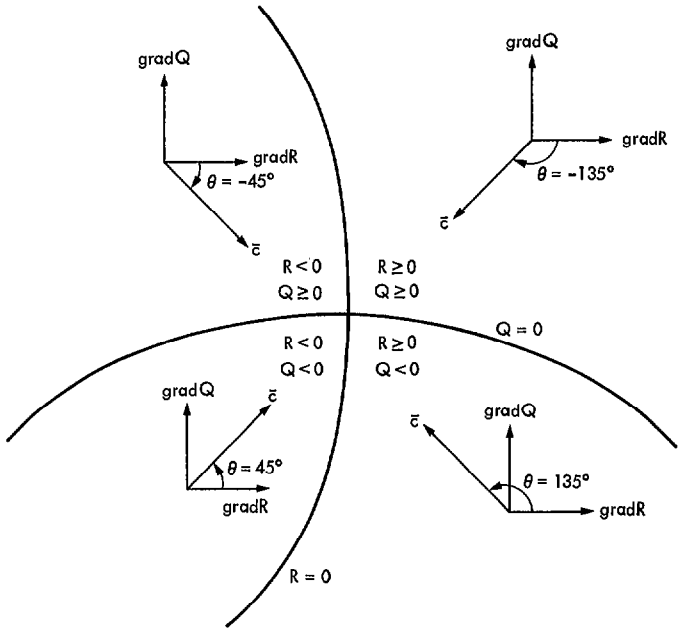

Fig. 1. Selection of search vector for the gradient method

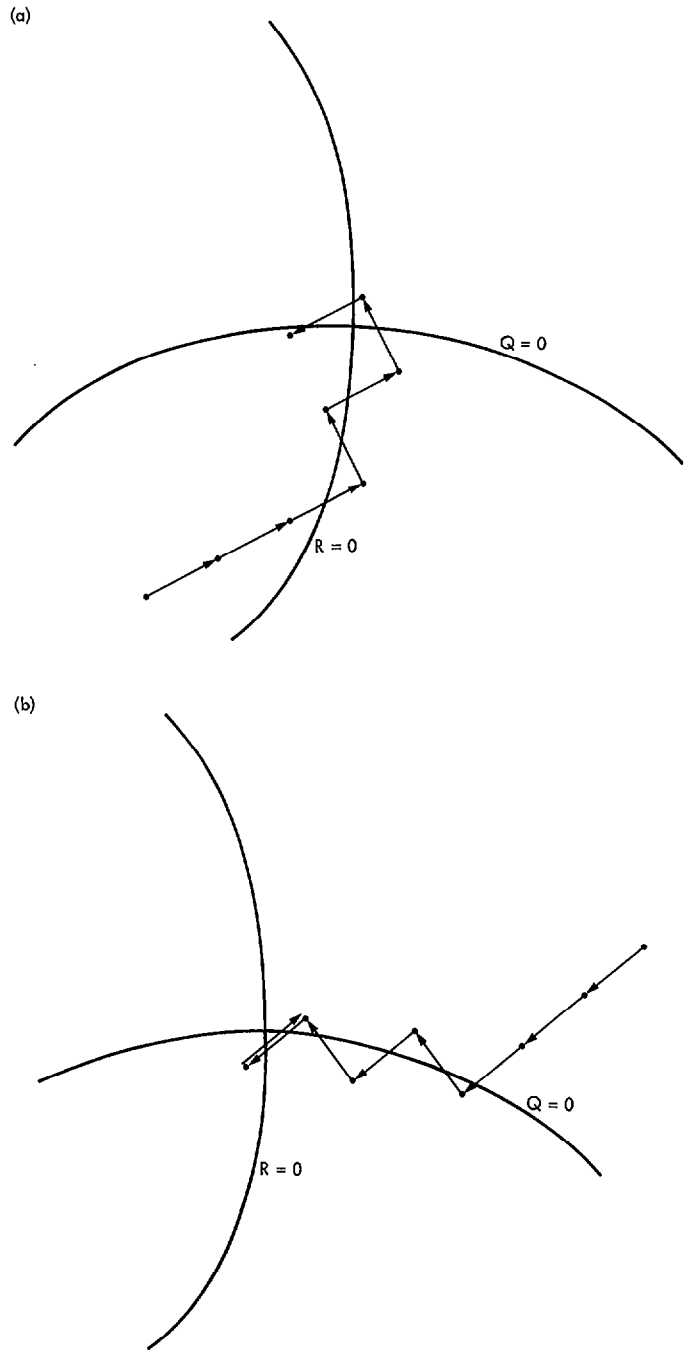

Fig. 2. Gradient method search pattern – 180-deg change of direction within the last three steps