IFAC/IMACS Symposium on Computer Aided Design in Control Systems, University of Wales, Swansea, UK, July, 1991

CAE TOOLS FOR NONLINEAR SYSTEMS ANALYSIS AND DESIGN BASED ON SINUSOIDAL-INPUT DESCRIBING FUNCTIONS

James R. O'Donnell, Jr.
Department of Electrical, Computer, and Systems Engineering
Rensselaer Polytechnic Institute
Troy, New York 12180-3590

James H. Taylor
Control Systems Laboratory
General Electric Corporate Research and Development
PO Box 8, Schenectady, New York 12301

Abstract: We report on recent progress in the development of a computer-aided engineering (CAE) environment for nonlinear control system analysis and design based on sinusoidal-input describing function (SIDF) methods. Several major additions have been made to our nonlinear controls CAE software: ACSL macros were developed to allow the generation of SIDF models of nonlinear plants in a manner analogous to that of the SIMNON-based software developed earlier, and MATLAB routines were developed for the analysis of these models and for the design of general nonlinear controllers based on them. This software provides an integrated tool set for treating very general nonlinear systems with no restrictions on system order, number of nonlinearities, configuration, or nonlinearity type. Based on the new software presented here, the use of SIDF-based nonlinear control system analysis and design methods is substantially easier to carry out and more powerful than before.

Keywords: Nonlinear control systems, describing functions, control system design, computer-aided design, computer software.

INTRODUCTION

The theoretical basis for the work described herein has been described in earlier publications: Taylor (1982, 1983) lays the foundation for control design for systems with amplitude-dependent nonlinearities via quasilinearization using sinusoidal-input describing function (SIDF) techniques. Taylor and Strobel (1984, 1985a, 1985b) first describe the generation of nonlinear compensators with a single nonlinearity, generated by fitting gain-amplitude information to a piecewise-linear compensator gain nonlinearity, and then present the synthesis of fully nonlinear PID compensators using a frequency-response mapping technique. Taylor (1985) discusses software for the generation of SIDF models using simulation and Fourier analysis and for the synthesis procedure of (Taylor and Strobel, 1984).

Taylor and O'Donnell (1990), a companion paper to this one detailing the new algorithms developed, extends these techniques and demonstrates a new design procedure for nonlinear rate feedback and PI cascade compensation. The application of this last method to design a fully nonlinear PI compensator and rate feedback controller for a motor model with saturation and stiction is also presented. Nanke-Bruce and Atherton (1990) classify the different types of describing function characteristics encountered, describe an extension to the frequency-response mapping technique, and also show an alternative approach to the solution of the nonlinearity synthesis problem.

This paper describes two major additions to the CAE software for nonlinear controls detailed in (Taylor, 1985):

1. ACSL macros for the generation of SIDF models (these macros duplicate the functionality of the SIMNON-based software described in (Taylor, 1985), with a few extensions); and

2. MATLAB routines for the analysis of SIDF frequency-response models and for the synthesis of nonlinear controllers based on this information.

All the above nonlinear control design techniques use a set of SIDF models of the nonlinear plant as the basis for nonlinear compensator synthesis. SIDF models are used because they provide an excellent characterization of the major nonlinear effect with which we are concerned: the sensitivity of the nonlinear plant's input/output (I/O) behavior to the amplitude of the input signal; this issue has been discussed in detail in (Taylor, 1982, 1983; Taylor and Strobel, 1984). In summary, given an input in the form $u(t) = u_0 + a_i \cos(\omega t)$ the I/O model is of the form

$$y(t) = y_0 + Re[G(j\omega; u_0, a_i) a_i e^{j\omega t}] \qquad (1)$$

where higher harmonics are neglected in this representation. A set of SIDF models corresponding to an amplitude set $\{a_i\}$ is denoted $\{G(j\omega; u_0, a_i)\} = \{G_i\}$.

Once a set of SIDF models is available, the synthesis of a nonlinear compensator proceeds as follows: first, a *linear compensator set* is designed based on these models, one for each input amplitude a_i, with the objective of making the overall open-loop control system as insensitive to input amplitude as possible. This yields a parameterized set of compensators $\{C_i(a_i)\}$, where the configuration of each compensator is the same (e.g., PID) but the parameters differ (e.g., $\{K_{P,i}(a_i)\}$, $\{K_{I,i}(a_i)\}$, $\{K_{D,i}(a_i)\}$). Final synthesis of the nonlinear control system is then accomplished by SIDF inversion to determine the required compensator nonlinearities.

The particular approach presented here is the same as in (Taylor and O'Donnell, 1990), and involves the design of a tachometer inner-loop to provide nonlinear rate feedback and a nonlinear PI compensator in cascade with the resulting rate-feedback-compensated plant; hereafter this will be referred to as a *PI+Tach* controller to distinguish it from PID in the forward path (Taylor and Strobel, 1985a, 1985b). In general, there is no restriction as to compensator structure except that the linear controller set and final nonlinear controller must be of the same type.

This approach and corresponding software can treat nonlinear plants of a very general type, with no restrictions as to system order, number of nonlinearities, configuration, or nonlinearity type. These routines implement SIDF-based nonlinear controller design methods that are substantially more effective than our earlier CAE tools (Taylor, 1985). It is also believed that these results will provide a framework for further developments in the realm of analysis and design of nonlinear systems.

NONLINEAR PI+TACH DESIGN

First, it is important to state the premises of the SIDF design approaches that we have been developing:

1. The nonlinear system design problem being addressed is the synthesis of controllers that are effective for plants having frequency-domain I/O models that are *sensitive* to input amplitude (e.g., for plants that behave very differently for "small" and "large" input signals).
2. The primary objective of nonlinear compensator design is to arrive at a closed-loop system whose response is as *insensitive* to input amplitude as possible.

This encompasses a limited but important set of problems, for which gain-scheduled compensators cannot be used and for which other approaches (e.g., variable structure systems, model-reference adaptive control, global linearization) do not apply because their objectives are different (e.g., their objectives deal with *asymptotic* solution properties rather than *transient* behavior, or they deal with the behavior of *transformed* variables rather than physical variables).

The design algorithm for the nonlinear PI+Tach controller proceeds as follows:

1. Select a set of input amplitudes and frequencies that cover the operating regimes of interest for the plant under consideration.
2. Generate SIDF models of the plant corresponding to these input amplitudes and frequencies.
3. Examine these SIDF models to qualitatively determine:
   - appropriateness of the design approach,
   - appropriateness of the amplitude and frequency set,
   - severity of the nonlinear plant amplitude sensitivity, and
   - type of nonlinear controller that is likely to be needed.
4. Design a nonlinear inner-loop tach feedback compensator using a D'Azzo and Houpis algorithm (1960) extended to the nonlinear case (Taylor and O'Donnell, 1990).
5. Find SIDF models for the nonlinear plant plus nonlinear rate feedback.
6. Design a cascade nonlinear PI compensator using an extension of the frequency-response mapping technique described in (Taylor and Strobel, 1985a, 1985b).
7. Validate the design through simulation.

The resulting structure is shown in Fig. 1.

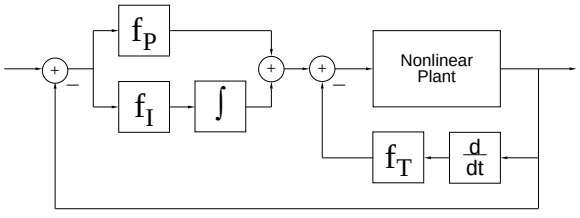

Fig. 1: Nonlinear PI+Tach Controller Structure

The software presented herein completely supports all but items 1 and 7 above, namely, selection of the input amplitudes and frequencies for SIDF model generation and simulation to validate the design, which depend in large part on the designer's judgment and familiarity with the system in question. The generation of SIDF models can be done using the SIMNON-based software described in (Taylor, 1985), or using the ACSL macros described here. Item 3, the qualitative analysis of SIDF models, is a current area of research, though the MATLAB routines that have been developed to facilitate this analysis will be described briefly. Finally,

the MATLAB routines which implement the two design steps, items 4 and 6, will be discussed below in detail.

ACSL SIDF GENERATION

The generation of sinusoidal-input describing function models that provide an amplitude-dependent I/O characterization for a nonlinear plant has been dealt with in detail in (Taylor and Strobel, 1984; Taylor, 1985). The two basic approaches to this—solving the nonlinear algebraic equations derived from the principle of harmonic balance and simulation coupled with Fourier analysis—have been described in those works. Here we focus on the latter approach.

The second technique is easier to implement, given a good package for integrating nonlinear differential equations, and avoids the need to justify the assumption that the inputs of every nonlinearity are nearly sinusoidal—there is no such assumption made using simulation. The only assumption is that a frequency-domain amplitude-dependent I/O model provides a good representation of the behavior of a nonlinear plant for control system design; that issue has been discussed in (Taylor, 1983; Taylor and Strobel, 1984). In our opinion, while SIDF models are not exact, a *set* of SIDF models covering the range of input amplitudes that will be encountered provides an excellent basis for "robust design", in the sense that the sensitivity of the plant behavior to input amplitude is one of the most important issues in robustness, and the SIDF I/O model is the least conservative model that accurately takes this factor into account.

As mentioned above, extensions to the nonlinear simulation package SIMNON for SIDF I/O model generation are discussed in (Taylor, 1985). The functionality of that software has been reproduced with a set of ACSL macros that can be included in an ACSL system model to allow SIDF model generation. The discussion that follows assumes some familiarity with the use of ACSL for modeling and simulation.

There are four ACSL macros, each of which must be included in a separate section of the model. These are named `DFINIT`, `DFFOUR`, `DFCHEK`, and `DFTERM`; they each use the same two arguments, i.e., the names of the input and output variables of the ACSL model to be used for SIDF I/O modeling.

The macro `DFINIT` is included as the first executable statement of the `INITIAL` section of the ACSL system model; it initializes the Fourier integrals before each simulation run and provides the logic for making multiple runs at different input amplitudes and frequencies. It also provides set-up variables allowing the user to specify the input amplitudes and frequencies at which SIDF models will be generated at run time: `ALIST` and `WLIST` allow a manual list of input amplitudes and frequencies, respectively, while `WINPUT` allows the definition of either a uniformly or logarithmically spaced range of input frequencies. Additionally, in an extension to the SIMNON-based software for SIDF generation, the macro `DFINIT` supplies the set-up array `CINPUT` which allows the user to define a set of ACSL *communication intervals* (one of the parameters ACSL uses to control integration step size) to be used at different input frequencies. This allows the user to adjust the integration step size for various frequencies to obtain accurate yet economical simulation throughout the run.

The macros `DFFOUR` and `DFCHEK` are included in the `DYNAMIC` section of the ACSL model. `DFFOUR` must be within the `DERIVATIVE` subsection describing the continuous dynamics of the model, and `DFCHEK` is located outside the `DERIVATIVE` subsection. `DFFOUR` sets up the sinusoidal input to the system model, and then uses the system's output to generate the Fourier integrands needed to define the SIDF model; ACSL will integrate these during simulation along with the system states. `DFCHEK` creates a `DISCRETE` section that is called at the end of every cycle (via the ACSL `SCHEDULE` command) to check for convergence of the Fourier integrals. The convergence tests and parameters used are identical to those used in the SIMNON-based software described in (Taylor, 1985); refer to that paper for details. One other small extension to the SIMNON-based software incorporated in the `DFCHEK` macro is that it outputs a summary each cycle of the convergence tests. In the event that convergence fails, this summary can be used to determine why.

Both the SIMNON and ACSL software are also capable of generating Fourier coefficients for the second and third harmonics of the system's output. This information can be used in initial runs to serve as a diagnostic aid in justifying the use of SIDF models for the nonlinear plant; in later runs, the generation of these higher-order coefficients can be disabled to save computer time.

Finally, the `DFTERM` macro is placed in the terminal section of the model. It outputs the SIDF model information for each input amplitude and frequency at which convergence is achieved, or displays an error message if not. It then transfers control to the `DFINIT` macro to run the next case (input amplitude/frequency), or terminates if all cases have been done.

MATLAB ANALYSIS AND DESIGN

The MATLAB tools developed for the analysis and design of nonlinear control systems can be divided into three general categories:

1. SIDF frequency-response manipulation and display, including importing data from ACSL or SIMNON SIDF model generation;
2. linear controller-set design algorithms; and
3. nonlinear controller synthesis via SIDF inversion.

For the procedure presented below and in (Taylor and O'Donnell, 1990) there are two instances of linear controller-set design algorithms (category 2), namely rate feedback and PI cascade compensator design.

SIDF Model Manipulation and Display

The first, and most important, consideration in applying any nonlinear control design approach is to analyze the system to make sure the approach is applicable. Towards this end, various MATLAB functions were written to allow the easy import, display, and manipulation of SIDF frequency-response models.

The first step is to load the SIDF models generated by either SIMNON or ACSL, and put them in a form convenient for further analysis within MATLAB. The function `load_df` was written to take the name of the SIDF output file(s) and load the data into a series of MATLAB arrays. The information read in includes the list of input amplitudes, frequencies, and DC offsets used for the SIDF generation, as well as the SIDF models themselves. This function also creates a string variable to be used in the title of all of the plots made of this information.

A menu-driven MATLAB function, `plot_df`, was written to allow the display of the SIDF frequency-response models. This function includes many options, allowing plots in Bode, Nyquist, and Inverse Nyquist formats. It also allows the display to be restricted to various input amplitudes or frequencies, and includes various other display and hardcopy options. A sample Bode phase plot from `plot_df` is shown in Fig. 2. Finally, there are other support functions that allow

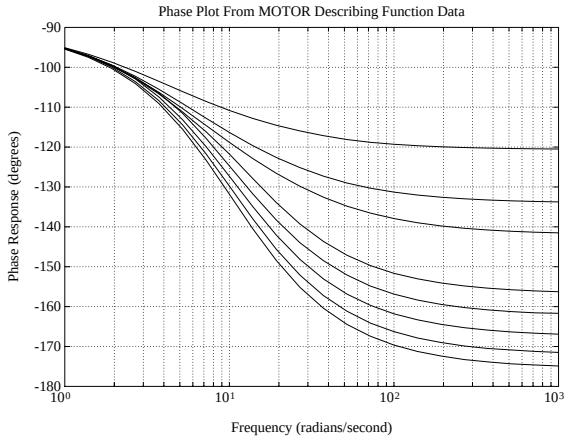

Fig. 2: SIDF Model Phase Response

the SIDF frequency-response information to be converted between different forms (e.g., magnitude in dB or relative gain, phase in degrees or radians), and to be combined in various ways with other frequency-response matrices.

Rate-Feedback Controller-Set Design

The general objective in using inner-loop rate feedback is stabilizing and increasing the damping the system, if necessary, and reducing the sensitivity of the system to disturbances and plant nonlinearities. We particularly wish to design a *nonlinear* tachometer loop to desensitize the inner-loop as much as possible with respect to input amplitude.

As shown in D'Azzo and Houpis (1960), it is convenient to work with inverse Nyquist plots of the plant I/O model, i.e., to invert the SIDF frequency-response information in complex-gain form and plot the result in the complex plane. In the linear case, this allows us to study the closed-inner-loop (CIL) frequency response $G_{CIL}(j\omega)$ in the inverse form

$$\frac{1}{G_{CIL}(j\omega)} = \frac{1 + G(j\omega)H(j\omega)}{G(j\omega)} = \frac{1}{G(j\omega)} + H(j\omega) \quad (2)$$

where the effect of $H(j\omega)$ on $1/G_{CIL}(j\omega)$ is easily determined, particularly when using rate feedback and $H(j\omega) = j\omega K_t$.

The inner-loop tach feedback design algorithm given by D'Azzo and Houpis uses a construction amenable to extension to nonlinear systems. For linear systems, this algorithm is based on adding a tachometer and external gain in order to adjust the inverse Nyquist plot to be tangent to a given M-Circle at a selected frequency. Referring to Fig. 3, the algorithm is ap-

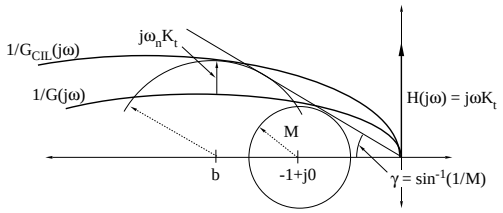

Fig. 3: Tach Feedback Design Algorithm 2

plied as follows: The point corresponding to the desired value of natural frequency ω_n is located, and the projection of this point onto the real axis (point b) is made the center of the scaled M-Circle (the original M-Circle, of radius M, scaled by the external gain A). The radius of this scaled M-Circle must be chosen to make it tangent to the line defined by the angle γ. Next K_t is determined to move the $1/G(j\omega)$ plot to be tangent to this scaled M-Circle, giving the plot $1/G_{CIL}(j\omega) \approx 1/G(j\omega) + j\omega K_t$. The gain A can then be determined by the distance to point b. Due to the complicated geometry involved when actually implementing this, some trial and error may be required; the software is designed to support this.

This algorithm has been extended to the nonlinear case, and is implemented by the MATLAB function `rfd2`. This function takes as input the SIDF frequency-response information of the plant, along with the desired M-Circle radius and natural frequency—these can be chosen to achieve desired time response and overshoot for the resultant system. For each input amplitude a_i, a tachometer gain, $K_{t,i}$, and external gain A_i is found. At this point in the design, the A_i values are not used, since the external gain will be subsumed

in the cascade PI portion of the controller that is synthesized as shown below.

The final step in rate feedback design is scaling the original set of plant input amplitudes $\{a_i\}$ by the gain of the plant and tachometer at the natural frequency ω_n in order to determine the corresponding amplitude set $\{e_i\}$ at the input of the tachometer nonlinearity. The set of tachometer input amplitudes and desired tachometer gains $K_{t,i}(e_i)$ can then be used to synthesize the tachometer nonlinearity (f_T in Fig. 1), using the nonlinearity synthesis routines described below.

Nonlinearity Synthesis via SIDF Inversion

For each value of e_i the above algorithm specifies a linear rate-feedback gain K_i. This information serves as the basis for nonlinear controller synthesis, as follows. Given the gain versus amplitude relation of the form $K_i(e_i)$ that is to be achieved by a single nonlinearity, adjust the parameters of a specified piecewise-linear nonlinear function $f(e)$ so that the SIDF of $f(e)$ provides the best fit to the gain/amplitude relation $K_i(e_i)$ in the minimum mean square error sense.

The nonlinear function used here is slightly more general than that used in previous work (Taylor, 1985). It is a piecewise-linear, odd, memoryless, static nonlinearity with an arbitrary number of linear segments and discontinuities (optionally) at each breakpoint. It may thus be adjusted to fit virtually any gain/amplitude characteristic. (It is recommended that the user keep the nonlinearity simple, with a small number of breakpoints, to avoid over-fitting.)

A series of MATLAB functions have been developed to allow the parameters of this nonlinearity to be adjusted to fit the $K_i(e_i)$ data as closely as possible. The first, `plot_nl`, is used to plot both the $K_i(e_i)$ information and the gain/amplitude plot of the SIDF for the latest iteration of the nonlinearity being found. At the beginning of nonlinearity synthesis, the user can view this plot to determine approximately how many piecewise-linear segments are needed and to identify regions of abrupt change in the gain/amplitude relationship where a discontinuity may be included.

The next function, `init_nl`, is then called to make an initial guess at the nonlinearity parameters. The user is prompted for the number of linear segments to use; `init_nl` then sets the initial values of the nonlinearity parameters based on the gain/amplitude relationship being fitted. Areas of slope change, identified by the local maxima and minima of the gain/amplitude data, are used to set the initial values for the slope and breakpoint parameters of the nonlinearity. Once the initial values of the parameters are set, `inv_nl` is called to determine the nonlinearity's parameters so that the $K_i(e_i)$ data is fit by the nonlinearity's SIDF as closely as possible. It does this by setting up a call to a MATLAB MEX function, `sidfx`, which in turn calls a MINPACK subroutine to minimize the mean square error of the fit.

Once the nonlinearity has been found, `plot_nl` can be used to check the quality of the fit that has been achieved, and perhaps to determine where the fit can be improved. The user has the ability to change parameters manually, and also to fix certain parameters so that they will not be adjusted by MINPACK. In fact, all of the discontinuity parameters are fixed at zero when the nonlinearity is initialized; `init_nl` relies on the judgment of the user to identify where discontinuities are needed and free the corresponding discontinuity parameters.

The final function used for generating a controller nonlinearity is `write_nl`, which is used when the user is satisfied with the nonlinearity fitted to $K_i(e_i)$. This function writes either a SIMNON `continuous system` file which implements the controller nonlinearity and can then be connected into a SIMNON system description, or an ACSL command file which will set the parameters of the controller nonlinearity previously included in the system model.

Cascade PI Controller Design

Referring back to Fig. 1 and to the design strategy outlined above, the final step in the complete controller design is generating the nonlinear cascade PI compensator. The general idea is to first generate SIDF models for the nonlinear plant with nonlinear rate feedback over the range of input amplitudes and frequencies of interest, using the ACSL or SIMNON-based software described above. This information forms a frequency-response map as a function of both input amplitude and frequency. Linear PI controllers are then found for each amplitude so that the compensated open-loop system satisfies specified frequency-domain objectives and is as insensitive to input amplitude as possible. The set of PI compensator gains at various amplitudes is then used to synthesize a nonlinear PI compensator via SIDF inversion. This last step is facilitated by the fact that the nonlinearity synthesis functions described in the previous section can be used to directly obtain the PI controller nonlinearities and to create an ACSL or SIMNON description of them.

More specifically, given SIDF models for the nonlinear plant with nonlinear rate feedback, a single nominal input amplitude is selected, a^*, and a linear compensator is found for $G(j\omega; a^*)$. The selection of a^* and compensation of $G(j\omega; a^*)$ is based on standard considerations of robust design, e.g., select a^* so that $G(j\omega; a^*)$ has minimum gain margin and then synthesize $C(jw)$ to achieve desired bandwidth, gain margin and steady-state error. The resulting linear compensator is placed in series with the nonlinear plant and simulated with amplitude a^* to calculate the corresponding desired open-loop I/O model $CG^*(j\omega; a^*)$, the *frequency-domain objective function*. Then, at each input amplitude a_i a least-squares algorithm is used to adjust the parameters of the linear PI compensator, $K_{P,i}(a_i)$ and $K_{I,i}(a_i)$, to minimize the difference between the resulting frequency response found using the linear compensator and interpolating on the

SIDF frequency-response map, and $CG^*(j\omega;a^*)$, as described in (Taylor and Strobel, 1985). The nonlinear PI compensator is then obtained by synthesizing the nonlinearities f_P and f_I in Fig. 1 by SIDF inversion as described in the previous section.

This basic algorithm for PI design has been improved and extended in comparison with (Taylor and Strobel, 1985), to facilitate its use in a more general setting. The original implementation included no mechanism for adjusting the compensator parameter fit to emphasize or deemphasize specified frequency ranges. It used a PID controller, which has three channels covering low, medium, and high frequencies, so this ability was not necessary. However, we now wish to use a PI cascade compensator, and in general don't wish to assume that the compensator will affect all frequency ranges equally, so the ability to weight the different frequencies is needed. Additionally, because different input amplitudes may produce effects that are predominant over different frequency regions (e.g., in the application in (Taylor and O'Donnell, 1990) large input amplitudes produce major differences in the high-frequency part of the transient response while small input amplitudes give rise to low-frequency effects), it is useful to provide different frequency weights at each input amplitude. The ability to specify various frequency ranges to be weighted at different amplitudes to achieve desired closed-loop time-domain objectives has thus been added in a very flexible framework.

The algorithm described above is implemented in a single MATLAB function `design` which in turn sets up a call to the MEX function `pides`. This function then calls a MINPACK subroutine which performs the least-squares procedure outlined above in order to set the parameters $\{K_{P,i}(a_i)\}$, $\{K_{I,i}(a_i)\}$ of the PI compensators in order to fit the frequency-response objective function as closely as possible according to the specified frequency-domain weighting function. The `design` function also provides a convenient way to set the weighting matrix, by allowing the user to easily select which range of frequencies to emphasize at each input amplitude. As mentioned previously, nonlinear PI synthesis is completed by using SIDF inversion to obtain the nonlinearities f_P and f_I in Fig. 1.

SUMMARY AND CONCLUSIONS

The algorithms and software outlined above are a specific realization of the basic concept of using SIDF I/O models as the basis for nonlinear compensator design proposed in (Taylor, 1982, 1983). We believe that this approach shows considerable promise in dealing with one of the more difficult problems in nonlinear systems design—the design of controllers to compensate for the amplitude-dependence of nonlinear plants. Developing software tools that function in a very integrated fashion, supporting them by a nonlinear simulation package (ACSL or SIMNON), and programming all other functions within MATLAB, results in a flexible and extensible framework for the design of nonlinear control systems.

REFERENCES

[D'Azzo and Houpis, 1960] D'Azzo, J. J. and Houpis, C. H. (1960). *Feedback Control System Analysis and Synthesis.* McGraw-Hill Book Company, New York.

[Nanke-Bruce and Atherton, 1990] Nanke-Bruce, O. and Atherton, D. P. (1990). Design of nonlinear controllers for nonlinear plants. In *Proceedings of 1990 IFAC World Congress*, Tailinn.

[Taylor, 1982] Taylor, J. H. (1982). Robust computer-aided control system design for nonlinear plants. In *Application of Multivariable Systems Theory*, Manadon, Plymouth, UK.

[Taylor, 1983] Taylor, J. H. (1983). A systematic nonlinear controller design approach based on quasilinear system models. In *Proceedings of the 1983 American Control Conference*, San Francisco, CA.

[Taylor, 1985] Taylor, J. H. (1985). Computer-aided control engineering environment for nonlinear systems analysis and design. In *Proceedings of the 3rd IFAC Symposium on CAD in Control and Engineering Systems*, Lyngby, Denmark.

[Taylor and O'Donnell, Jr., 1990] Taylor, J. H. and O'Donnell, Jr., J. R. (1990). Synthesis of nonlinear controllers with rate feedback via SIDF methods. In *Proceedings of the 1990 American Control Conference*, San Diego, CA.

[Taylor and Strobel, 1984] Taylor, J. H. and Strobel, K. L. (1984). Applications of a nonlinear controller design approach based on quasilinear system models. In *Proceedings of the 1984 American Control Conference*, San Diego, CA.

[Taylor and Strobel, 1985a] Taylor, J. H. and Strobel, K. L. (1985a). Nonlinear compensator synthesis via sinusoidal-input describing functions. In *Proceedings of the 1985 American Control Conference*, Boston, MA.

[Taylor and Strobel, 1985b] Taylor, J. H. and Strobel, K. L. (1985b). Nonlinear control system design based on quasilinear system models. In *Proceedings of Control '85*, Cambridge.