ROBUST COMPUTER-AIDED CONTROL SYSTEM DESIGN FOR NONLINEAR PLANTS

James H. Taylor
General Electric Company
Corporate Research and Development
Schenectady, New York 12345

INTRODUCTION

The last decade has seen the development of several powerful methods and software packages that are very effective in facilitating the design of control systems for multivariable *linear* plants. Among the more noteworthy of these, in terms of power and easy usage, are the Cambridge Linear Analysis and Design Package (CLADP) (Edmunds, 1979), based on MacFarlane's multivariable root locus technique (MacFarlane, Kouvaritakis and Edmunds, 1977); the University of Manchester multivariable control system design suite using the inverse Nyquist technique of Rosenbrock (1969); the Massachusetts Institute of Technology package for linear quadratic regulator and Kalman-Bucy filter design (Posbergh and Chen, 1980); and the Honeywell HONEY-X system (Stein and Pratt, 1981).

Compared with the advanced state of the art for control systems design for linear plants, parallel developments for *nonlinear* plants are still in their infancy. In terms of industrial practice, the basic approach for nonlinear systems design is often unchanged from that of a generation ago: linearize the plant to obtain a small-signal perturbation model, use that model to affect a preliminary design, and iterate and patch it up until acceptable performance is obtained. This ad hoc procedure is often very wasteful in engineering effort, and results in a final product that is very sensitive to assumed operating condition, to nonlinear effects, and to the experience and "feel" of the designer—the very antithesis of a robust design. The following computer-aided control system design (CACSD) environment and quasilinearization approach are proposed as a way to ameliorate these problems.

In developing CACSD software, it must be recognized that the practicing engineer rarely encounters control systems development projects in which the system model is *readily* or *realistically* formulated in either of the two standard linear forms: an n-variable state-space model or an n^{th} order transfer function (scalar or matrix). More typically, the *first* and *best* models available for the system are nonlinear, either in the form of nonlinear state equations or block diagrams with intermingled linear parts and nonlinear operators.

In either case, it is often essential that the nonlinear model be investigated and manipulated quite extensively before the system is understood well enough for control systems to be designed with any assurance of success. In particular, meaningful answers to the following questions are usually of great interest:

1. Is the model *realistic*?
2. How does the system *equilibrium* (steady-state values of the system variables) vary as the inputs take on different constant values?
3. How does the system *small-signal linear model* (especially its eigenvalues or modes) vary as the inputs are varied?
4. What is the system *dynamic response* to various input forms (e.g., step inputs, sinusoids)?

These explorations are performed to achieve the following objectives:

1. Determining if the system can be made to behave properly without changes in hardware or configuration,
2. Determining one or more good operating points (constant values of input and state that result in well-behaved linearized models), or determining pathological situations,
3. Understanding which nonlinear effects are important, and
4. Determining linearized models (conventional linear or quasilinear [describing function]; state space or transfer function) that can serve as the basis for control systems design.

It is evident from the above outline that there is often a great deal of analysis and study required before an engineer is prepared to use the standard linear systems CACSD packages that are now available. This heavy *preliminary burden* associated with the effective use of linear CACSD can be a major deterrent to its utilization. Another important consideration in CACSD is the *iterative nature* of the control system design activity. One often returns to the above procedures (modeling, finding equilibria, linearization, and simulation) many times. One view of the total design process, including linear CACSD as one very important element, is provided in Figure 1. This "closed loop" concept further emphasizes the great need to minimize the engineering time (and especially

the manual labor) that must be spent in these activities.

CACSD FUNCTIONS

Initial Model Formats

The usual model formulations that are developed for systems are of two types: state space and block diagram. A state space model is often obtained directly from the application of basic principles (e.g., classical mechanics, circuit laws, etc.) to a unitary system (e.g., an aircraft). The block diagram model format usually arises in describing systems that are not unitary, i.e., that are made up of components, each of which is modeled individually and appropriately interconnected with the other system parts.

The state space model is easy to represent mathematically as

$$\dot{\underline{x}} = \underline{f}(\underline{x},\underline{u}) \ , \ \underline{y} = \underline{g}(\underline{x},\underline{u}) \tag{1}$$

where x is the state vector (made up of all variables needed to describe the system dynamically), u is a vector of inputs, y is a vector of outputs, and $\underline{f}(\cdot)$ and $\underline{g}(\cdot)$ represent general algebraic relationships among the variables. This model format is the easiest to manipulate and study, so we will consider it to be primary.

The block diagram model can dealt with readily by using a *preprocessor* to create the corresponding state space model. One procedure is as follows: Identify the input(s) and output(s) of each block as $\underline{u}_i$ and $\underline{y}_i$, respectively. For each linear dynamic block, represented by $G_i(s)$ of order m_i, one allocates m_i state variables, and a subsystem state vector differential equation is set up using a convenient canonical form (e.g., phase variables). For each nondynamic block, the input/output equation is expressed in algebraic form, e.g., $\underline{y}_i = \underline{f}_i(\underline{u}_i)$. The final state variable model (equation 1) is obtained by the preprocessor by concatenating the dynamic subsystem states into one state vector, and using the subsystem input/output relations (linear and nonlinear) to couple the subsystems dynamics. Since this preprocessor concept is quite straightforward, it will not be discussed further. Henceforth, we assume that a state space model (equation 1) is available for the purpose of technical discussion.

The formulation and input of the nonlinear system state space model should be made as simple as possible. Many engineers have worked primarily in Fortran, so the need for ready accessibility would seem to dictate using that language. On the other hand, it is widely recognized that Fortran is quite cumbersome as a modeling language, so the introduction or use of a substantially better modeling language might gain ready acceptance. For greatest utility, it may be best to permit models to be written either in Fortran or in a flexible simulation language, as is the case with the Lund Institute of Technology nonlinear system simulation package SIMNON (Elmqvist, 1975). In fact, the SIMNON language (which is based on Algol 60) has many excellent features that make it a strong contender for use as the general CACSD model formulator. The only major drawback to the SIMNON language is its inability to perform matrix mathematical operations; this deficiency is easy to correct. The ease with which a model can be formulated and interfaced with the CACSD program is a key factor, so the selection of modeling languages should be given a great deal of consideration.

Equilibrium Determination

Mathematically, the procedure for determining the operating point(s) of a nonlinear system is to solve

$$\underline{f}(\underline{x}_o,\underline{u}_o) = \underline{0} \tag{2}$$

to obtain $\underline{x}_o$ corresponding to the value(s) of $\underline{u}_o$ of interest. Assuming that the equations are well posed, especially, that solutions exist, then available routines may be used—routine ZSYSTM from the International Mathematical and Statistical Library (IMSL) is a possible choice that the author has used successfully. Another more powerful set of routines for solving this problem may be found in MINPAC (More, Garbow and Hillstrom, 1980).

Small-Signal Linearization (SSL)

Conventional linearization, also called Taylor series linearization, can be expressed in terms of an operating point $(\underline{x}_o,\underline{u}_o)$ as

$$\begin{aligned} \delta\dot{\underline{x}} &= \left[\frac{\partial \underline{f}}{\partial \underline{x}}\right]\delta\underline{x} + \left[\frac{\partial \underline{f}}{\partial \underline{u}}\right]\delta\underline{u} \\ &= A_o\,\delta\underline{x} + B_o\,\delta\underline{u} \end{aligned} \tag{3}$$

$$\begin{aligned} \delta\underline{y} &= \left[\frac{\partial \underline{g}}{\partial \underline{x}}\right]\delta\underline{x} + \left[\frac{\partial \underline{g}}{\partial \underline{u}}\right]\delta\underline{u} \\ &= C_o\,\delta\underline{x} + D_o\,\delta\underline{u} \end{aligned}$$

where $\delta\underline{x}, \delta\underline{u}, \delta\underline{y}$ represent perturbations or "small signals" around $\underline{x}_o$, $\underline{u}_o$, and $\underline{y}_o = \underline{g}(\underline{x}_o, \underline{u}_o)$, respectively. The arrays $[\partial \underline{f}/\partial \underline{x}]$ etc., are evaluated as the partial derivatives at $\underline{x}_o$, $\underline{u}_o$, and are matrices; the subscript o stresses the dependence of the arrays upon the operating point.

The matrices A_o, B_o, C_o, D_o could be evaluated *algebraically*, either using an existing high-level symbolic manipulation program, or by hand. It is probably easier, in most cases, to take the partials *numerically*, e.g., in the scalar case,

$$\frac{\partial f}{\partial x} \cong \frac{f(x+\delta) - f(x-\delta)}{2\delta} \tag{4}$$

This process would be faster than explicit computer differentiation by symbolic manipulation (unless the matrix set $\{A_o, B_o, C_o, D_o\}$ is to be evaluated for many operating conditions, in which case having computer-written code would be an advantage), not as error-prone as differentiating and coding the matrices by hand, and would allow model updates to be made in only one place—the routine for evaluating $f(\cdot)$ and $g(\cdot)$—which is highly desirable. Numerical differentiation is also the method of choice if some model nonlinearities are only available in tabular form (e.g., aircraft "aerodata"). For poorly conditioned tabular data, it might even be necessary to consider spline function curve fitting and differentiation.

From these considerations, it would seem that the best investment of first effort in this area would be to find or develop a *robust* numerical differentiator. It should be recognized, however, that using finite difference methods for obtaining the linearized model introduces numerical problems that require careful attention. A useful discussion of this topic may be found in Dahlqvist and Bjorck (1974). Basically, one must be concerned with *round-off errors* if δ in equation 4 is too small, and *truncation errors* (due to the curvature of f) if δ is too large. The author is now using the following extension of Richardson extrapolation (Dahlqvist and Bjorck, 1974):

$$\text{(i)} \quad \frac{\partial f^{(1)}}{\partial x} = \frac{f(x+\delta) - f(x-\delta)}{2\delta} \tag{5}$$

$$\text{(ii)} \quad \frac{\partial f^{(2)}}{\partial x} = \frac{f(x+2\delta) - f(x-2\delta)}{4\delta} \tag{6}$$

(iii) δ is appropriate if $\partial f/\partial x^{(1)} = \partial f/\partial x^{(2)}$ with q significant digit accuracy; if accuracy is greater, increase δ, if accuracy is less, decrease δ.

$$\text{(iv)} \quad \frac{\partial f}{\partial x} = \frac{4}{3}\frac{\partial f^{(1)}}{\partial x} - \frac{1}{3}\frac{\partial f^{(2)}}{\partial x} \tag{7}$$

Equation 7 results in minimum truncation error for the δ selected.

Quasilinearization

There are two fundamental quasilinearization methods (cf., Atherton, 1975; Gelb and Vander Velde, 1968): the sinusoidal-input describing function (SIDF) approach and the random-input describing function (RIDF) approach. In the former, signals in the nonlinear system are assumed to be comprised of constant (dc) values plus sinusoids, while in the latter case, signals are of the form bias plus gaussian random variable. Quasilinear gains that depend on the signal amplitudes are calculated for each nonlinearity, and they collectively define the quasilinear system model that can be used for a wide variety of analysis and synthesis tasks. The major advantage of a DF model over an SSL model is that the former characterizes the system with nominal signal amplitudes (as specified by the analyst), instead of for infinitesimal variations. Another benefit of quasilinearization is that meaningful DF gains exist for nonlinear elements such as the ideal relay and hysteretic nonlinear effects for which SSL gains are completely undefined. The efficacy of the DF approach in capturing nonlinear effects in an efficient and accurate manner is well documented; cf., the above-named texts.

Both of these DF techniques have recently been extended so that general nonlinear system models can be quasilinearized (Taylor, 1980; Taylor, Siegel, Price, and Gelb 1980). The use of both methods in the context of designing controllers for nonlinear plants is discussed in Taylor (1982); a detailed treatment is not repeated here. An important point made in the latter reference is that the SIDF approach generally appears to be the better technique for control system design.

The major requirement for the purposes of CACSD is the set of SIDF gain matrices $\{A_{df}, B_{df}, C_{df}, D_{df}\}$. It is well known that this matrix set provides a more realistic characterization of the behavior of the nonlinear plant compared with SSL, as mentioned above; however, evaluating the describing functions is not as simple. Two approaches are discussed in Taylor (1982): using a library of describing functions (coded directly from Atherton (1975), for example), and using a direct numerical method that combines simulation of the plant with sinusoidal inputs of specified amplitudes and fast fourier transform techniques to obtain the gains associated with the first harmonic (fundamental) component of the output of each nonlinearity. The latter approach would seem to be the most suitable for use in a CACSD environment such as that proposed here, for the same reasons that numerical differentiation is best suited for SSL.

Eigenvalues

Linearized models (SSL or DF) can reveal a great deal of information about the behavior of the system in the vicinity of a given operating point. The most important characterization of the dynamic performance of a system is provided by the eigenvalues, which allow the dynamic response to be categorized for perturbations as being unstable, underdamped or overdamped. Many linear system performance measures can be directly related to eigenvalues—in particular, bandwidth, percent overshoot, and risetime. With appropriate interpretation, the eigenvalues of a DF model can also provide some idea of how a nonlinear system will behave. Another valuable use of the DF system eigenvalues is to obtain an indication of how important the system nonlinearities are for the assumed signal amplitude levels. Clearly, if the eigenvalues of the SSL and DF models are very nearly equal, then the dynamic behavior of the system is not being influenced strongly by the system nonlinearities for the signal amplitudes under consideration. There are many standard routines available for obtaining eigenvalues, some of which are robust and well proven, so we need not consider the numerical aspects of this procedure.

Simulation

There are many aspects of nonlinear system behavior that can only be revealed by simulation. For example, no amount of SSL analysis will give an engineer a meaningful idea of how a fighter aircraft will perform in a violent maneuver. While there are mathematical techniques for studying nonlinear effects (e.g., absolute stability criteria and describing function methods), none of them provide all of the answers. Many well-proven simulation routines are available (e.g., SIMNON, as mentioned previously), so that aspect of nonlinear CACSD presents no problems.

"Closing the Loop" Around the Design Process

Once a designer has explored the behavior of the nonlinear plant, determined the desired operating conditions(s), and obtained linearized models at those points, he will generally use an existing linear CACSD package to design an appropriate controller. Since the resulting closed-loop system design is based on an idealized model, it is usually necessary to simulate the system using the more realistic nonlinear plant model in conjunction with the controller. This often results in an iterative controller design procedure, wherein the designer refines the plant model, adds features or "fixes" to it, performs the exploratory steps outlined above, and redesigns controllers until he is satisfied that the model is sufficiently realistic and that the closed-loop system including the nonlinear plant behaves satisfactorily. This control systems design methodology has been illustrated in Figure 1.

Many essential tasks in control system design for nonlinear plants have been discussed above, in some detail. From the iterative flow of this process, it is clear that automating this procedure will be highly beneficial, in terms of obtaining a better design, and increasing engineering productivity.

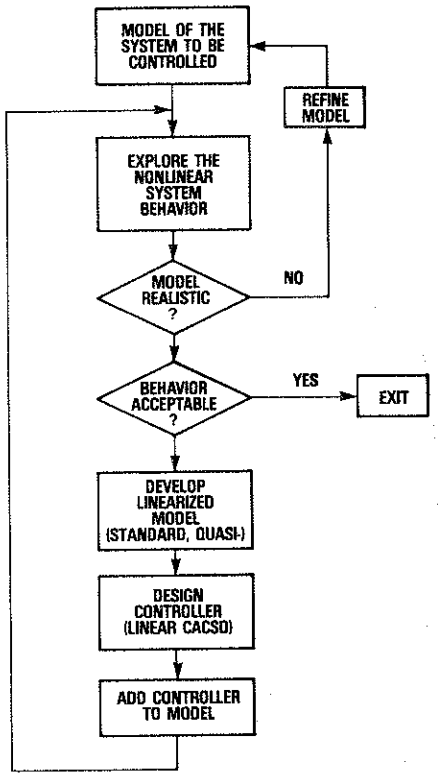

Figure 1. Control system design process.

CACSD ENVIRONMENT

The procedures delineated above can be greatly facilitated by the appropriate computer-aided environment. The functional diagram of Figure 2 illustrates the concept presently being implemented at General Electric Corporate Research and Development (GE CRD) by the author and his colleagues (most notably, H. Austin Spang, III). The manner in which this interactive (command-driven) CACSD package addresses the needs presented above is as follows.

1. *Modeling.* Developing a realistic and appropriately detailed model of the plant is the most critical requirement for control system design. The environment depicted in Figure 2 provides a

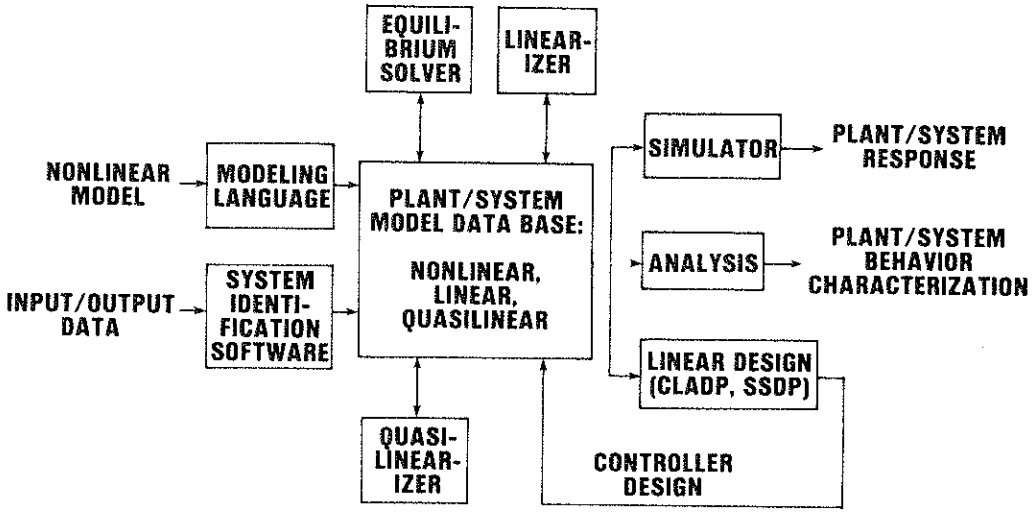

Figure 2. Functional CACSD environment.

unified model database, convenient *modeling languages* (the SIMNON language mentioned above, based on Algol, which we are presently extending to include matrix/vector model formulations; and Fortran) a *parameter identification package* (IDPAC, also from Lund; Wieslander, 1976), and a *simulator* or interactive numerical integration and display routine (SIMNON) to study the behavior of the plant and to validate or refine the model.

2. *Regime definition.* Several features of the construct in Figure 2 aid in regime selection. The *simulator* allows the user to determine operating points where the plant behaves well in terms of dynamic response; the *equilibrium solver* (added by GE CRD to SIMNON as a new interactive command) allows the user to obtain the exact steady-state value of the state for a given constant input vector, and the *linearizer* (also added directly to SIMNON by us) permits the analyst to study the small-signal behavior of the plant near any operating point using *analysis* routines, such as eigenvalue/eigenvector solution, etc.
3. *SIDF model development.* Currently, the user must supply his own quasilinear model of the plant. We envision implementing two approaches to SIDF modeling: an extensive *catalogue* of SIDFs as in Atherton (1975) or Gelb and Vander Velde (1968), and a combined *simulation/fast fourier transform* "direct method" to determine $\{A_{df}, B_{df}, C_{df}, D_{df}\}$ numerically, as outlined above.
4. *Control system design.* The CLADP design suite is currently the only completely checked out and integrated approach available. An interactive state-space design package (SSDP) (Spang, 1981) is defined and being incorporated. The CACSD environment has been conceived to allow any other design package(s) to be added with little effort. The design process results in a specification of a control system structure, and additional subsystems ("compensators") that are written directly into the system model database in the SIMNON modeling language.
5. *Control system validation.* The "closed-loop" structure of the CACSD environment, whereby the controller design is directly added to the system model database without user intervention, is ideally suited for testing the controller design realistically (in conjunction with the nonlinear plant model) and performing sensitivity analysis.

The basic CACSD structure depicted in Figure 2 has been completed, except for the direct interface of IDPAC-identified system models into the model database, and direct SIDF model generation. Even at this preliminary stage of development, it has been demonstrated to be an efficient, flexible, and effective tool for designing controllers for nonlinear plants.

SUMMARY AND CONCLUSION

The ultimate goals of CACSD are reducing the engineering effort involved in control systems design to the greatest extent possible, and achieving the best possible designs. While the linear system CACSD

software developed in the last decade addresses very important aspects of the control system design process, it is hoped that the software concept presented here that permits taking a more global approach to this problem will prove to be valuable, especially in light of the need to deal with plant nonlinearity in an effective manner.

REFERENCES

D.P. Atherton (1975), *Nonlinear Control Engineering,* Van Nostrand Reinhold, London.

G. Dahlqvist and A. Bjorck (1974), *Numerical Methods,* Prentice-Hall, Englewood Cliffs, N.J.

J.M. Edmunds (1979), "Cambridge Linear Analysis and Design Programs," IFAC Symposium on Computer Aided Design of Control Systems, Zurich, Switzerland, pp. 253-258.

H. Elmqvist (1975), "SIMNON - An Interactive Simulation Program for Nonlinear Systems," Report 7502, TFRT-3091, Lund Inst. Tech., Dept. Auto. Control, Lund, Sweden.

A. Gelb and W.E. Vander Velde (1968), *Multiple-Input Describing Functions and Nonlinear System Design,* McGraw-Hill, New York.

A.G.J. MacFarlane, B. Kouvaritakis, and J. M. Edmunds (1977), "Complex Variable Methods for Multivariable Feedback Systems Analysis and Design" *Proc. Alternatives for Linear Multivariable Control,* Natl. Engg. Consortium, Chicago, Ill.

J.J. More, B.S. Garbow, and K. E. Hillstrom (1980), "User Guide for MINPAC-1," Argonne Natl. Lab. Report ANL-80-74, Argonne, Ill.

T. Posbergh and S. Chen (1980), "Dynamic Control Systems Software-Users' Manual," MIT-LIDS Report, Mass. Inst. of Tech., Cambridge, Mass.

H.H. Rosenbrock (1969), "Design of Multivariable Control Systems Using the Inverse Nyquist Array," *Proc. IEE (UK) 116, 1929-1936*

H.A. Spang III (1981), State Space Design Program (SSDP) General Electric Internal Report, Schenectady, N.Y.

G. Stein, and S. Pratt (1981), "LQG Multivariable Design Tools," AGARD-LS-117.

J.H. Taylor (1980), "Applications of a General Limit Cycle Analysis Method for Multivariable Systems" *Nonlinear System Analysis and Synthesis: Vol. 2 - Techniques and Applications, ch. 9* R.V. Ramnath, J.K. Hedrick, and H.M. Paynter, ed., Amer. Soc. of Mech. Engrs.

J.H. Taylor, C.F. Price, J. Siegel, and A. Gelb (1980), "Covariance Analysis of Nonlinear Stochastic Systems via Statistical Linearization," *Nonlinear System Analysis and Synthesis: Vol. 2 - Techniques and Applications, ch. 13,* R.V. Ramnath, J. K. Hedrick, and H. M. Paynter, ed., Amer. Soc. of Mech. Engrs., New York.

J.H. Taylor (1982), "Robust Control Systems Design for Nonlinear Plants," submitted to *Applications of Multivariable Systems Theory,* Manadon, UK, October 1982.

J.Wieslander (1976), "IDPAC-User's Guide," Report 7605, TFRT 3099, Lund Inst. Tech., Dept. Auto. Control, Lund, Sweden.