Control system design and analysis package

G.J. Lastman,[a] N.K. Sinha[b]
[a]*Department of Applied Mathematics, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1*
[b]*Department of Electrical and Computer Engineering, McMaster University, Hamilton, Ontario, Canada L8S 4L7*

Abstract

A control system design and analysis package, developed recently by the authors, is described. The entire package fits on a 3.5-inch high density floppy disk, and runs on any IBM compatible personal computer equipped with a numeric co-processor. The package allows the analysis and design of single-input single-output systems using transfer functions or state equations. It is suitable for use by students in an undergraduate engineering program as well as for practising engineers, and runs well under DOS, Windows, and OS/2 Warp. It is very user-friendly and provides information and on-line help in each of the programs.

1 Introduction

Teaching control theory, with application to real-life problems usually requires a great deal of computation. Therefore, it is desirable that students have access to reliable computer programs which are easy to use and remove the drudgery out of the numerical computations, so that the students can work out realistic problems that provide the basic understanding of the underlying principles and provide numerical insight. At present several program packages are available commercially, but they tend to be expensive as well as demanding powerful computer hardware. In addition, often they require that a student learn a particular programming language, and are not very user-friendly. The present programming package is an attempt to remove these difficulties. This package covers the complete set of computations required in a typical undergraduate course in control systems, while requiring no more space than one high density floppy disk (less than 1.44 megabytes). In addition, on-line help

is available, and the user can obtain necessary information about each program directly from the package. The hardware requirements are minimal, as the program runs quite fast even on a 80386-based computer. However, it is desirable that the computer be equipped with a numeric co-processor, in view of the large number of complex numerical computations required for most of the programs. In fact, these programs can serve most of the needs of a practising control engineer. Since most of the programs also provide graphical output, a VGA colour monitor is required.

The program package contains a suite of programs which can also be used independently, if desired, but it is preferable to run them through the systems program which acts as the selector program. It will be described in the next section.

2 The Systems Program

To start this program in the DOS mode, the user need only type the word SYSTEMS and press the <Enter> key. In the windows mode one may either install it as a program item and double click on the corresponding icon, or run it through the file menu. In the OS/2 Warp platform, it will be run as a DOS program by clicking on the icon corresponding to the program. The following menu appears on the screen, with the cursor highlighting the first program.

Systems
Frequency response of continuous-time linear systems
Frequency response of discrete-time linear systems
Phase plane plot of second-order ordinary differential equations
Root locus of linear systems
Time-domain response of continuous-time linear systems
Nichols Chart method
Design of lead, lag, or lag-lead compensators
Pole-placement compensator design
Solution of the algebraic Riccati equation
Stability of linear systems by the Routh criterion
Stability of linear systems with parameter uncertainty
State variable feedback for single-input single-output systems
System matrices of discrete-time and continuous-time systems
Information about this program
Exit the systems program
Menu bar: <↑>, <↓> Accept: < Enter >

Figure 1: The main menu

By pressing the down arrow key on the keyboard, the highlight bar may be moved to the desired item. Pressing the <Enter> key will cause that

program to be run. Note that the menu allows the user to obtain information about the systems program, and also to exit the program. For example, if the highlight bar is moved to "Information about this program", and the <Enter> key is pressed, the following is displayed on the screen.

The Systems Program

This program acts as a selector program for a suite of programs for continuous-time and discrete-time systems that may arise in control systems. The programs in the suite are

1. Frequency response of continuous-time linear systems (FRPC.Exe)
2. Frequency response of discrete-time linear systems (FRPD.Exe)
3. Phase plane plot of second-order ordinary differential equations (Phase.Exe)
4. Root locus plot of linear systems (Locus.Exe)
5. Time-domain response of continuous-time linear systems (Response.Exe)
6. Nichols Chart method (Nichols.Exe)
7. Design of lead, lag, or lag-lead compensators (Compens.Exe)
8. Pole-placement compensator design (Pole.Exe)
9. Solution of the algebraic Riccati equation (Riccati.Exe)
10. Stability of linear systems by the Routh criterion (Routh.Exe)
11. Stability of linear systems with parameter uncertainty (Khariton.Exe)
12. State variable feedback for single-input, single-output systems (StateFb.Exe)
13. System matrices for discrete-time and continuous-time systems (SysMat.Exe)

Each program contains on-line information about the program as well as help explanations where appropriate. A VGA color monitor is required for all programs; a color VGA graphics card is required for the graphical output of programs 1 to 6. Because of the intensive floating point computations the programs should be run on a PC with a numeric coprocessor.

Although the 13 programs in this suite of control systems programs have been thoroughly tested by the authors, no warranty, express or implied, is made by the authors as the accuracy and functioning of the programs, nor shall the fact of distribution of these programs constitute such warranty, and no responsibility is assumed by the authors in connection therewith.

In particular, these 13 programs should not be relied on for solving a problem whose incorrect solution could result in injury to a person or loss of property. If you do use the programs in such a way, it is at your own risk.

Figure 2: Screen showing information about the systems program

It is seen from figure 2 that the package consists of 13 different programs, useful in the analysis and design of control systems, described in reference [1]. These will be described in the following sections.

3 Description of the programs

The first program (FRPC.Exe) displays plots of the frequency response of a continuous-time linear system from its transfer function, which may contain an ideal delay. An early version of this program was presented at a conference in 1990 [1]. From the main screen of the program (obtained by pressing the <Enter> key while the corresponding choice is highlighted in the main menu of the systems program), the user can alter the numerator and the denominator of the transfer function from the default value, alter the desired frequency range, alter the value of the delay, and the co-ordinates. Three types of plots are available. These are (i) Bode plots, (ii) Gain-phase plot (the gain in decibels is plotted against the phase shift, with the frequency as a parameter along the plot), and (iii) polar plots. All of these plots are used extensively in the analysis and design of control systems. In each case, the user can cause a cursor to move along the plot, showing the values of the frequency, the gain and the phase shift at the location of the cursor. Note that in the case of Bode plots, both the gain (in decibels) and the phase shift are plotted against the frequency in logarithmic coordinates. To improve the readability of the plots, the two curves are displayed in different colours, matching the labels on the corresponding coordinate axes. On-line help is available from the main menu of this program.

The second program (FRPD.Exe) displays different types of plots of the frequency response of a discrete-time linear system from its z-transfer function, in a manner similar to the first program.

The third program (Phase.Exe) obtains the plots of the trajectories in the phase plane of the solution of second-order ordinary differential equations. Although this was originally developed as a graphical method for solving second-order nonlinear differential equations, it is still useful since the trajectories reveal a great deal of information about the nature of the differential equation. In this program we actually use a Runge-Kutta-Fehlberg (RKF45) algorithm to numerically solve the differential equation [3,4] and then plot it in the phase plane. Thus we combine the power of a good numerical algorithm with the graphical display of the trajectory. Again, one can use a cursor on the graph to reveal the coordinates of any point on the trajectory, including the total time elapsed from the initial point.

The fourth program (Locus.Exe) shows the locus of the roots of the equation

$$KP + Q = 0$$

where P and Q are polynomials in the complex variable s or z, as the parameter K varies from zero to infinity. In Control Theory this plot is used to show how the poles of a closed-loop system move in the s-plane (or the z-

plane) as the parameter K varies from zero to infinity. An early version of this program was presented in the Electrosoft Conference [5] held in 1993. Here, again, we utilize an efficient algorithm for calculating the roots of the resulting polynomial for different values of K, and then plot the loci. The user can choose either the s-plane of the z-plane. For the latter, the unit circle is shown as it is the boundary of stability of the resulting closed-loop systems. As in other programs, a user can move a cursor along the plot, displaying the values of the axes, K and the damping ratio at that location of the cursor. The program uses some empirical constants which can be changed by the user. Several problems may arise due to the discrete nature of changes in K as we proceed, and a great deal of effort has gone into ensuring that the plots are continuous and accurate. This has required setting up certain empirical constants which may have to be changed by the user in some special cases. The information screen for this program provides further details.

The fifth program (Response.Exe) determines and plots the response of a single-input single-output continuous-time linear system to some "standard" inputs. An earlier version of this program was described in reference [6]. The system may be described either in the form of its transfer function or by its state equations. In the latter case, the program uses Leverrier's algorithm to determine the transfer function. The standard inputs considered in the program are (i) a unit impulse, (ii) a unit step, (iii) a unit ramp, (iv) cos bt (where the constant b is to be specified by the user), (v) sin bt, (vi) exp(at) cos bt (where the constants a and b are to be specified by the user), and (vii) exp(at) sin bt. Since the ramp function approaches infinity for large t, this is replaced by the following function

$$r(t) = \begin{cases} \dfrac{t}{a}, & \text{for } t \le a \\ 1, & \text{for } t > a \end{cases} \tag{1}$$

The program calculates the transient response of the system to the desired input by determining the inverse Laplace transform of the product of the transfer function and the Laplace transform of the input. The analytical expression for the output is displayed, and the user is given the choice to store it in a file. The program then plots this function against time. Initially the time is taken from zero to five times the largest time constant of the system, but the user can specify the time interval and scale the plot, if desired. If the program finds that the system is unstable, the user is notified that no plot was attempted since the system is unstable.

The sixth program uses a modification of Nichols chart to solve the problem of designing a compensator so that the maximum value of the frequency response of the closed-loop system, denoted by M_m, does not exceed a specified value, usually given in decibels, occurring at a specified frequency ω_m. Originally, control theorists solved this problem by using Nichols charts where the main axes are the open-loop gain in decibels and the open-loop phase shift in degrees. Superimposed on this are the curves for constant gain

of the closed-loop system (M-curves) and constant phase shift in the closed-loop system (N-curves). The user had to determine graphically how much gain and phase-shift should be added at ω_m to meet the desired specifications and then design a suitable compensator to achieve this. Details are given in reference [1]. The current version of the program plots the frequency response of the uncompensated system, and the specified M-curve using the log-magnitude/phase coordinates. Note that plotting only one M-curve (others are not needed for this application) keeps the graph uncluttered. It then determines how the frequency response curve should be moved so that at the specified value of ω_m, it is tangent to the M-curve. This gives the values of the gain and the phase shift that the compensator must provide at the frequency ω_m. The transfer function of the compensator is then calculated, and if desired, plots of the frequency response of the uncompensated and the compensated system are shown along with the specified M-curve. The program uses some empirical constants, which may have to be changed in some cases. An information screen provides all the necessary details.

The seventh program (Compens.Exe) determines the transfer function of a lead, lag, or lag-lead compensator that will provide a specified gain and phase shift at some given frequency. The procedure for lag or lead compensators requires the solution of a quadratic equation, and a compensator exists if and only if this quadratic has real roots, one of which must be positive. The details of the algorithm are given in reference [1], and can also be obtained from the information screen of this program. For lag-lead compensator, roots of two quadratic equations have to be evaluated. It may be noted that the algorithm given in reference [1] gives only an approximate solution for lag-lead compensators, but the program uses a modified iterative method to obtain the exact solution. This design technique is useful for designing compensators for control systems when the specifications for the system are given either in terms of the phase margin or M_m. In fact, the transfer function of the compensator in Nichols.Exe is calculated using this program after first determining the gain and phase shift that it must add at ω_m.

The eighth program (Pole.Exe) can be used for designing compensators so that the closed-loop system will have all its poles at specified locations. The scheme for compensation is shown in figure 3 [1]. The two compensators, $G_u(s)$ and $G_y(s)$ have a common denominator, that is, they have the same poles. The system to be compensated can be specified either by its transfer function, $G_p(s)$, or by its state equations. The user must specify the desired locations of the poles of the compensated system, as well as the location of the poles of the compensator. The latter may be either of the same order as $G_p(s)$, or of the minimal order (one less than that of the system). It is known that a solution to this problem exists only if the system is both completely controllable and completely observable, or alternatively, the numerator and the denominator of the transfer function are relatively prime, that is, do not have a common factor. The program also allows the gain constant K to be adjusted so that the system has d.c. gain equal to one,

corresponding to zero steady-state error to step inputs.

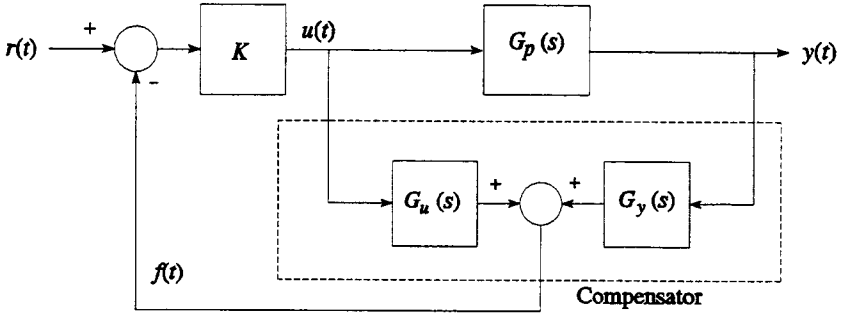

Figure 3: Pole-placement compensation

The ninth program (Riccati.Exe) can be used for solving the algebraic Riccati equation that occurs in the infinite horizon quadratic regulator problem. The algorithm described in reference [7] is used for this purpose. The solution is obtained in an iterative manner, as described in the reference, as well as the information screen of the program. This program can be useful in introducing an undergraduate class to optimal control theory.

The tenth program (Routh.Exe) uses the Routh criterion of stability to determine the stability of a continuous-time linear system. This is done by obtaining the Routh table from the denominator polynomial of the transfer function. Only the first column of the Routh table is displayed, since stability depends only on the number of sign changes in this column. The program also allows a user to shift the imaginary axis in the s-plane in order to determine how close the system is to becoming unstable. The program can also be applied to determine the stability of discrete-time systems described by their z-transfer functions. Since the Routh criterion determines the number of roots of a polynomial with positive real parts, we must introduce a transformation which maps the unit circle of the z-plane into the left half of the w-plane. This bilinear transformation is given by

$$w = \frac{z+1}{z-1} \tag{2}$$

which maps the region inside the unit circle of the z-plane into the left half of the w-plane. The Routh table with the resulting characteristic polynomial is now calculated and its first column is displayed.

The eleventh program (Khariton.Exe) can be used for determining the stability of linear systems in the more practical situation when there is some uncertainty in the values of one or more parameters. Consider the polynomial

$$D(x) = c_n x^n + c_{n-1} x^{n-1} + \cdots + c_1 x + c_0 \tag{3}$$

where the upper and lower bounds on the coefficients are given by

$$a_i \le c_i \le b_i \tag{4}$$

The independent variable, x, can be either s (for a continuous-time system) or z (for a discrete-time system), and the values of a_i, and b_i are given. Normally, to determine the stability of the system, one would have to apply the Routh criterion (after using the bilinear transformation for the discrete-time system) to a large number of polynomials formed by all possible different polynomials within the intervals of the coefficients.

The recent seminal work of the Russian mathematician Kharitonov makes it possible to determine stability for systems with coefficients in the specified ranges through the use of only four "corner" polynomials (see reference [8] for a good description). The Routh criterion is applied to each of these four polynomials. The system is stable if and only if there is no sign change in the first column of the Routh table for each polynomial.

It may be pointed out that after transformation to the w-plane, and multiplication by $(w\text{-}1)^n$, we get the polynomial

$$P(w) = \sum_{i=0}^{n} c_i (w+1)^i (w-1)^{n-i} \tag{5}$$

We note that $w=1$ is the image of the point at infinity, which is not a root of $D(z)$ if $a_n b_n > 0$. The information screen of this program describes how this difficulty is overcome, so that the tests on the four polynomials are sufficient but not necessary for stability with parameter uncertainty.

The twelfth program (StateFb.Exe) can be used to calculate the state feedback vector that will place the eigenvalues of the closed-loop system at specified locations. Given the state equations, the program first determines if the system is completely controllable. If this condition is satisfied, the matrix for transforming the system to the controller canonical from is calculated [1]. The state feedback vector for this canonical form is easily obtained and the transformation used to determine the state feedback vector for the original form of the state equations.

The thirteenth program (SysMat.Exe) can be used for determining the state equations of the corresponding discrete-time system from those of a continuous-time system, and vice-versa. It is assumed that the continuous-time system is sampled at a constant rate and the sampler is followed by a zero-order hold. The sampling interval must be known. The transformation from the continuous-time to the discrete-time requires finding the exponential of a square matrix A, called the system matrix. This can be done by adding an infinite matrix series, which is uniformly convergent. The series can be truncated after about 15 terms if the sampling interval T is selected so that the spectral radius of AT is less than 0.5. The algorithm is described in most text books (for example see [1]). The inverse transformation from discrete-time to continuous-time requires finding the natural logarithm of a square matrix. This also is in the form of an infinite series which can be truncated after a certain number of terms to obtain desired accuracy. The algorithm used in this program is described in reference [9].

4 Conclusions

We have described above the programs in the package developed by us. Each program has on-line help available, and all have built-in example problems for illustration. The programs in this package cover the entire spectrum of computations needed in an undergraduate course on control systems. Although we have not included explicitly in this package a program for determining the transfer function of a single-input single-output system from its state equations, this can be done simply by using the program for transient response (Response.Exe). Similarly, although we do not have a program for designing asymptotic state observers, it is possible to use the program StateFb.Exe, due to the duality of the two programs. This is accomplished by entering the transpose of the A-matrix of the system instead of A, and the vector C instead of B, and then proceeding as in the case of state feedback. Similarly, one can determine the inverse Laplace transform of a rational function through the program Response.Exe. We hope that these features will make this program package a pleasure to use. As these programs are intended for students, we shall welcome feedback from them.

The programs described in this paper all work well, and have been fully tested by the authors. We shall be glad to share these programs with others working in the field and welcome their feedback about the user-friendliness and other suggestions. Participants at the Electrosoft conference are invited to copy these programs.

At present, there is one short-coming in these programs. The printed copies of the graphical outputs of the various programs can be obtained only by using some commercially available screen-capturing programs (for example, Pizzaz). We would like very much to make the programs independent of this so that a user can obtain a hard copy without using these commercial programs. We welcome the suggestions from the participants at the Electrosoft '96 conference, and others on ways to overcome this shortcoming.

References

1. N.K. Sinha, *Control Systems*, second edition, Wiley Eastern, New Delhi, 1994.

2. N.K. Sinha, K. Hood and G.J. Lastman, Analysis and design of control systems using personal computers, *Proceedings of the 33rd Midwest Symposium on Circuits and Systems*, Calgary, Alberta, August 1990, pp. 500-503.

3. G.J. Lastman and N.K. Sinha, *Microcomputer-based Numerical Methods for Science and Engineering*, Holt, Rinehart and Winston, New York, 1989.

4. G.E. Forsyth, M.A. Malcolm and C.B. Moler, *Computer Methods for Mathematical Computations*, Prentice-Hall, Englewood Cliffs, New Jersey, 1977.

5. G.J. Lastman and N.K. Sinha, Computer program for root locus plots, *Proceedings Electrosoft '93*, Southampton, U.K., July 1993, pp. 127-136.

6. N.K. Sinha and G.J. Lastman, System Theory Applications of Personal Computers - III, *I.E.E.E. Circuits and Devices Magazine*, vol. 5, November 1989, pp. 34-36.

7. P.H. Petkov, N.D. Christov and M.M. Constantinov, *Computational Methods for Linear Control Systems*, Prentice-Hall, Englewood Cliffs, New Jersey, 1991.

8. R.J. Minnichelli, J.J. Anagnost and C.A. Desoer, An elementary proof of Kharitonov's stability theorem, *I.E.E.E. Transactions on Automatic Control*, vol. AC-34, 1989, pp. 995-998.

9. G.J. Lastman and N.K. Sinha, Infinite series for logarithm of a matrix, applied to identification of linear continuous-time multivariable systems from discrete-time models, *Electronics Letters*, vol. 27, 1991, pp. 1468-1469.