OPTIMIZATION APPROACH TO THE FREQUENCY DESIGN OF COMPENSATORS FOR NONLINEAR SYSTEMS WITH DEAD TIME

by

Marios Stavrou

Dissertation submitted in compliance with the requirements for Masters Degree in Technology in the Department of Electrical Engineering (Light Current) at Technikon Natal.

This Dissertation represents my own work

M. Stavrou

APPROVED FOR FINAL SUBMISSION

December 14, 1998

Professor Vladimir B. Bajic, Pr.Eng., D.Eng.Sc.(EE) Date
Supervisor

Durban, 14th of December, 1998

Acknowledgements

I express my sincere gratitude to Professor Vladimir Bajic for his help, patience and the long hours of discussions, which we have had regarding this project. Without his help, knowledge and encouragement this project would never have been finalized. I would also like to thank Professor Bajic's family for their hospitality and patience during my frequent work–related visits to their home.

I also wish to thank Mr Poobalan Govender, Mr Meredith McLeod, Mr Stefan Mare and Dr Kevin Duffy for their invaluable comments made during several readings of the previous drafts of this dissertation. Their comments helped me a lot to more clearly express my thoughts and to finalize the text. Special thanks are due to Pooby who has spent many weekends of his free time to help me in this exercise.

Lastly, but nevertheless most importantly, I wish to thank my wife for her patience and understanding.

ABSTRACT

Designing compensators in the frequency domain is a complicated problem even for linear systems that have dead time. The situation is far more difficult if the system is also nonlinear. This study introduces a new method for the design of compensators for time-invariant, nonlinear systems that have dead time. The method is based on an optimization approach and utilizes large signal linearization methodology.

The problem can be defined as follows: There is a nonlinear plant that has dead time and whose frequency characteristics are not as required. There is another, reference, system, which is linear, time invariant, and possibly with dead time, which possesses desired frequency characteristics. A compenator has to be designed so that the system, which consists of the original plant and the compensator in the closed loop configuration, obtains the frequency characteristics of the reference system.

The process of designing compensators for the above-mentioned problem consists of two phases. First, the major nonlinearity of the plant is subject to partial linearization. The linearizing block is part of the compensator. This linearization is performed in the frequency domain. Then, in the second phase, the linear part of the compensator is tuned to adjust the overall frequency characteristics of the closed loop system. The design process is computationally intensive, as the minimization of the suitably defined error has to ensure that the deviation of the behavior of the system is minimal compared with the behavior of the reference system in the specified frequency range.

The proposed method has been tested on several examples and shows good performance in the selected frequency ranges. Also, the behavior of system in the time domain appears to be satisfactory. The method can be applied to problems associated with system design in the domains of control and communications.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Most physical systems have many nonlinear components, as well as the so-called dead time. These systems may become quite complex and difficult to analyze and control when using the traditional methods of mathematical analysis and design. The nonlinearities in the system pose serious problems when attempting to use methods based on linear models. Additionally, if nonlinear systems have a non-negligeable dead time, in general they remain a challenge for engineering analysis and design, as there are no, as yet, efficient methods for analyzing such systems.

In this study we attempt to find a feasible technology to design frequency compensators for these systems. The method will tackle the problem from a different angle than that of conventional time domain or frequency domain analyses. This method was proposed by Bajić (1996) and in essence was based on application of optimization techniques to find the most suitable compensator for nonlinear time delayed systems in the frequency and combined time-frequency domains. The method developed in this study relates to the frequency domain and will allow the design of compensators for different system models having very long dead times and a major nonlinearity. The manual design of the compensator in many instances can be eliminated and the method developed should

generally provide a different basis for compensator (controller) design in the frequency domain. There are, however, some other, similar methods, that are used for problem of identification of linear time invariant systems (Pintelone *et al.* 1992, Schoukens and Pintelon 1991, Kollar 1993, 1994). However, these methods are not developed for nonlinear cases, and they are not directly applicable to compensator design problem.

Frequency response methods are generally suitable for the design and analysis of linear time invariant systems (Bode 1945, Nyquist 1932, Horowitz 1963, Jackson 1991, Phillips & Harbor 1988, Narendra & Valvani 1978, Narendra & Lin 1980, Saunders *et al.* 1989). Although these methods have been adapted to deal with specific problems of nonlinear systems, they, however, are not very suitable for the study of nonlinear systems. In most cases they produce insufficiently accurate results when applied to nonlinear systems. The situation becomes much more complicated if the system possesses a significant inherent dead time.

Design of compensators (controllers) for time delayed nonlinear systems is generally undeveloped. Hence our interest to propose a feasible method that can be used to solve a problem of compensator design for a class of such systems. In the solution of this problem we resort to optimization methods in the frequency domain. It is also hoped that this frequency domain optimization will lead also to the acceptable behavior of the compensated system in the time domain - although this is not the objective of the compensator design request in this study.

In our study the general system for which a compensator will be designed has the following components:

- the linear or dynamical part,
- the nonlinear part,
- the dead time.

The method to be proposed for the compensator design requires that the structure of the compensator be known. In general, we consider the compensator with the following

parts:

- the nonlinear part,
- the linear part.

Because the design procedure to be developed is very universal in the sense that it is applicable to a wide class of nonlinear systems (not necessarily for the nonlinear systems in the form that we considered), and because it can be implemented completely as a computer program based on Digital Signal Processing (DSP) (see Proakis & Manolakis 1992, Kraniouskas 1992) and general optimization techniques (Grace 1994, More & Wright 1993, Gill *et al.* 1981, Fletcher 1987), we expect that the results of this research may have many positive implications in the computer assisted design of compensators. We also expect that the results to be developed may lead to an efficient compensator design method in the frequency domain for nonlinear time delayed systems.

1.2 The framework for the solution

In the upper part of Fig.1-1 a typical system, that will be optimized using our approach, is depicted. The system is composed of the following components:

- D_s is the transfer function of the linear part of the system; it will be denoted by $D_s(s)$, where s is the complex variable in the Laplace transform domain;
- N_s denotes the nonlinear part (block) of the system; the nonlinearity itself does not contain dynamic elements;
- T_s denotes a block that accumulates the apparent dead time in the system; we usually have this dead time being represented by the transfer function $e^{-\lambda s}$, where λ is the dead time.

This system is closed in the loop with the compensator. The compensator will be used to adjust the behavior of the original system. The original system by assumption has behavior different from the desired one. In our study the desired behavior is the behavior of the system depicted in the bottom of Fig.1-1. Since we want that the closed loop system (which is now nonlinear and with possibly very long dead time) behave as close as possible to the reference system with the desired behavior, the most suitable approach seems to be via optimization techniques. The compensator that we have introduced into the system to be optimized is made of the following elements:

- D_c is the transfer function of the linear part of the compensator; its tuning is done in the second phase of the design process;
- N_c denotes the nonlinear part of the compensator; it will be used mainly to compensate for the nonlinearity N_s in the system; It will be tuned separately from D_c.

In the ideal case we require that after the optimization (after the compensator has been designed) the compensated system behave as the reference system. The reference system is represented as L_d in Fig.1-1. The reference system that we consider is linear time invariant, although the method can be adjusted to deal with a nonlinear time invariant time delayed reference system.

The compensator design utilizes an optimization procedure to obtain certain parameter values for the compensator model. Optimization of this system is a very complex procedure. In order to determine the necessary change in the current compensator parameter values, during the search for the optimal values, the optimizer requires the simulation of the closed loop system repeated a sufficient number of times; after each simulation cycle the specific error function has to be calculated and fed to the optimizer, which uses this in the determination of better values for compensator parameters. Iterative repetition of these steps will finally provide the optimum values of the compensator parameters. The error function is determined by means of the frequency domain representation using

a suitably defined difference between the desired system characterization and the closed loop system with the current parameter values.

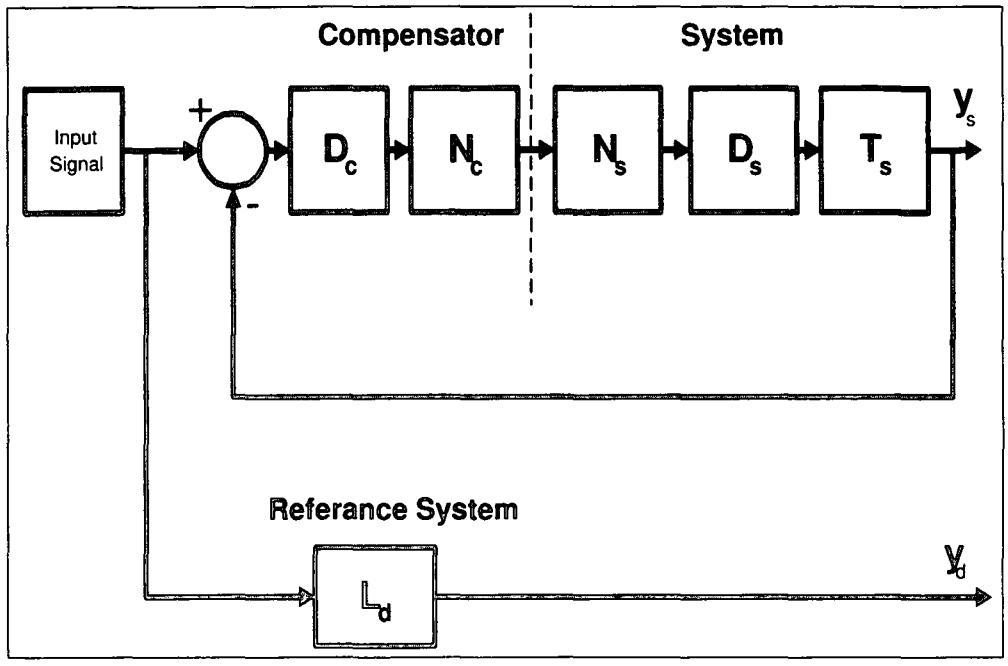

Figure 1-1: Model representation of our problem

1.3 Current design methods

Here we mention some of the existing methods used in the design of nonlinear systems that possibly may have dead time. Two points should be noted. First, there are no efficient methods to deal with the problem of compensator design for these systems in the frequency domain. Second, other methods that can be used for a partial solution of this problem are developed in the field of control systems. However, none of them can handle the problem mentioned above. In what follows we will refer to those methods we consider partly relevant to our design problem.

1.3.1 Typical Control System

A typical simplified structure of a control system is given in Fig.1-2. Engineering requests normally are that the system output y should be as close as possible to the desired (reference) input r. The difference between r and the actual output of the system y is called the error and is denoted by e ($e = r - y$). By default, the system output will differ from r; so we insert a device called the controller and close the loop. The insertion of the controller is aimed at changing the behavior of the system, so that, with the properly designed controller, the closed loop system can behave much closer to what designer wants. Input to the controller is generally e. The controller should function in such a way so as to attempt to minimize the deviation of the system output from the desired values of r. If the whole loop is designed properly, the output y will always follow the input r with very small discrepancy. However, the efficiency of this is based on how good the design of the compensator is, as well as on our knowledge of the characteristics of the system to be controlled. At this point we should indicate that in this study we will use the term compensator instead of controller, simply to indicate that the major objective of the design is to achieve a specific characteristic in the frequency domain of the compensated system. Alternatively, when the term controller is used (in this study) the implication is that design requests need not be limited to the frequency domain only.

The controller design process

The designer who wishes to achieve system compensation using the control scheme depicted in Fig.1-2, will usually follow the 3-stage process indicated below:

1. modeling of system behavior to get a mathematical description suitable for design; this is usually done using 2 methods:

   (a) by applying the laws of physics to determine the system model from first principles and/or,

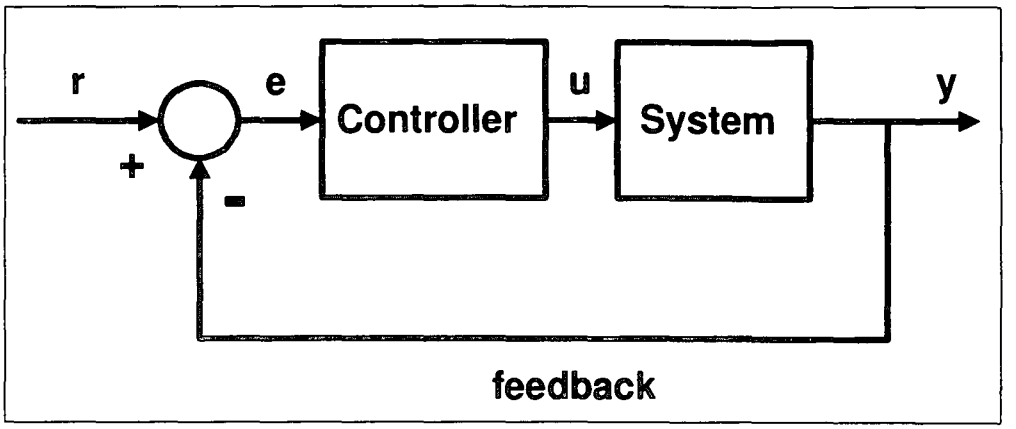

Figure 1-2: Standard feedback controller loop.

(b) by the use of empirical methods to determine the system model from the observed behavior of the system (usually known as system identification);

2. design the controller to achieve the design objectives using the process model obtained in 1/;

3. implementation and validation of the designed controller via simulation and/or real world implementation to assess if the closed loop system has achieved the desired objectives.

1.3.2 Nonlinearities in Engineering Systems

Nonlinearities are inherent to all real-world systems (Macfarlane 1970, Khalil 1992). The nonlinear character of the systems is often ignored and a simplistic approach in analysis and design is utilized. This is done mainly due to the greater simplicity of methods that deal with linear systems. However, nonlinear phenomena cannot be adequately handled by linear system design methods. Conventional linear analysis may frequently yield unrealistic results. Thus the necessity to take into account nonlinear phenomena in critical applications.

Some of the negative effects which nonlinearities have on the performance of engineering systems are given below:

1. degeneration of static accuracy and precision in the presence of small signals (dead band effect);

2. bandwidth variations with input amplitude; this is a very common situation in control systems (Coals 1956);

3. stability for some amplitudes of signals and instability for others;

4. development of subharmonics and/or superharmonics of the input frequency at the output of a nonlinear system;

5. the jump phenomenon may occur.

The analysis of a nonlinear system is far more difficult than that of a linear system (Khalil 1992). Because the superposition principle does not apply to nonlinear systems we cannot find the response to a complex signal by summing the individual responses to constituent component signals of the input, which is the general background of many methods for analysis and design of linear systems. Due to the above mentioned problems, and a number of others, the design of the controller for a nonlinear system is a very complex task and the general design methodology does not exist. Some specific methods are mentioned below.

1.3.3 Methods of analysis and synthesis of control systems containing nonlinearities

A few techniques used to investigate the design of nonlinear control systems will be mentioned.

Direct controller parameter selection

Using this method, system behavior is tested using a trial and error approach. The controller parameters are adjusted in a trial and error fashion to produce the desired system response. Synthesis can be achieved indirectly using a number of trial and error adjustments of parameter values. The disadvantage is that some controllers may have many adjustable parameters, so this approach to parameter adjustments make synthesis impractical. However, it should be noted that, although this technique is one of the least efficient, in the process control practice it is very customary for operators to tune control loops by trial and error. For a variety of reasons good selection of controller parameters by this approach may not be feasible. Thus the testing of the controlled system behavior may be impossible if the system is of the "one shot" type, such as a guided rocket, making the repeated trial and error testing costly and/or hazardous; another example may be an industrial process already in operation that cannot be interrupted for tests without expensive curtailment of production. When problems such as these exist, computer tests help to determine the optimum design parameters if a suitable mathematical model of the process is available. The usual technology of this approach is based on Monte Carlo method.

Computer simulation

In many cases nonlinear systems can effectively be mathematically modeled and used in computer simulation. In this way the designer can economically run a simulation of the system many times. Computer modeling eliminates some of the disadvantages of a direct method of testing. This method however is only applicable to systems that can be suitably represented by a well defined mathematical model and this frequently is not the case.

The case of mild nonlinearities - linearization

In instances where nonlinearities mildly deviate from the behavior of linear components, the nonlinearities can be reasonably well modeled as linear blocks (Franklin *et al.* 1990, Phillips & Harbor 1988). In this way all the known methods of linear analysis and design, such as the root locus method, Nyquist plots, etc. apply. However, these linear analysis and design techniques will yield inaccurate results when used for systems containing significant nonlinearities. Also, the discrimination between the mild nonlinearity and non-mild nonlinearity is relative.

Describing function method

In this approach certain nonlinear elements may be replaced by linear elements which operate on the main harmonic of the nonlinearity in the steady-state. The basic idea is that a sinusoidal input into a linear element will produce an output that has the same frequency as the input. The fundamental frequency harmonic is considered the most important in this approach and it relates the amplitude and phase of the fundamental harmonic of the nonlinear element output to the amplitude and phase of the sinusoidal input (Franklin *et al.* 1990, Phillips & Harbor 1988). However, as with any approximation, the accuracy of the results obtained with the describing function method is reduced. This type of analysis is only applicable to certain types of nonlinearities such as backlash (Nichols 1952), limiter (Kochenburg 1953), Coulomb friction (Boonton, Jr. 1952), etc.

Equivalent gains method

In this approach a memoryless nonlinearity is replaced by the so-called equivalent gain. Equivalent gain analysis is then performed. For a range of amplitudes the equivalent gain will take on a range of values and the roots of the equivalent system transfer function are examined in this range as if the gain was fixed. This technique can however be proved false in some instances (Franklin *et al.* pp. 551-558, 1990).

Lyapunov's second method

This is one of the general methodologies for dealing with nonlinear time delayed systems. However, there are not effective results that can be applied to the all desired cases. The principle of this approach is based on the fact that physical systems store only finite energy (Ogata 1996, Vydiasagar 1993). This energy is always being dissipated except at equilibrium when the system assumes its minimal energy state. The mathematical representation of the system "energy" is via the Lyapunov functions. This method of analysis does not require solutions of the system model and is applicable to nonlinear and time delayed systems, although the analysis can be very complex and not suitable for efficient engineering design.

Large signal linearization

This method is based on the fact that nonlinear systems exhibit different behavior for different amplitudes of input signal, although the input signals may have the same waveform. This is opposite to the behavior of linear systems. Thus, a kind of system linearization is achieved by adding a nonlinearity in series with the original system to make the behavior of such composition as independent as possible of the input signal amplitude. The result is an equivalent system having a linear or close to linear response for a range of amplitudes of the input signal. This situation is indicated symbolically in Fig.1-3. This method is of specific interest to us, as our approach uses it to great extent.

Our approach in this study will implement large signal linearization. For example, we can eliminate the main system nonlinearity by this method. This assumes that the main nonlinearity and its inverse are memoryless and monotone. It should be noted that since the linearization is only an approximation within the neighborhood of an operating point, by this approach only 'local' behavior of the nonlinear system within the vicinity of that point can be predicted (Du & Sun 1994). It cannot predict 'nonlocal' behavior far from the operating point and certainly not the 'global' system behavior (Floudas & Pardalos 1992, Horst & Tuy 1993, Horst *et al.* 1995, Schoen 1993). However, the

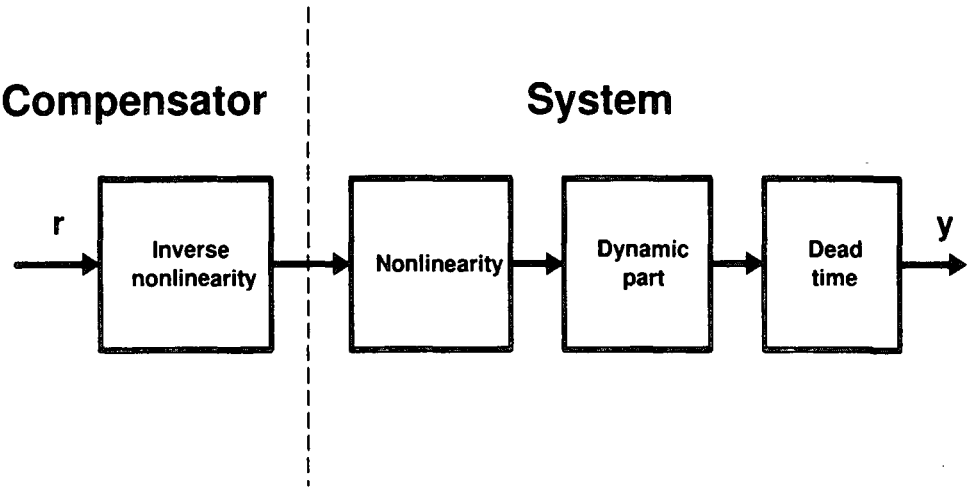

Figure 1-3: Figure illustrating linearization

particular implementation of the design method in the frequency domain will allow us to have very good behavior also far from the system's operating point.

1.3.4 Design for systems having dead time

Dead time appears as a phenomenon in all real world systems. There are some simplified analytical methods of resolving (although incomplete) the design of a compensator for a linear system that contains dead time. The design of the compensator becomes however far more complex when we consider also nonlinearities in the system. The only method that can deal with a problem of this type in a general case seems to be Lyapunov's direct method.

For illustration purposes we will consider a simple case of a linear time invariant system and the corresponding compensator design. Consider the system with dead time as shown in Fig.1-4.

Let the overall transfer function of an open loop (SISO) system with dead time without the compensator be:

$$\frac{Y(s)}{U(s)} = G_I(s) = G(s)e^{-\lambda s},$$

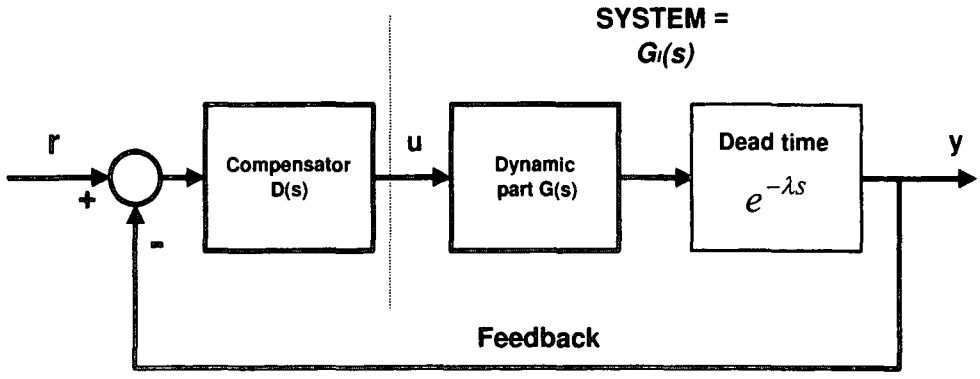

Figure 1-4: Illustrating design with dead time

where λ is the dead time. Let the linear controller with the transfer function $D(s)$ be inserted into the forward branch of the closed loop system. Then the closed loop transfer function of the system is given by

$$\frac{Y(s)}{R(s)} = \frac{D(s)G(s)e^{-\lambda s}}{1 + D(s)G(s)e^{-\lambda s}},$$

where $Y(s)$ is the Laplace transform of the output y of the system and the $R(s)$ is the Laplace transform of the reference input r to the system. Let us attempt to make a compensator with the transfer function $D_m(s)$, such that the desired transfer $H_d(s)$ function of the compensated closed loop system is

$$H_d(s) = \frac{Y(s)}{R(s)} = \frac{D_m(s)G(s)}{1 + D_m(s)G(s)}e^{-\lambda s}.$$

Hence, we get:

$$D_m(s) = \frac{H_d(s)}{G(s)\left[e^{-\lambda s} - H_d(s)\right]}.$$

Obviously, we cannot always determine $D_m(s)$ to satisfy the above expression due to the realizability problem. This indicates serious limitations of this analytical technique. If there are nonlinearities in the system, which is normally the case, then this technique

is not applicable.

1.4 Research Goals

We have thus far briefly mentioned several different techniques used to analyze the behavior of systems containing nonlinearities or dead time. The existing methods do not offer a complete solution to our problem, namely that of designing a compensator of nonlinear time delayed systems. Hence the reason for our research.

Our research will attempt to:

1. find a practical method of designing a compensator to optimize a nonlinear system with dead time;

2. analyze the results and investigate how accurate the optimization-based compensator design was;

3. investigate if the optimization in the frequency domain also provides acceptable results in the time domain.

Our research goals are oriented toward the development of an efficient technology based on optimization for the frequency design of compensators in nonlinear systems having dead time. This technology will result in a suitable software package to enable design for the most common structures of the system to be compensated. A solution will be achieved by merging DSP and constrained minimax optimization techniques (Du & Pardalos 1995).

The following research subproblems will be analyzed:

1. Development of techniques for optimization in the frequency domain of the selected system topologies;

2. Development of an algorithm for optimization of frequency characteristics based on DSP (Bringham 1974);

3. Development of software for automated off-line procedures of optimization based on the DSP technique.

We expect the complete system to yield good frequency and acceptable time domain behavior.

Chapter 2

Frequency Domain Design

2.1 Introduction to our method

The method that we propose for frequency design of compensators (Optimization Based Frequency Domain Design of Compensators - OBFDDC) essentially uses two systems. One is the reference system which has the desired behavior in the frequency domain. The other is the system that we want to make behave in the frequency domain as close as possible to the reference system. We will call this system the plant. For simplicity, the reference system is considered as a linear time invariant. It should be noted that the method can be adapted to cater for the case when the reference system is nonlinear time invariant. The system to be compensated due to its inadequate frequency domain characteristics is nonlinear system with dead time. We assume that both of these systems are with the fixed structures and that the system parameters cannot be changed. Therefore we will add a compensator in-line with the plant, close the loop, and tune the compensator until the closed loop system behaves as close as possible to the reference system (from the viewpoint of characterization in the frequency domain).

This study proposes a method (OBFDDC) for frequency tuning of compensators based on DSP and optimization techniques (Alkin 1994, Du & Pardalos 1995, More' & Wright 1992, Ingle & Proakis 1997). This method requires intensive computation, as

frequency characteristics are given for the range of frequencies. In the OBFDDC method it is assumed that:

- the frequency characteristics of the overall system with the compensator can be determined in simulation
- the desired frequency characteristics are specified;
- the system structure is known;
- the frequency characteristics of the plant, with the exception of the compensator, are known;
- nonlinearities in the system are known (for simulation purposes only).

The OBFDDC method will attempt to determine the optimal frequency characteristics of the compensator using different techniques for constrained optimization (cf. Marquardt 1963, Fletcher 1988, Du & Pardalos 1995, Dennis & Schnabel 1996, Gill 1982, Schoen 1993, Wright 1992), so that the overall system with the compensator achieves the frequency characteristics which differ minimally from the desired predefined ones.

In this process, in each phase, we start by selecting some initial values for the compensator's tunable parameters. This determines the initial frequency model of the compensator. The frequency characteristics of the overall system are calculated and compared to the desired ones. The optimizer will then search for a better set of tunable parameters to result in a better frequency model of the compensated system. In this way the optimizer changes the adjustable parameters in the compensator. The process is repeated until the discrepancy of the achieved frequency model of the compensated system is sufficiently close the predefined desired one. The proposed approach will be applied specifically to different models of process control systems having a long dead time and containing a nonlinear actuator. For such systems this method will allow the elimination of manual tuning of controllers in instances where a frequency domain design of the system is re-

quired and should generally provide a different basis for compensator (controller) design in the frequency domain.

2.2 Details and Practical Implementation

We will begin with the model of the plant. The general structure of the plant used in this study is given in Fig.2-1. It has three parts:

- the system nonlinearity,
- the linear dynamic part,
- the dead time.

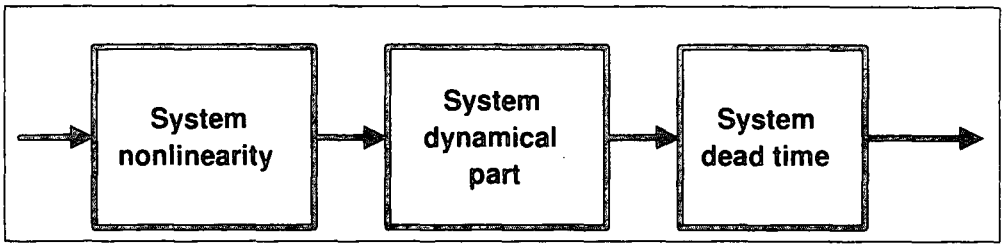

Figure 2-1: The plant to be compensated

This system, by assumption, does not have the frequency characteristics we would like, i.e. those of the reference system. So, we will attempt to optimize this system in the sense that we adjust its frequency characteristics. When we say optimize this means that this system should behave the same as our reference system. In order to achieve this we add a compensator at the input to the system and close the loop. This situation is represented in Fig.2-2. In order to know how closely our system behaves to the desired system we have to somehow compare the two. Roughly speaking, we do this by comparing the outputs of the two systems to the same input signal. The difference between the desired system output y_d and the plant output y is the error. We make

use of this error as one of the inputs to the optimizer. The compensator parameters are adjusted in such a way so as to produce a minimum error. When we compare the two output signals y_d and y in the time domain then we call this time domain error. When we transform signals into the frequency domain by the use of DFT (Discrete Fourier Transform) then we call this error the frequency domain error. In our research we make use of the frequency domain error and the whole design process relates to the frequency domain optimization.

The compensator shown in the Fig.2-2 consists of a linear part L_c and a nonlinear part N_c. The compensator design process has two phases for reasons given below. In order to make the compensator design more effective, we want to first minimize the effects of the main nonlinearity in the system. We do this by a process called large signal linearization. This process was referred to in the introduction. The linearization is achieved by placing an "inverse" nonlinearity N_c in front of the system nonlinearity N_s, as indicated in Fig.2-2. The two nonlinearities then tend to cancel themselves out. The inverse nonlinearity N_c is determined by experiments and thus it is not the exact inverse of the main system nonlinearity. Since in practice with this approach we cannot obtain the perfect inverse nonlinearity, the result of this linearization is a new system which is still nonlinear to an extent, but its nonlinear effects are considerably reduced. This linearization is done initially, in the first phase of the compensator design, to reduce main nonlinear effects.

The design of the linear part of the compensator L_c for the system in Fig.2-1 is achieved by creating the closed loop system shown in Fig.2-2.

2.2.1 The optimizer function

As described previously the optimization function in the design process is achieved by a complex mathematical algorithm. The algorithm is implemented in Matlab software as the `minmax` routine and is a part of the Optimization Toolbox of Matlab (Grace 1994). However, the programs that allow its use in the context of this study are developed separately and they are given in the appendix. We briefly mention here the functioning

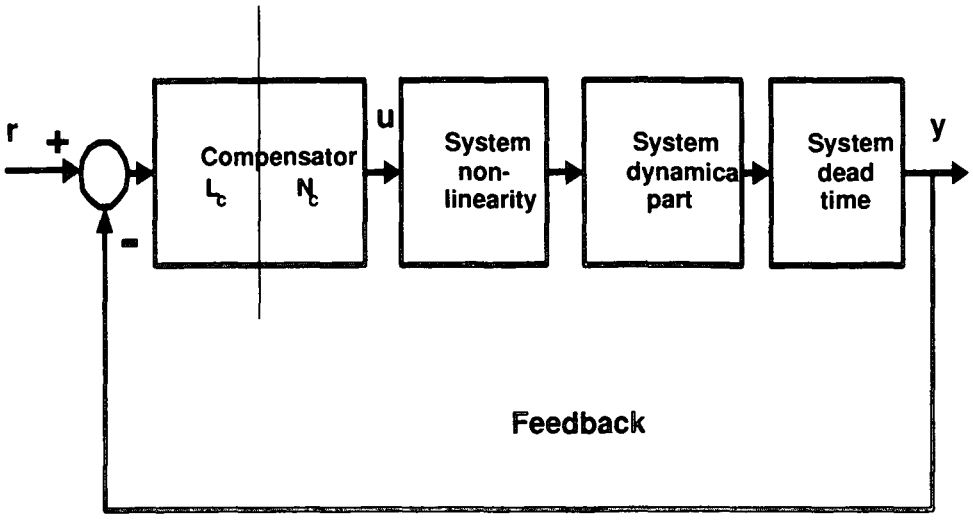

Figure 2-2: Controlled closed loop system.

of this algorithm with a rough mathematical description.

The `minimax` routine minimizes the worst case values for a set of multi-variable functions. As with all optimization routines it need the starting estimate of adjustable parameters. The current and estimated values of parameters are subject to constraints in order to limit the possibility of the system to generate an unstable solution - i.e. to result in a system that is unstable.

The minimax problem that is used in this study can be described mathematically as:

$$\texttt{minimize}\{\max F(x)\}$$

when

$$x \texttt{ is subject to constraint} : G(x) \leq 0$$

where $F(x)$, $G(x)$ and x may be complex vectors. The function G was used only to limit the allowable bounds for tunable parameters. The criterion function F is defined in

terms of a frequency domain presentation of system characteristics and their deviation from the desired characteristics.

Minimax uses a Sequential Quadratic Programming method (see Brayton *et al.* 1979, Corana 1987, Pardalos 1991) with some modifications in the original algorithms made to the line search and Hessian determination. This method is originally developed for the case when the function to be optimized is continuous; computer implementation of this technique inevitably deals with discrete values of the function to be optimized. As with all optimization techniques, this method may frequently lock into the local minimum. For this reason the optimization procedures proposed in what follows are recommended for off-line implementation only.

2.2.2 Linearization of the main nonlinearity in the frequency domain

The nonlinearity in the system prevents proper description of the system by a transfer function (cf. Coals 1956, Hassan 1992, Franklin *et al.* 1991). We can also say that the 'transfer function' of the nonlinear system changes with the input signal. We could design a compensator for the system, say for a chirp signal of amplitude 5, but the nonlinear system may behave very differently when presented with the same signal of amplitude, say, 10. In linear systems this does not happen. Linear systems retain the same type of behavior, only the amplitude of the output signal will change accordingly.

How is this linearization achieved?

The plant is linearized by means of large signal linearization technique. Using this method we insert a nonlinearity in the front of the system under compensation. This is actually the nonlinear part of the compensator and in the linearization process it has some parameters that can be adjusted so as to obtain an approximate inverse nonlinearity of the main system nonlinearity. Then, we tune the parameters of this nonlinear block until the system characteristics become almost invariant in the frequency domain for different

levels of the input signal. This whole process is done automatically with the aid of the optimizer routine. The optimizer is presented with the error calculated in the frequency domain. The error is made in such a way as to indicate, in a sense, the amount of nonlinearity still present in the system. The optimizer will attempt to minimize the error by adjusting the parameters of the adjustable part, i.e. the parameters of the nonlinear part N_c. These parameters are adjusted repetitively until the error is sufficiently minimized or when it cannot be reduced more.

In what follows we describe how to generate the input to the optimizer, i.e. the error function. There are many ways of how this error function can be suitably defined. We however use the technique explained below. We apply chirp signals of different amplitudes to the system. The input signals are with the same waveform, but differ in amplitude. The exact ratio of two different inputs is used and compared to the exact ratio of the respective outputs produced. For example, if the first input has amplitude 5 and the second input amplitude 10, then their ratio is 0.5. Let us assume that we present both of these input signals to the system, record their responses, convert these responses by means of DFT to the frequency domain, and then compare the ratio of the responses obtained. If the ratio of magnitudes of DFT of responses is different from 0.5 at any of the frequencies in the frequency domain, then it can be concluded that the system is nonlinear. The amount of nonlinearity can then be calculated as proportional to the deviation from the response of a linear system. This deviation will be the error used by the optimizer. The optimizer will then tend to adjust the inverse nonlinearity until this error becomes very low. This is the basic process used to estimate and reduce the nonlinearity in the system. Finally, the combined effect of both nonlinearities (N_c and N_s) in the forward branch should eliminate or sufficiently linearize the influence of the main nonlinearity N_s of the system.

The disadvantage of the method above is that the system will behave linearly only over the amplitude range of the input signals used in the process of determining N_c. A complete range of input amplitude variations would not be practical to consider due

to extremely costly computation. However for our purposes this method with limited amplitude ranges seems to be satisfactory.

2.2.3 Adjusting the linear part of the compensator

Once the system has been 'sufficiently' linearized by the method described above, the next step is to actually determine optimized parameter values of the linear part of the compensator. Here we make an assumption that if the frequency characteristics of the controlled system are the same as the frequency characteristics of the reference model, the behavior of the compensated system in the time domain will be similar to the reference model. This assumption will not be required if we are dealing with completely linear systems. However, the linearization that can be achieved is only 'partial' - the system still remains nonlinear, although to a much smaller extent. The only plant parameter that cannot be compensated for is the dead time. This dead time cannot be compensated for because it would mean that the compensator would have to act before it receives an input signal; this is physically impossible.

Fig.2-3 illustrates the system structure that can be used in the time domain optimization. The error is generated in the time domain between the system output y and the desired system output y_d. Notice that the desired system does not contain dead time. Now, if we compare the systems outputs in the time domain, then even if both systems are completely identical with the exception of the dead time, we will have some error. Since dead time can not be compensated for, the error will always be present. Thus, we conclude that this schema is a not very good one for the basis of design of compensators for systems that contain dead time. The correct schema for performing optimization in time domain is given in Fig.2-4. However, since the system that needs to be compensated is nonlinear it time domain characteristics need not necessarily reflect the features of the model satisfactory. Thus, it is necessary to look for an alternative way to generate the error. For this reason we resort to the error calculated from the frequency domain characteristics of the response signals.

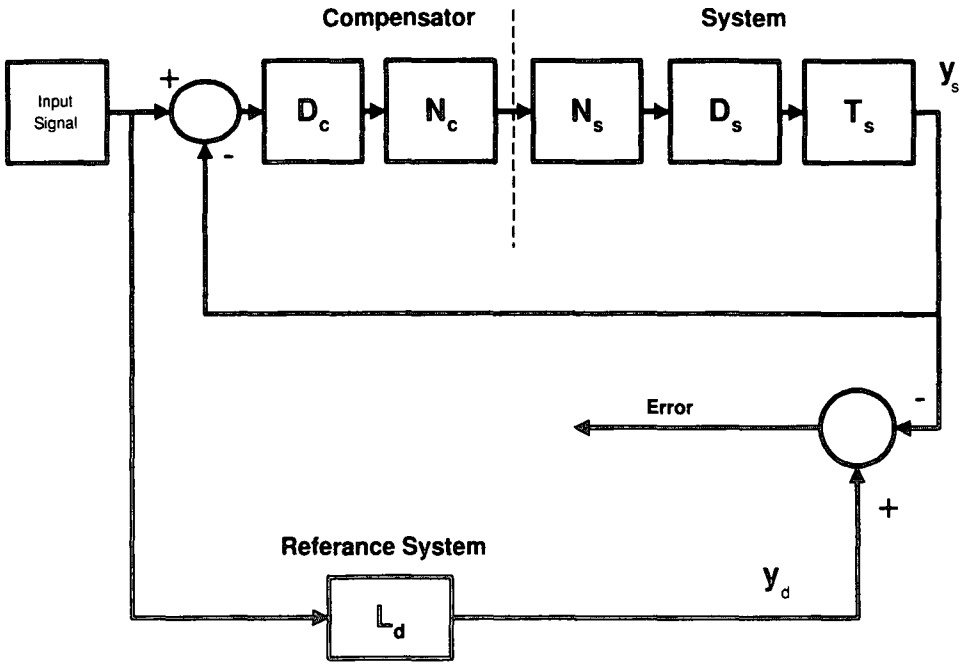

Figure 2-3: Schema for time domain compensator design

The output of the system under compensation and the output of the desired system are transformed into the frequency domain. This is done by means of DFT. Then the absolute values of the difference of the magnitude spectra of these two signals are presented to the optimizer as the error. It goes to say that if both magnitude spectra are identical, then the frequency domain of the system has been compensated for. Note at this point that we do not consider phase spectra characteristics. The input to the optimizer is the error obtained from magnitude spectra and the output are the parameters of the linear part of the compensator. The linear part of the compensator can be represented by a Laplace domain transfer function. This transfer function is chosen by the designer. Each coefficient in the polynomials of this transfer function can be adjusted until the frequency characteristics of the overall system match (or are as close as possible) to the reference model frequency characteristics. The process is repeated until the optimizer cannot further reduce the error. The input signals to the system are predefined to be

chirp signals of different amplitudes. The reason for using the chirp signal is that it has suitable frequency characteristics over a sufficiently wide range of frequencies (see Fig.3-4). Various amplitudes of the chirp signal will be used to determine how well the system is optimized over a range of varying amplitudes in order to compensate for the remaining uncompensated nonlinearities. So, in this phase we also use the large signal linearization technique.

The advantage of this approach is that the optimizer does not attempt to identify the model of the system. Due to this approach the actual structure of the system to which this design method can be applied, and that of the compensator, remain flexible. The compensator does not have to be limited to a certain structure i.e. to be a PID compensator, etc. The designer can select the structure he considers the most suitable. The OBFDDC method will adjust linear part and nonlinear part of the compensator in any way, so as to minimize the discrepancies in the frequency domain between the two systems.

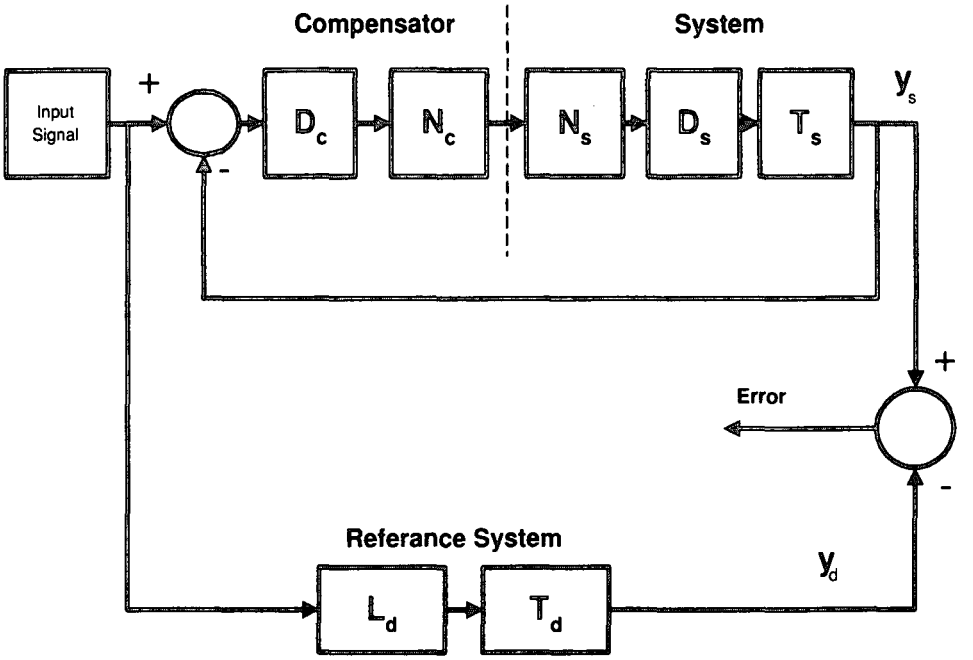

Figure 2-4: Correct schema for comparison of outputs in the time domain

The plant is also not limited to a particular structure (although we used a specific structure in this study) and the reason for this is that the OBFDDC method does not perform the identification of the system, but only deals with the global system input-output behavior, i.e. the system response. This approach therefore is applicable to systems of arbitrary complexity that may also be unknown to some extent, provided they are SISO (single input single output) systems and time invariant. It is clear that the compensator to be used may remain as flexible as the designer requires. This allows for parameters such as dead time, nonlinearities, discrete type filters, linear filters, etc. to be used as parts of the compensator.

The optimization procedure described above also has some shortcomings. One problem is that due to the nature of the empirical compensation, the results are not as accurate as with pure mathematical analysis and synthesis. These inaccuracies exist due to factors such as alliasing, leakage factors of the DFT, rounding error, and the most important, the ability of the optimizer to minimize the error globally. The main disadvantage with respect to error minimization is that there is a possibility that the optimizer may achieve a local minimum value instead of a global minimum and the system thus does not become fully optimized (for local and global optimization refer to Schoen 1993, Pardalos 1987, Horst & Pardalos 1995). This problem may be partially solved by restarting the optimization process with some new initial parameter values and in many cases a better minimum may be achieved, but there is no guarantee about the global minimum. With respect to the compensator, the proposed optimizer is not suitable for on-line implementation due to the fact that in its attempt to optimize the system it may produce inadequate values for the compensator parameters. This may not be allowed in a real life situation because this can cause the output to become saturated or the system to become unstable.

In the next two chapters we describe two sets of computational experiments that we used to test the validity of the proposed method for compensator design in the frequency domain.

Chapter 3

Main Linearization

3.1 Objective of experiments

The computational experiments that follow are aimed at demonstrating that the methodology proposed in the previous section of creating an inverse (or close to the inverse) nonlinearity in the compensator block may make the overall system sufficiently linear. These experiments focus on providing linearization in the frequency domain by means of a large signal linearization approach. This linearization is a preliminary step in designing the compensator. This first phase is necessary in order to ensure that the frequency design method applied later produces the correct results.

3.2 Method of linearization

Here we will illustrate through several examples the large signal linearization and its implementation in the frequency domain. The system for which the linearization is to be implemented is described in the previous chapters and is depicted in a part of Fig.3-1. We select nine different rational transfer functions that represent the dynamic part of the system without the dead time. These are given in Table 3.1. The main nonlinearity is shown in Fig.3-2 and the dead time is given as 0.4 seconds.

Fig.3-1 represents the system model for linearization of the main nonlinearity present in the system. The compensator contains in this phase only the nonlinear part. The linear part of the compensator will be taken into account later once the system is sufficiently linearized.

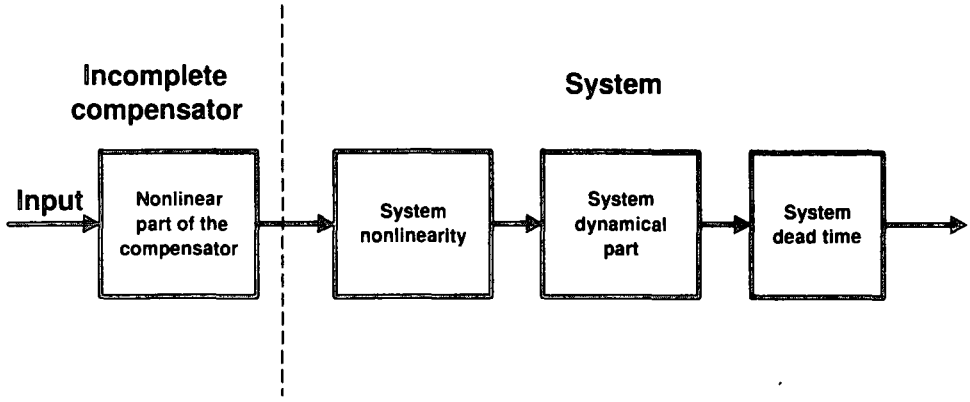

Figure 3-1: Portion of the system to be used in linearization

3.2.1 Dynamic part of the plant

The dynamic part of the system without the dead time is represented as a transfer function of the continuous system. This part does not necessary have to correspond to the continuous system; a discrete z-transform could have also been used. Table 3-1 gives these transfer functions used in the computational experiment:

3.2.2 Main nonlinearity

The selected nonlinearity does not have any dynamic elements. For example, for most control systems this nonlinearity would typically characterize an actuator. Fig.3-2 shows the selected nonlinearity used in our computational experiments.

The system nonlinear part is modeled (implemented) by means of the Matlab look-up-table function. The look-up-table block defines a piecewise linear input/output non-

Model No.	Transfer functions
1	$\frac{1}{2s+1}$
2	$\frac{1}{2s+2}$
3	$\frac{1}{s+2}$
4	$\frac{1}{s+3}$
5	$\frac{1}{s^2+s+1}$
6	$\frac{1}{s^2+3s+1}$
7	$\frac{1}{s^2+2s+2}$
8	$\frac{1}{s^3+3s^2+4s+2}$
9	$\frac{1}{s^4+4s^3+6s^2+4s+1}$

Table 3.1: Rational transfer functions of dynamic part of the system

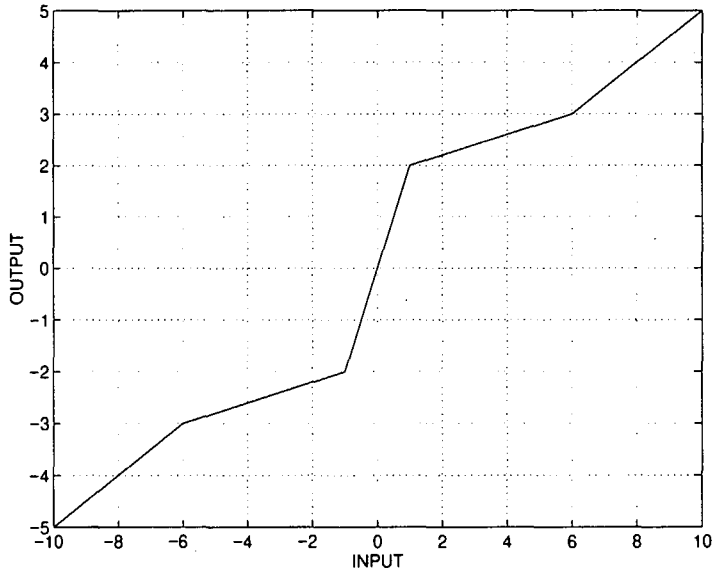

Figure 3-2: Nonlinearity of the system

linearity, which is evaluated at each time step. Outputs are interpolated on the basis of the look-up-table. Inputs falling outside the range are extrapolated (for further information about the look-up-table refer to the MATLAB manuals (MathWorks Matlab 1997a, 1997b). The final result of linearization should produce a nonlinearity of the compensator that sufficiently linearizes the nonlinearity in the system. This however is subjected to the range of the variations of the input signal amplitude, because the system may behave linearly only over the simulated range of input amplitudes. At the end of the linearization the two nonlinearities are expected to be very close to the "inverse" of each other. The linearization process would then ideally create two nonlinearities that "cancel" each other out. It is expected that the nonlinear part of the compensator will sufficiently linearize the system.

3.2.3 System dead time

The dead time of the system was taken to be $\lambda = 0.4$ seconds. This dead time was chosen arbitrarily. One of the requirements was that the dead time be sufficiently long so that it cannot be ignored in the design process.

3.2.4 The input signals used

The input signals used in these experiments are chirp signals. Chirp signals have the property of linearly increasing frequency with time. They are often used in the frequency analysis of nonlinear systems. The following expression describes a chirp signal:

$$y(t) = \sin(kt^2),$$

where $y(t)$ is the signal value at the moment t (given in seconds) and k is a constant. These signals have the continuous change of amplitudes over a well defined range and also they have a suitable frequency domain content, i.e. their spectrum is 'everywhere dense' within the range of frequencies considered. In our case it will be the range $[0, f_{\max}]$ and

f_{max} can easily be controlled in MATLAB. Fig.3-3 and Fig.3-4 illustrate the time domain waveform and the frequency magnitude spectrum of a chirp signal. We notice that the spectrum of the chirp signal covers a wide range of frequencies, therefore making it a good signal source for frequency analysis and optimization. Various maximum amplitudes of the chirp signals were used, as these variations are needed for the generation of the different frequency characteristics. Three values of amplitudes were used, namely 0.5, 2 and 5.

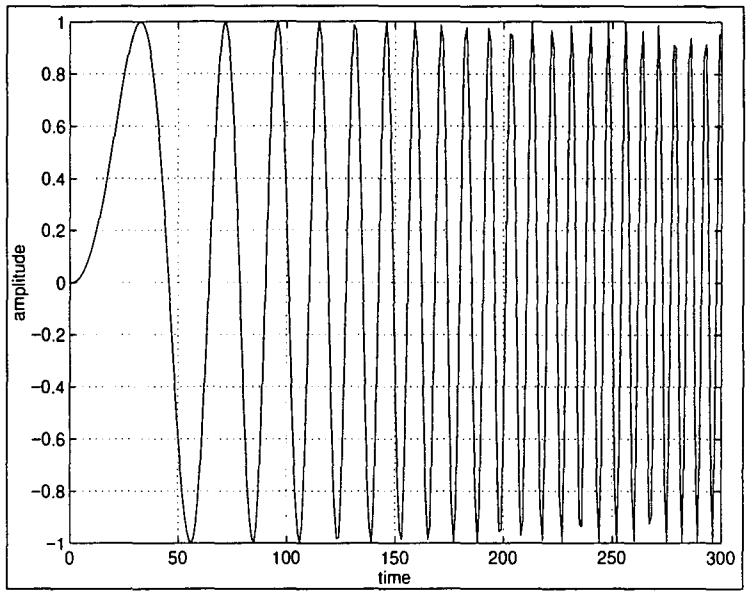

Figure 3-3: Time domain characteristics of the chirp signal

3.2.5 Generation of the error for the optimizer

The prime objective of linearization is to create a linear system. In a linear system various amplitudes of input signals will produce the same normalized output values (i.e. the transfer function of a linear system remains invariant). When the transfer function of a system varies with amplitude variations of the input, the system is considered to be nonlinear. This is the background of the large signal linearization method. Our

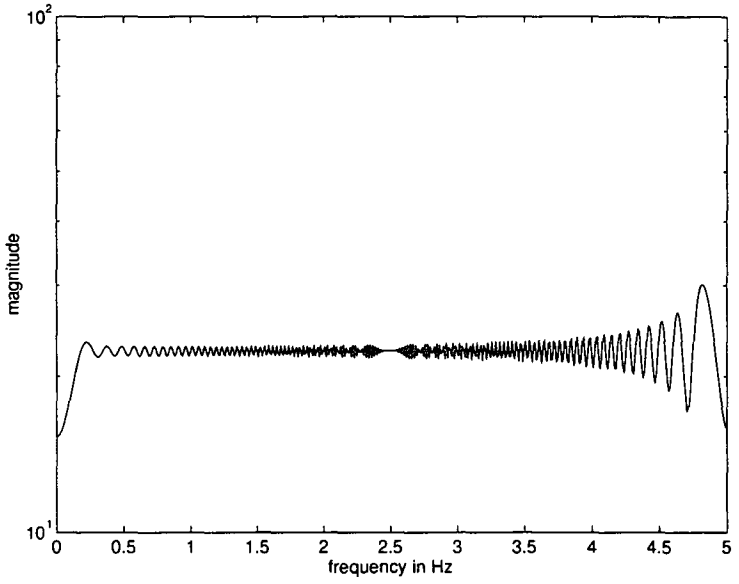

Figure 3-4: Frequency characteristics of the chirp signal

linearization procedure will reduce the effect of the nonlinearity by reducing the transfer function variations to the amplitude variations of the input signal. This method will reduce the overall nonlinearity in the system.

The compensator's nonlinear part N_c is placed just before the system nonlinearity as shown in Fig.3-1. If we obtain compensator nonlinearity close to the "inverse" of system nonlinearity, then a significant reduction of the overall nonlinearity in the system is achieved. N_c is implemented as a piecewise linear function with 5 braking points that serve as the adjustable parameters in the process of optimization.

The diagram in Fig.3-5 illustrates the structure of the system used for achieving linearization. In this figure we attempt to make three transfer functions T_1^*, T_2^* and T_3^* invariant with regard to the input signals used, or in other words, to make them the same for all three input signals. The input signals are chirp signals with the same waveform, but with different amplitudes. The discretization of the signal was achieved with the aid of a zero hold interpolator (analog to digital conversion) using a sampling period of 0.1 seconds. For DFT calculation the Matlab's function '`fft`' was used. The leakage factor

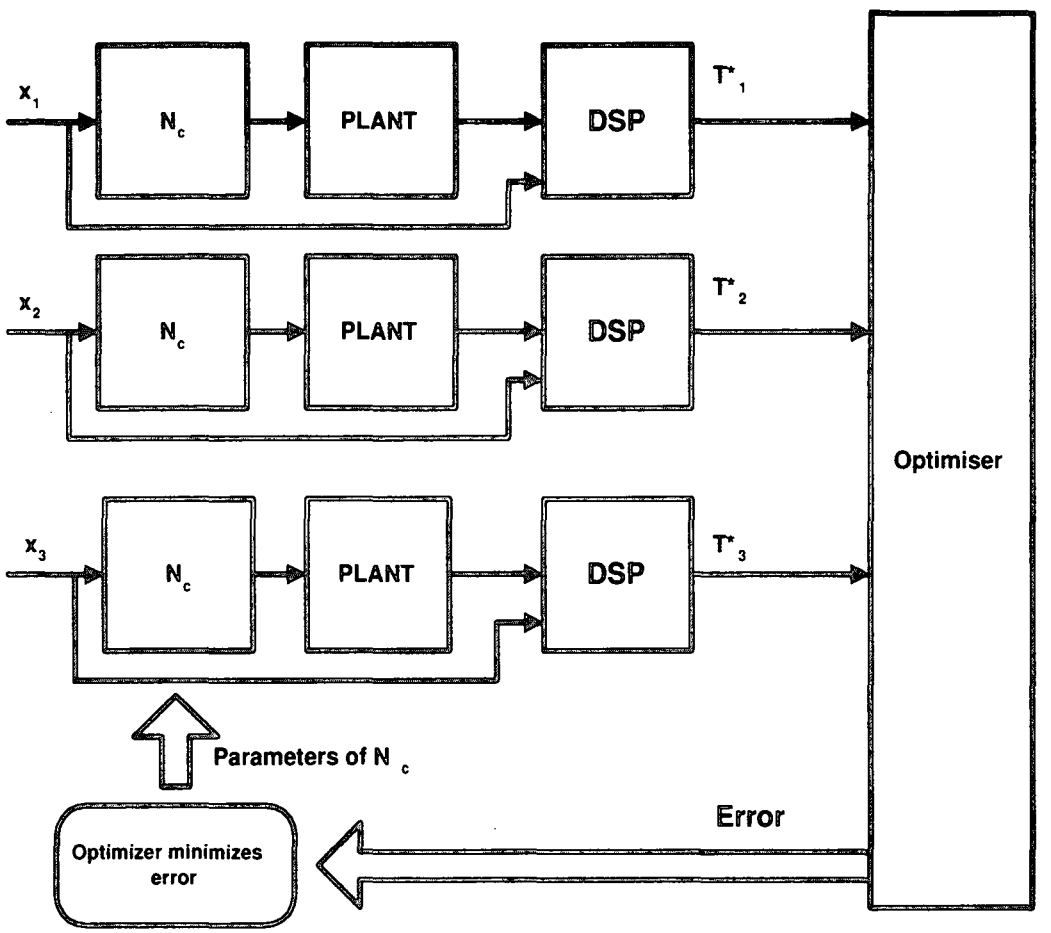

Figure 3-5: Schema for linearization experiment.

of the DFT appeared to be small in this application and hence was ignored. The total system simulation time was 100 seconds with a sampling rate of 0.1 seconds i.e. 1000 samples were collected in each window and analyzed. The alliasing effect was eliminated by bounding the input frequencies of the chirp signal. The highest frequency generated was less than the maximum Nyquist frequency for that sampling rate (Nyquist 1932). In our case the maximum frequency that could be measured for the sampling rate of 0.1 seconds is 5Hz according to the Nyquist criterion.

The system in Fig.3-5 consists of three parallel branches that produce the relevant transfer functions. The inputs to these branches are denoted by x_1, x_2, x_3 and the plant outputs are y_1, y_2, y_3. The respective transfer functions for each individual input and output pair, T_1, T_2 and T_3 are:

$$T_i(jf) = \frac{Y_i(jf)}{X_i(jf)}, \quad i = 1, 2, 3,$$

where Y_i and X_i are the Fourier transforms of the signals y_i and x_i, respectively. In the calculation we used the DFT as the appropriate tool due to the discrete nature of signals in computer simulation.

If the system is completely linear then T_1, T_2, and T_3 should all be equal (Shearer *et al.* 1971, Macfarlane 1970, Wellstead 1979). However since nonlinearity does exist, there will be a discrepancy between the T_1, T_2, T_3. If we could adjust our compensator in such a way so as to reduce the discrepancies in the calculated transfer functions, then it could be assumed that the system is more linear over the region of the used input amplitudes. Therefore we use this discrepancy between transfer functions as our error for the optimizer.

$$error = [err(f_0),\ err(f_1),\ \ldots, err(f_n)]^T$$

where

$$err(f) = | T_1(f) - T_2(f) | + | T_2(f) - T_3(f) | + | T_3(f) - T_1(f) |$$

Note that T_1, T_2, T_3 are complex functions of f, while *error* is a real vector. Values of $f_0, ..., f_n$ are equidistant values of frequency f that cover the linearization range of frequencies.

3.3 Analysis of the linearization results

After linearization, the system was simulated with the nonlinearity N_c obtained, as well as without it, using chirp input signals of various amplitudes. The responses obtained were further analyzed to indicate the amount of nonlinearity that remained in the system. The results are finally presented graphically and in a table.

The linearized and nonlinearized systems were analyzed in the time domain. Three chirp signals of different maximal amplitudes were passed through the system. The responses of the system to these input signals were recorded and normalized. The error function vector was derived using the following equation:

$$err^0(t) = [| \frac{1}{g_1} y_1(t) - \frac{1}{g_2} y_2(t) | + | \frac{1}{g_1} y_1(t) - \frac{1}{g_3} y_3(t) | + | \frac{1}{g_2} y_2(t) - \frac{1}{g_3} y_3(t) |]$$

Here, g_1, g_2, g_3 are the scaling factors of the system under test. They had values 0.5, 2 and 5 (the same as the maximal amplitude of the input signal) to compensate for the difference in the maximal amplitude in the chirp signals used as inputs. Signals y_1, y_2, y_3, are the responses of the systems to the associated chirp input. The err^0 is the error in time domain shown in the diagrams in Fig 3.6 to 3.14. The error is shown for the systems before linearization and after linearization. For the programs that generated these results refer to Appendix A.

These results are also presented by means of the integral of the absolute errors for

easier numerical comparison. The following formula was used to calculate this integral error:

$$err_I = \sum_{t=0}^{t=t_{final}} \mid err_O(t) - err_L(t) \mid,$$

where err_I is the integral absolute error, err_O is the normalized error in the time domain of the original system and err_L is the normalized error in time domain of the linearized system. Moments $t = 0$ and $t = t_{final}$ denote the start and the end of the simulation period. In the calculation of the integral absolute error the step of integration is assumed to be 1.

3.3.1 Graphical results

Figures 3.6 - 3.14 illustrate the compensation between the linearized and nonlinearized systems using a chirp signal input. The thick curve indicates the total instantaneous error err^0 between the linearized system responses, and the thin line indicates the total instantaneous error err^0 between the original nonlinearized system responses when the previously mentioned chirp signals were used to excite the systems. Obviously, for the ideal situation we would have the $err^0(t) = 0$ for the linearized system; therefore the respective graph would also tend to zero.

The amount of nonlinearity present in the system is indicated Fig 3.6 - Fig 3.14. The extent of the existing system nonlinearity is reflected by amplitude of the signal: the larger the values, the larger is the nonlinearity in the system. Viewing these graphs we note that there was a significant improvement in the linearization of the system.

3.3.2 Numerical results

Table 3-2 illustrates the total absolute error before and after the linearization process

It is also very clear from the numerical values of the error err_I that there was a significant reduction in the systems nonlinearity before and after linearization. The

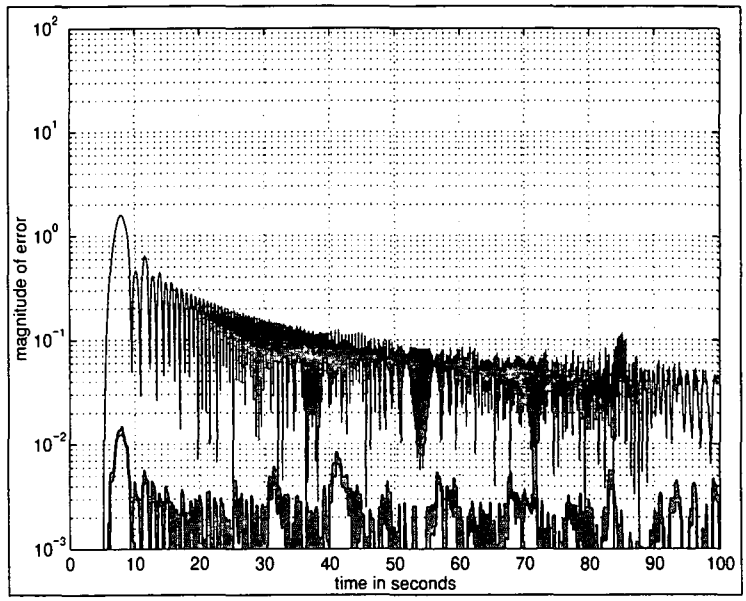

Figure 3-6: Linearization results for model No: 1.

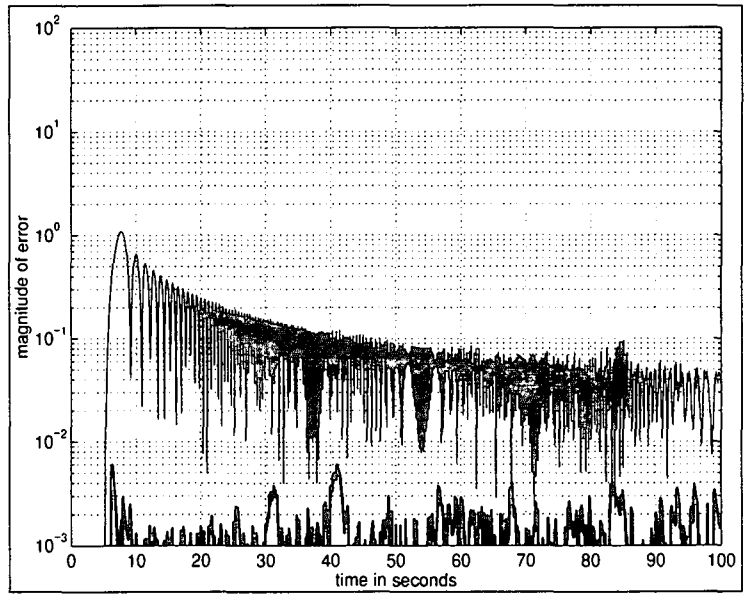

Figure 3-7: Linearization results for model No: 2.

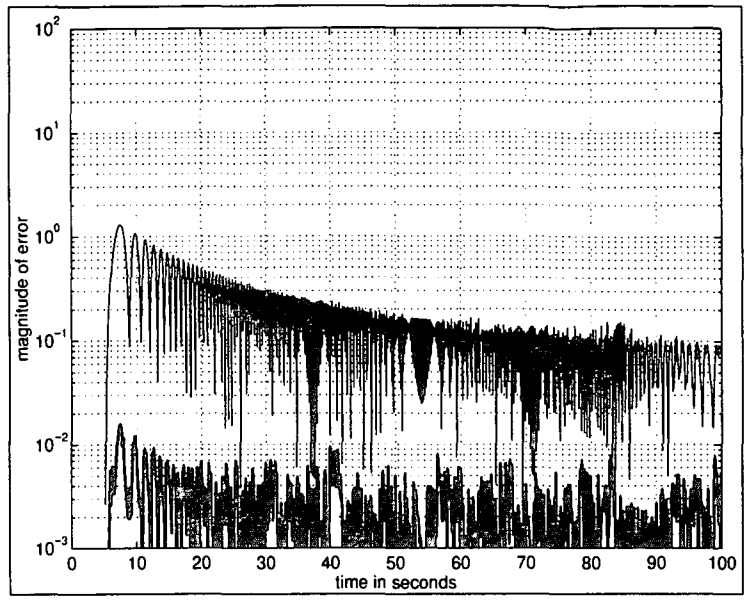

Figure 3-8: Linearization results for model No: 3.

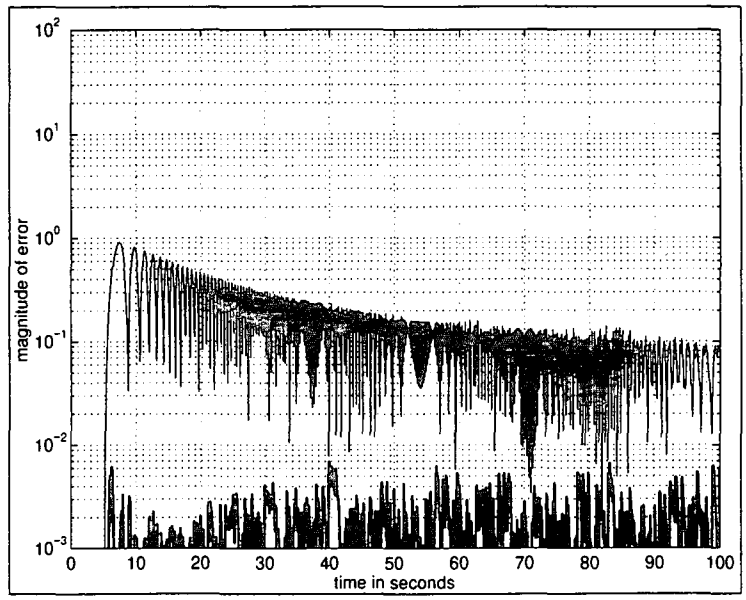

Figure 3-9: Linearization results for model No: 4.

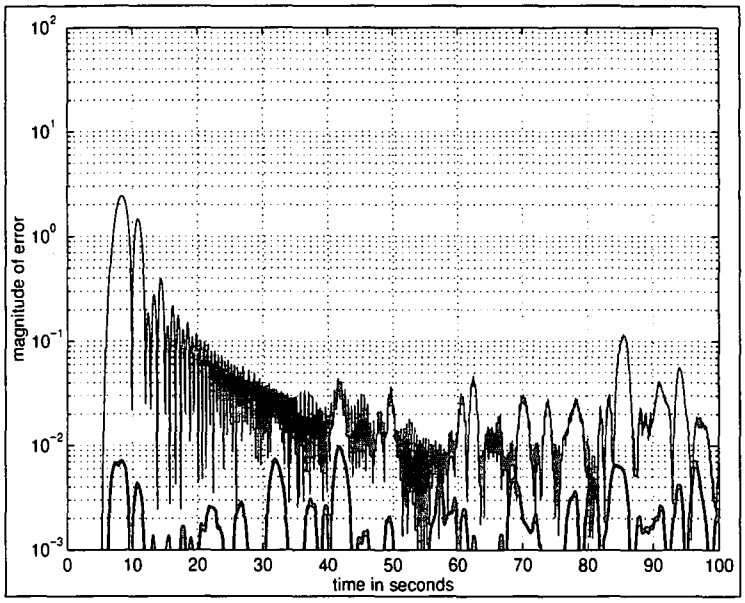

Figure 3-10: Linearization results for model No: 5.

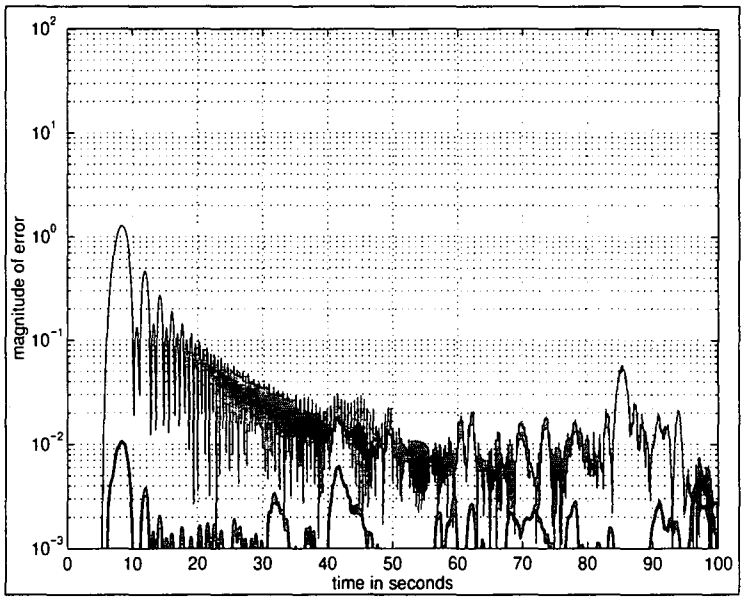

Figure 3-11: Linearization results for model No: 6.

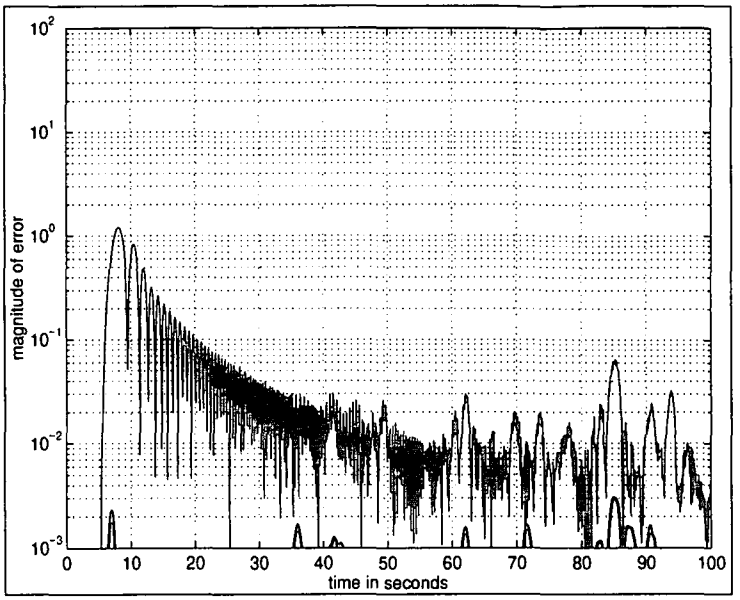

Figure 3-12: Linearization results for model No: 7.

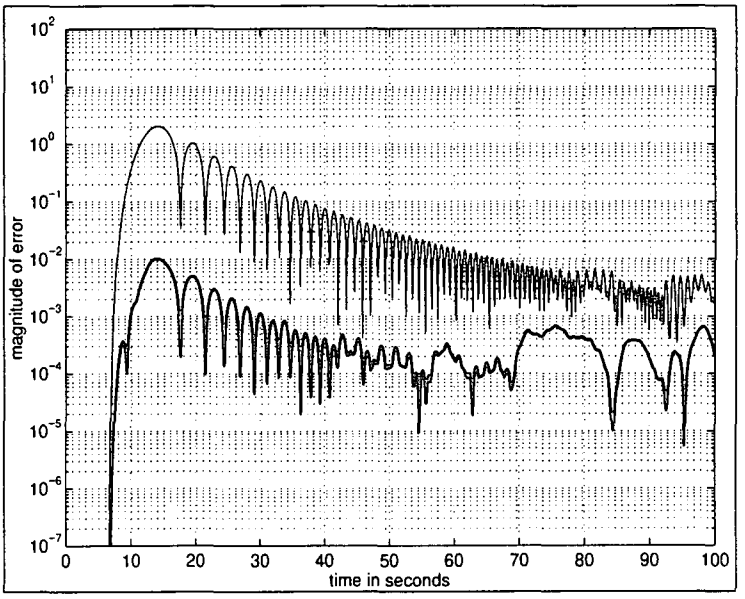

Figure 3-13: Linearization results for model No: 8.

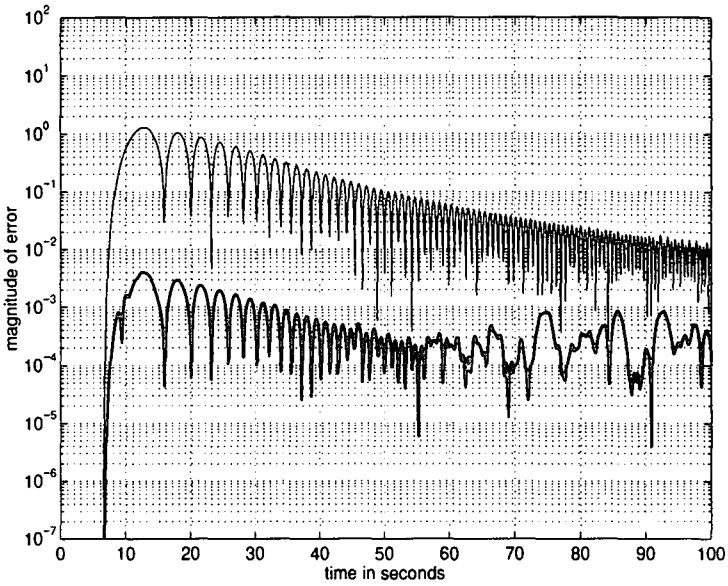

Figure 3-14: Linearization results for model No: 9.

Model No	err_I after linearization	err_I before linearization
1	1.99	114.60
2	1.16	102.30
3	2.53	176.17
4	1.49	157.40
5	1.91	101.00
6	1.41	56.40
7	0.48	61.86
8	0.58	172.80
9	0.90	167.60

Table 3.2: Integral absolute error of linearized and nonlinearized systems

reduction is estimated in excess of 99%. However, we have to note that there is still a small amount of nonlinearity present.

3.4 Conclusions

The nonlinear portion of the compensator has linearized the nonlinearity in the system by placing another nonlinearity at the input of the system. This method of linearization is effectively a large signal linearization. We have seen very good results concerning this method. However one can not overlook the fact that there is still a small amount of nonlinearity present, even after linearization. This small amount of nonlinearity could be due to the empirical method used and insufficient amplitude variations used to linearize the system.

The limitation of this method is that the nonlinearity in the system has to be memoryless and placed before the dynamical part of the system. Also, we noted that if the nonlinearity is not at the plant input, then this method is less effective. This can be explained by the fact that the dynamic part of the plant distorts a lot the signal that enters the nonlinear block, which is not the case if the nonlinearity is at the plant input. However, many process control systems have the main nonlinearity exactly at the plant input and thus our method seems to be appropriate for such cases.

Chapter 4

Final Design of Compensator

4.1 Objectives of frequency domain optimization

By using frequency domain design of the linear part of the compensator we attempt to achieve that the overall linearized system, i.e. the closed loop system with the compensator, has frequency characteristics as close as possible to that of the reference system. This design again will be based on optimization made in the frequency domain. The optimization will be achieved by comparing the response in the frequency domain of the reference system to the closed loop linearized system. The complete experimental results that follow should illustrate how well this design is made.

4.2 Method of frequency domain optimization

The systems that were linearized by the technique used in the proceeding chapter will behave as mildly nonlinear systems. Now we have to design the linear (dynamic) part of the compensator. The linear part of the compensator is placed in front of the nonlinear part of the compensator as indicated in Fig 2.3. This part of the compensator may be suitably represented by the transfer function of continuous systems. We will tune the linear part of the compensator by adjusting the coefficients in the numerator and the

denominator of the transfer function of this block.

The error is generated by comparing the output of the system under optimization to the output of the reference system. These outputs are transformed to the frequency domain with the aid of the DFT. We consider only the magnitude spectra of the outputs in this design. The difference between the two magnitude spectra (those of the output signals y_d of the reference system and y of the system under design) are made and this determines the error that we intend to minimize. Since the resulting closed loop system still only approximates a linear system even after linearization, we will use the chirp signal with different amplitudes for the generation of the error. We then calculate various errors generated by signals of different amplitudes to obtain the total error. This total error is the error that is actually presented to the input of the optimizer.

Fig. 4.1 illustrates the block diagram of the system for the generation of the error signal. The signal which is in the time domain must be transformed to the frequency domain. Modeling the closed loop system within the frequency domain is achieved by transforming the output of the reference system and the output of the system that is being compensated, into the frequency domain. The difference of the two magnitude spectra are obtained and its absolute values becomes determine the error used by the optimizer. Fig. 4.1 illustrates this process.

The difference of the relevant DFTs generates the error which the optimizer attempts to minimize. Note that only the linear part of the compensator is tuned (Fig. 1.1). The error to the optimizer is characterized by the following equation:

$$\min[J(p,f)]$$

where

$$J(p,f) = \sum_{i=1}^{k} abs[\mid Y_i^n(jf) \mid - \mid Y_d(jf) \mid]$$

where Y_i^n is the DFT of the measured and normalized system output for the i-th input and Y_d is the DFT of the output obtained for the normalized input of the reference

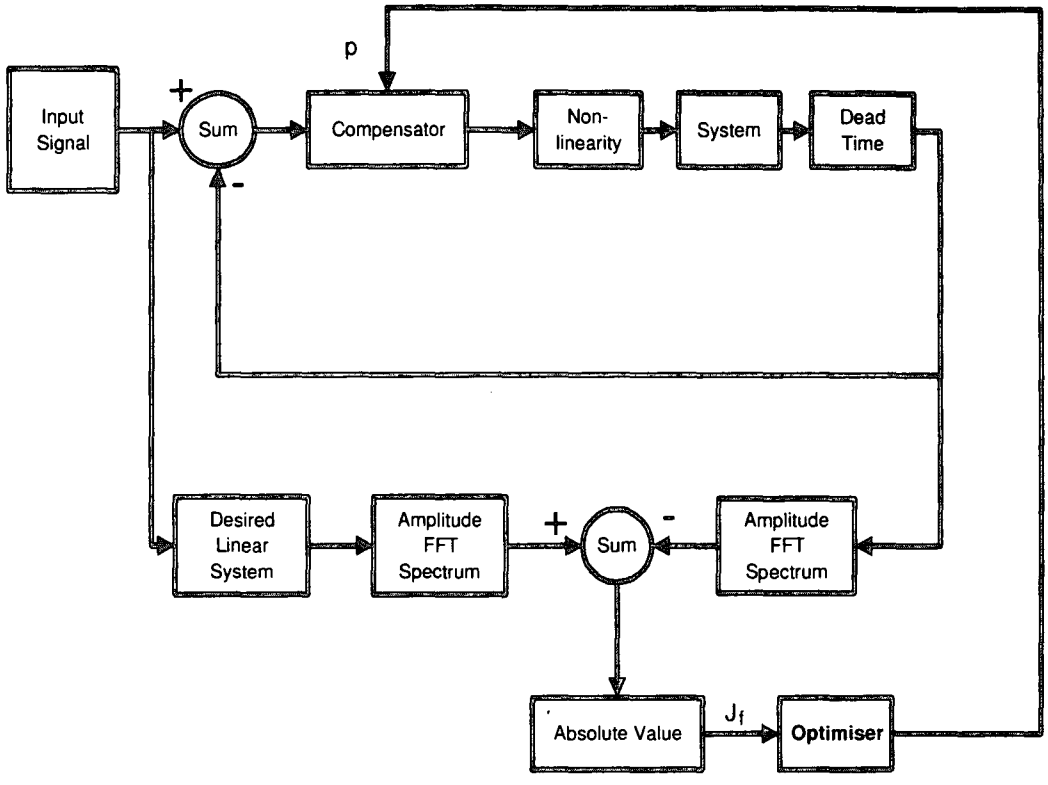

Figure 4-1: Model used for optimizing the nonlinear system in the frequency domain

Model No.	System transfer functions
1	$\frac{1}{2s+1}e^{-0.4s}$
2	$\frac{1}{2s+2}e^{-0.4s}$
3	$\frac{1}{s+2}e^{-0.4s}$
4	$\frac{1}{s+3}e^{-0.4s}$
5	$\frac{1}{s^2+s+1}e^{-0.4s}$
6	$\frac{1}{s^2+3s+1}e^{-0.4s}$
7	$\frac{1}{s^2+2s+2}e^{-0.4s}$
8	$\frac{1}{s^3+3s^2+4s+2}e^{-0.4s}$
9	$\frac{1}{s^4+4s^3+6s^2+4s+1}e^{-0.4s}$

Table 4.1: System transfer functions with dead time

system. $J(p, f)$ is the total error generated in the frequency domain (at the frequency f) for all used input signals; p is the vector of adjustable parameters and k is the number of input signals with different amplitudes used in experiments. In our case $k = 3$.

The linear part of the compensator is free to be set before the optimization (tuning) begins. So, initially we specify a transfer function of this part and give the initial values of the coefficients of this transfer function. The optimization process tends to adjust the numerical values of the parameters until minimization of the error is sufficient for all frequencies f involved. Then the parameters are at their final value and the optimization process is complete.

4.2.1 Systems under optimization

The linear parts of the systems to be optimized are given in Table 4-1. They are same as those used in the section about linearization.

4.2.2 The reference system

The reference model (see Fig.1.1) in this optimization can have any transfer function. In our experiments it was chosen that this system has no dead time and its model transfer function is selected as

$$\frac{2s+1}{s^4+8s^3+4s^2+4s+1}.$$

However, we should emphasize that with small alteration the method used in this study can use an arbitrary system as the reference one.

4.2.3 The linear part of the compensator

The the linear part of the compensator is connected in series to a tuned nonlinear block (Fig. 1.1). The nonlinear part was optimized in the linearization process described earlier and so remains fixed. We select an arbitrary structure for the compensator's linear part. Any structure with tunable elements can be used for this method, although the final results depend also on the structure selected. The compensators dynamic part was taken as a third order transfer function

$$C_d(s) = \frac{p_1 s + p_2}{s^3 + p_3 s^2 + p_4 s + p_5}.$$

Here p_1, p_2, p_3, p_4, p_5, are the parameters that the optimizer will attempt to adjust during the design process. These parameters were initially set to 1 each. During the optimization process the optimizer will change the individual parameters so as to minimize the error over the range of frequencies.

4.3 Implementation of the frequency domain design

Firstly we convert both output signals of the tuned and untuned system from the time domain to the frequency domain by means of DFT. The input signal used in the experiment was a chirp signal. The discretization of the signal was achieved with the aid

of a zero hold interpolator (analog to digital conversion) using a sampling period of 0.1 seconds. For DFT calculation the Matlab's function '`fft`' was used. The leakage factor of the DFT appeared to be small in this application and hence was ignored. The total system simulation time was 100 seconds with a sampling rate of 0.1 seconds, i.e. 1000 samples were collected in each window and analyzed. The alliasing effect was eliminated by bounding the input frequencies of the chirp signal. The highest frequency generated was less than the maximum Nyquist frequency for that sampling rate. In our case the maximum frequency that could be measured for the sampling rate of 0.1 sec is 5 Hz according to the Nyquist criterion.

The next step was to calculate the error to be used by the optimizer. The error was defined in such a way to show the level of discrepancy that took place between behaviors of the system under tuning and the desired system. It was defined in a subsection previously.

After the design of the linear part of the compensator is finalized, the results obtained were analyzed in the following way. Fig. 4.2 illustrates the conceptual error generation schema for testing the quality of the frequency domain design. Here Y_d is the magnitude of the DFT of the desired system output, Y is the magnitude of the DFT of the output of the untuned system in the frequency domain, Y^t is the magnitude of the DFT of the output of the tuned system; err_u & err_t are the error vectors for the untuned and the tuned system, respectively. The same three input chirp signals u_1, u_2 and u_3 as in the linearization experiments were used.

The equation determining the cumulative error of the untuned system in the frequency domain for a particular frequency f is given by

$$
\begin{aligned}
& err_u(f) \\
= \; & [\mid Y_1^n(jf) - Y_d^n(jf) \mid + \mid Y_2^n(jf) - Y_d^n(jf) \mid + \mid Y_3^n(jf) - Y_d^n(jf) \mid],
\end{aligned}
$$

where

- err_u is the cumulative error (obtained for all three input signals) of an untuned system
- Y_1^n, Y_2^n, Y_3^n are the DFTs of the measured and normalized outputs of the untuned system for inputs u_1, u_2, u_3
- Y_d^n is the DFT of the measured and normalized outputs for any of the inputs u_1, u_2, u_3.

The equation describing the cumulative error of the tuned system in the frequency domain for a particular frequency f is

$$\begin{aligned} & err_t(f) \\ = \; & [\mid Y_1^{tn}(jf) - Y_d^n(jf) \mid + \mid Y_2^{tn}(jf) - Y_d^n(jf) \mid + \mid Y_3^{tn}(jf) - Y_d^n(jf) \mid], \end{aligned}$$

where

- err_t is the final cumulative error (obtained for all three input signals) of a tuned system
- $Y_1^{tn}, Y_2^{tn}, Y_3^{tn}$ is the DFT of the measured and normalized outputs of the tuned system for inputs u_1, u_2, u_3.

If an ideally compensated system is obtained by this method the system response would be identical to that of the reference model. The error in the frequency domain between them will therefore be zero for every frequency in the range considered. The final results obtained, expressed in terms of err_u and err_t, are shown for each of the 9 tested systems in Figs. 4.3 to Fig. 4.11. The compensation has been very good for almost all frequencies in the utilized frequency range. Notice also that generally better compensation has been achieved for higher frequencies.

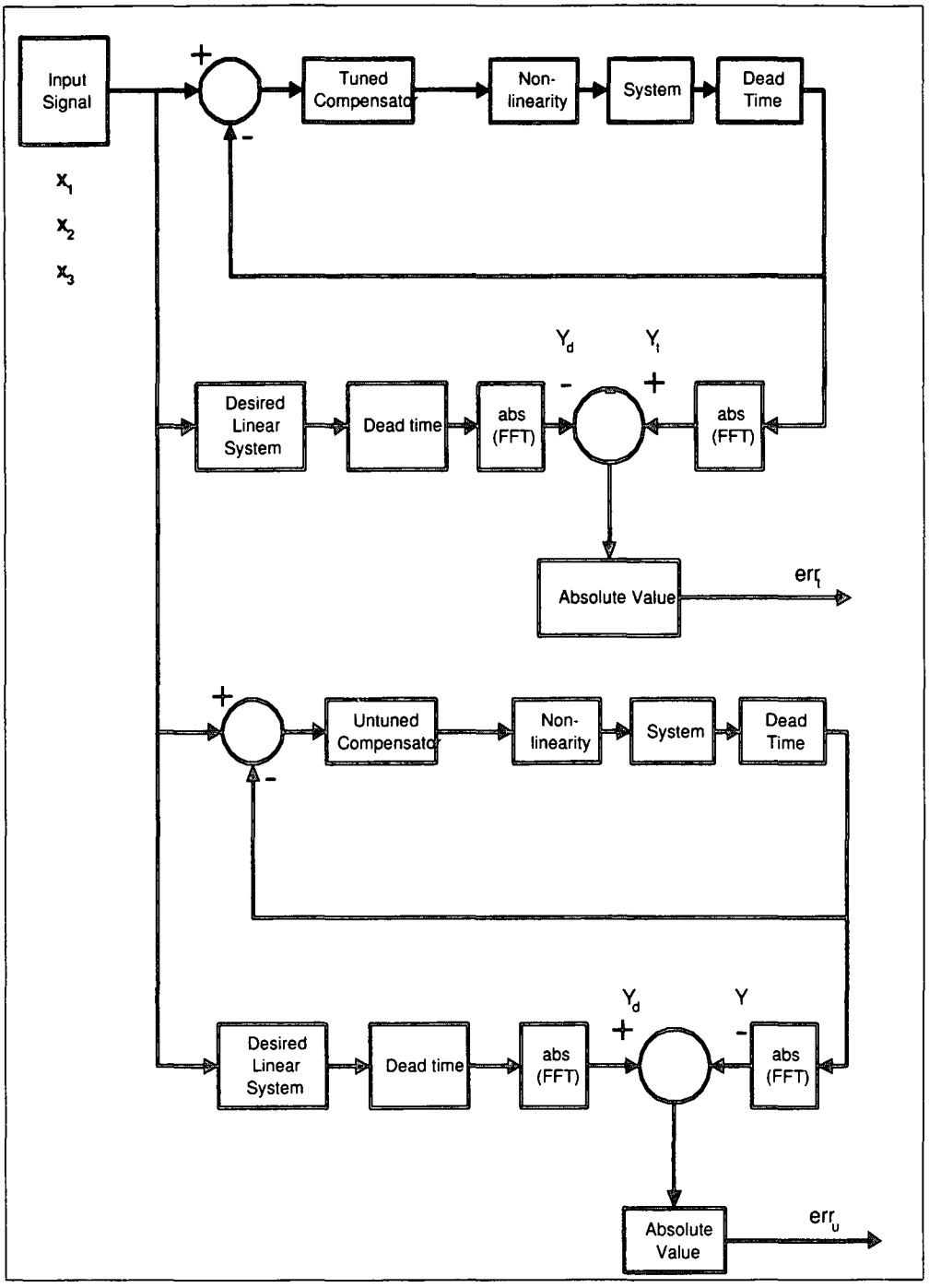

Figure 4-2: Figure illustration the generation of the error in the frequency domain.

To obtain a numerical representation of the total error achieved we calculate the sum of absolute values of errors at each of the relevant frequencies and in this way are able to observe the total amount of compensation actually achieved. These errors are calculated to provide the overall comparison of the error obtained between the tuned and untuned systems.

The equation describing the cumulative total error for the untuned system is given as:

$$\begin{aligned} err_{untuned} &= \sum_{f_i=f_0}^{f_{final}} (\mid Y_1^n(jf_i) - Y_d^n(jf_i) \mid + \\ + &\mid Y_2^n(jf_i) - Y_d^n(jf_i) \mid + \mid Y_3^n(jf_i) - Y_d^n(jf_i) \mid) \end{aligned}$$

where

- $err_{untuned}$ is the total (i.e. taken for all frequencies) cumulative (taken for all three input signals used) error for the untuned system
- Y_1^n, Y_2^n, Y_3^n are the DFTs of the measured and normalized outputs of the untuned system for inputs u_1, u_2, u_3;
- Y_d^n is the DFT of the normalized output measured for any of the inputs u_1, u_2, u_3;
- f_0 and f_{final} determine the range of frequencies used (as determined by the limiting case of the Nyquist Frequency Criterion, i.e. $f = \frac{1}{2d}$, where d is the sampling rate in seconds and f is frequency in Hz).

The equation describing the tuned system error is :

$$
\begin{aligned}
& err_{tuned} \\
& f_{final} \\
= & \sum_{f_i=f_0} (\mid Y_1^{tn}(jf_i) - Y_d^n(jf_i) \mid + \\
+ & \mid Y_2^{tn}(jf_i) - Y_d^n(jf_i) \mid + \mid Y_3^{tn}(jf_i) - Y_d^n(jf_i) \mid)
\end{aligned}
$$

- where err_{tuned} is the final (obtained at the end of tuning process) total (obtained for all frequencies in the investigated range) cumulative (obtained for all three input signals) error for the tuned system;

- $Y_1^{tn}, Y_2^{tn}, Y_3^{tn}$ are the DFTs of the measured and normalized outputs of the tuned system for inputs u_1, u_2, u_3.

The Table 4-3 illustrates the actual results obtained.

The analysis of these results should provide a clear indication of what and how much compensation took place. Looking at the graphs one can see immediately which frequencies were effected and by how much, while the numerical values of total cumulative errors achieved will give us some definite numerical representation of the compensation that took place.

4.4 Testing results in the frequency domain

The results were analyzed by comparing the tuned and untuned system outputs to the reference system output. The comparison is made via magnitude spectra to indicate what frequencies were compensated for. For the frequency analysis, plots were used to compare the effects on the process output of the initial and final values of parameters in the linear part of the compensator. Comparisons were also made between the magnitude spectra of responses of the reference system and the compensated closed loop system.

System model	initial compensator	final compensator
1	$\frac{s+1}{s^3+s^2+s+1}$	$\frac{0.3660s+0.1388}{s^3+0.003s^2+0.5786s+0.0503}$
2	$\frac{s+1}{s^3+s^2+s+1}$	$\frac{0.1852s+0.2234}{s^3+0.1567s^2+0.53s+0.116}$
3	$\frac{s+1}{s^3+s^2+s+1}$	$\frac{0.2351s+0.3110}{s^3+0.5724s+0.0435}$
4	$\frac{s+1}{s^3+s^2+s+1}$	$\frac{0.1603s+0.5405}{s^3+0.1249s^2+0.5690s+0.1148}$
5	$\frac{s+1}{s^3+s^2+s+1}$	$\frac{0.4225s+0.5537}{s^3+1.6252s^2+0.7846s+1.1629}$
6	$\frac{s+1}{s^3+s^2+s+1}$	$\frac{0.7626s+1.3556}{s^3+2.0312s^2+0.4724s+1.3717}$
7	$\frac{s+1}{s^3+s^2+s+1}$	$\frac{0.8768s+0.994}{s^3+1.0671s^2+0.8255s+0.8002}$
8	$\frac{s+1}{s^3+s^2+s+1}$	$\frac{0.49s+0.24}{s^3+0.0s^2+0.47s+0.04}$
9	$\frac{s+1}{s^3+s^2+s+1}$	$\frac{63.71s+0.0}{s^3+0.0006s^2+31.38s+0.18}$

Table 4.2: Table indicating initial and final parameters of compensator

4.4.1 Initial and final parameter values of the linear part of the compensator

Before the optimization is started the parameters of the linear part of the compensator are set up. These are the initial parameters and in this situation the compensator is called untuned. After the optimizer finished its job, the final parameter values obtained determined the tuned compensator. The initial and the final compensator parameters are given in the Table 4-2.

We note from table 4.2. that the coefficient for s^3 remained unchanged in the tuned compensator.

4.4.2 Graphical representation of results in the frequency domain

The original design was made using the chirp signal. So, in order to test the quality of the design, we first use another type of signal, the step signal, and observe the resulting

frequency domain characteristics. We expect that our design will also be good for this changed type of signal. Figures 4.3 - 4.11 illustrate the compensation level between the tuned and untuned system using a step type input signal. The thick line in the figures indicates magnitude spectra of the error between the tuned system and the desired system (err_t). The thin line indicates the magnitude spectra of the error between the untuned system and the desired system (err_u). The ideal situation would be zero error between the reference model and the tuned system in the frequency domain. This however was not achieved, but a significant reduction of the magnitude of the error in the frequency domain for the tuned system response is obtained, compared to the untuned one.

The programs used to generate these graphs in Matlab are given in Appendix A.

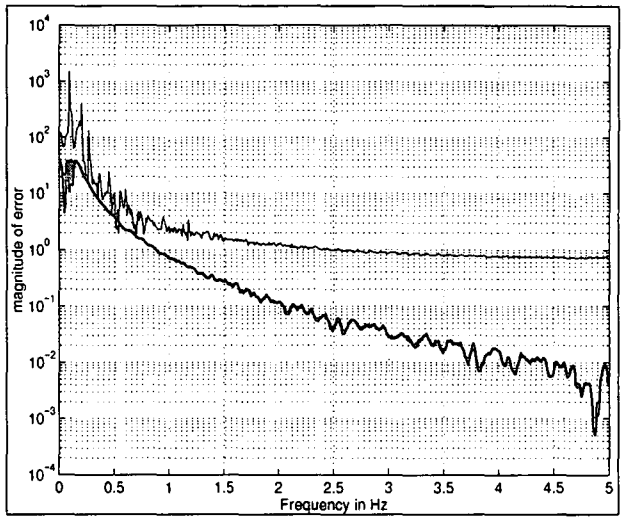

Figure 4-3: Magnitude spectra of errors for model No.1

4.4.3 Numerical results for tuned and untuned systems

In Table 4.3 the errors between the tuned and untuned system are given. The model of each system is indicated and for each of the systems a single rounded value of cumulative total error is shown before and after optimization. This is appears to be very useful in determining approximately how much optimization has taken place.

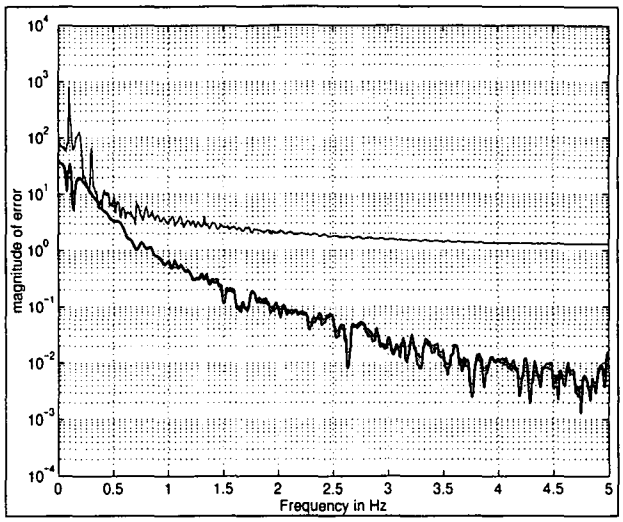

Figure 4-4: Magnitude spectra of errors for model No.2

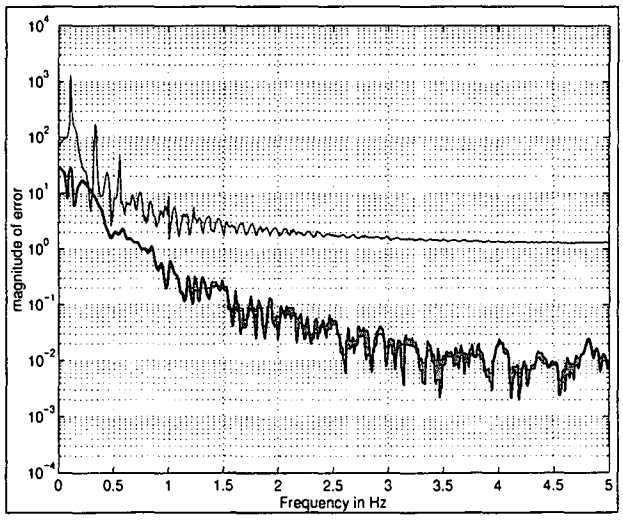

Figure 4-5: Magnitude spectra of errors for model No.3

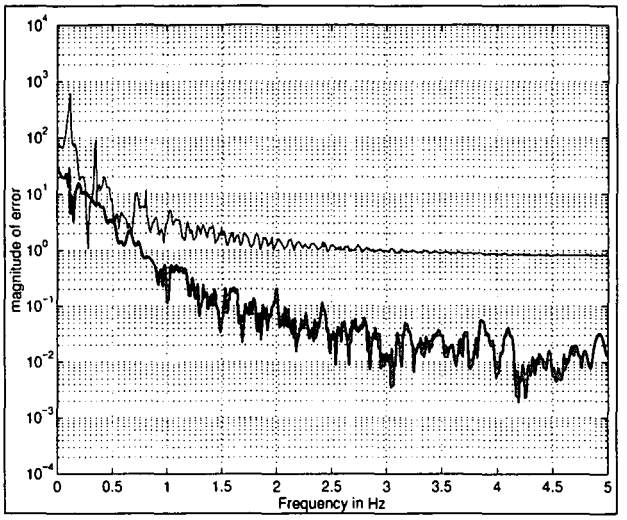

Figure 4-6: Magnitude spectra of errors for model No.4

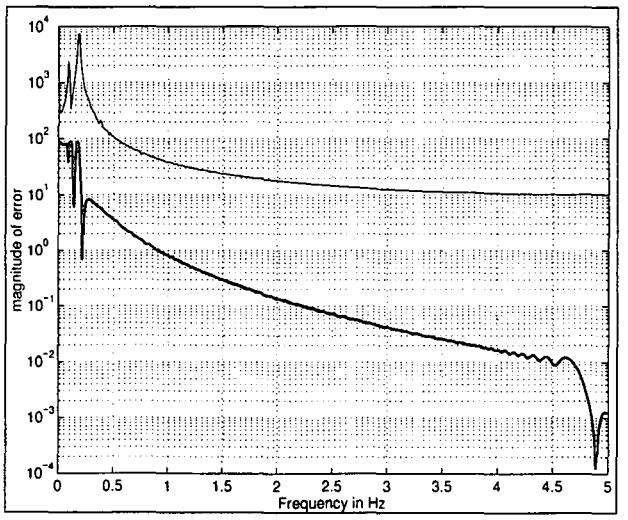

Figure 4-7: Magnitude spectra of errors for model No.5

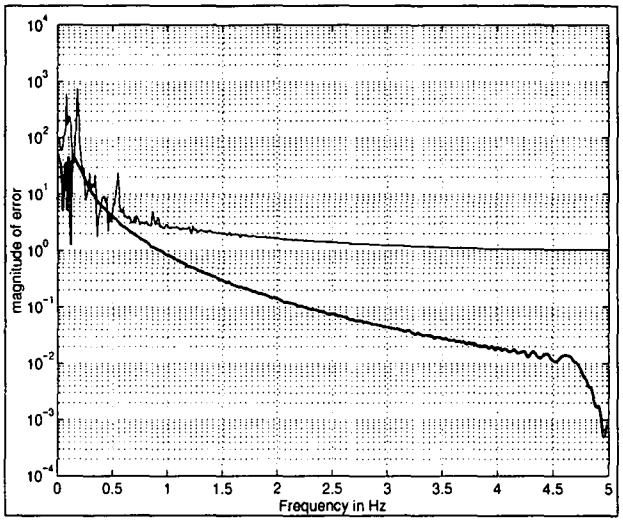

Figure 4-8: Magnitude spectra of errors for model No.6

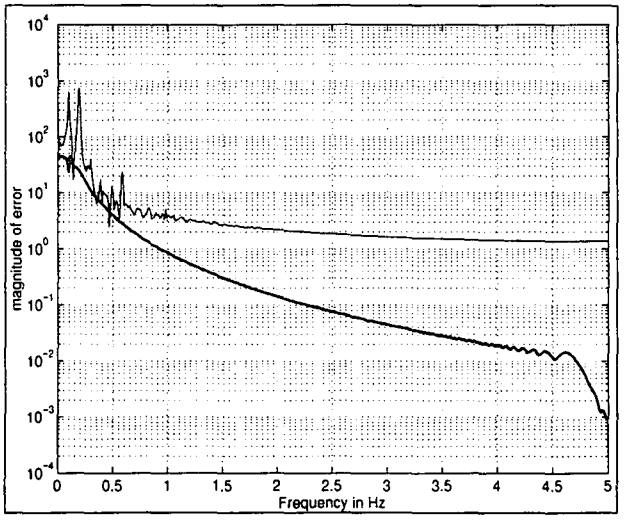

Figure 4-9: Magnitude spectra of errors for model No.7

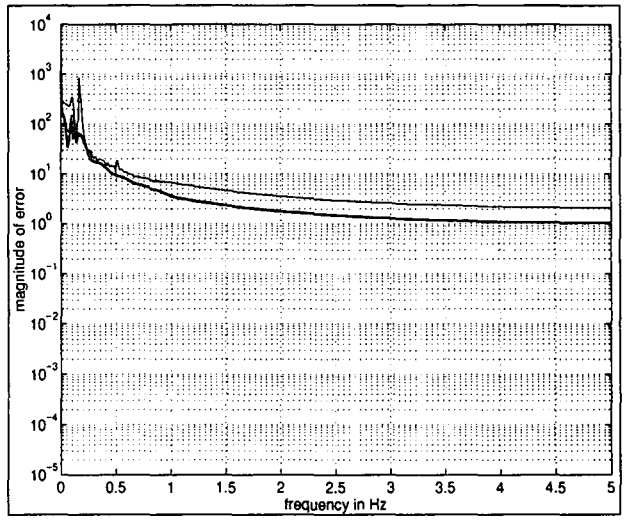

Figure 4-10: Magnitude spectra of errors for model No.8

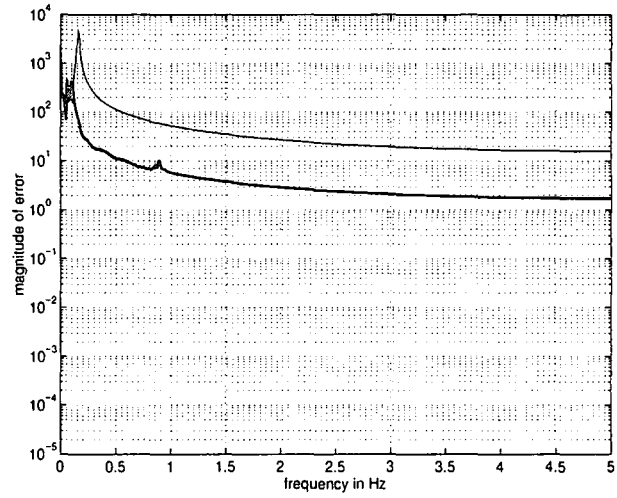

Figure 4-11: Magnitude spectra of errors for model No.9

Model No.	tuned system	untuned system
1	1017	5739
2	860	4097
3	706	5481
4	649	3884
5	1721	55521
6	1099	4784
7	1201	5545
8	5090	11005
9	8172	43110

Table 4.3: Table indicating total cumulative error for each of the examined systems

4.4.4 Comments

Inspection of the results obtained for the frequency domain characteristics show that the optimizer performed very well. Systems have been compensated for virtually every frequency to a great extent. Numerical values of the cumulative total errors show that there is still some discrepancy between the desired system and the tuned system. However, for all the systems significant optimization was achieved. So, we can conclude that the main aim has been achieved and this was tested with a different input signal than those used for design.

4.5 Time domain testing of results: chirp input signal

We will further test in the time domain how the tuned and untuned systems respond to the chirp signal. This signal has been used in the design process. It should be pointed out that our primary interest was not a design to achieve time domain specification, but rather frequency domain behavior.

4.5.1 Method of testing system response to a chirp signal input

Again we compare the outputs of the tuned and untuned systems to the output of the reference system. This comparison is done in the time domain and thus conversion of the response signals to the frequency domain is not necessary. The input signal used in the simulation of the tuned and untuned system is the chirp signal. To get accurate and representative results for time domain analysis the difference in the dead times of the two systems have to be taken into account. In our experiments the reference system does not contain the dead time, while the compensated system does. This problem could be overcome by introducing an equal amount of dead time in the reference system. So, the dead time of $\lambda = 0.4$ seconds was introduced into the reference system to enable proper comparison in the time domain (Fig. 4.1).

Fig. 4.2 illustrates the process of error generation in the time domain. With reference to Fig 4.2, y_{d1} is the output of the desired system in the time domain; y_1 is output of the untuned system in the time domain; y_2 is the output of the tuned system in the time domain; and err_1 and err_2 are the errors between the reference system and the untuned and tuned systems respectively.

The comparison of the tuned and untuned systems is achieved in the time domain by comparing the error of responses between the desired system and the tuned system (err_t) and the error between the desired system and the untuned system (err_u).

Error vectors are presented in the graphs Fig. 4.13 to Fig. 4.21. They are defined by

$$err_t(t) = \mid y_1(t) - y_{d1}(t) \mid,$$

where y_1 is the time domain response of the untuned closed loop system to the chirp signal input and y_{d1} is the response (to the same input signal) of the reference model shifted by dead time λ.

Also,

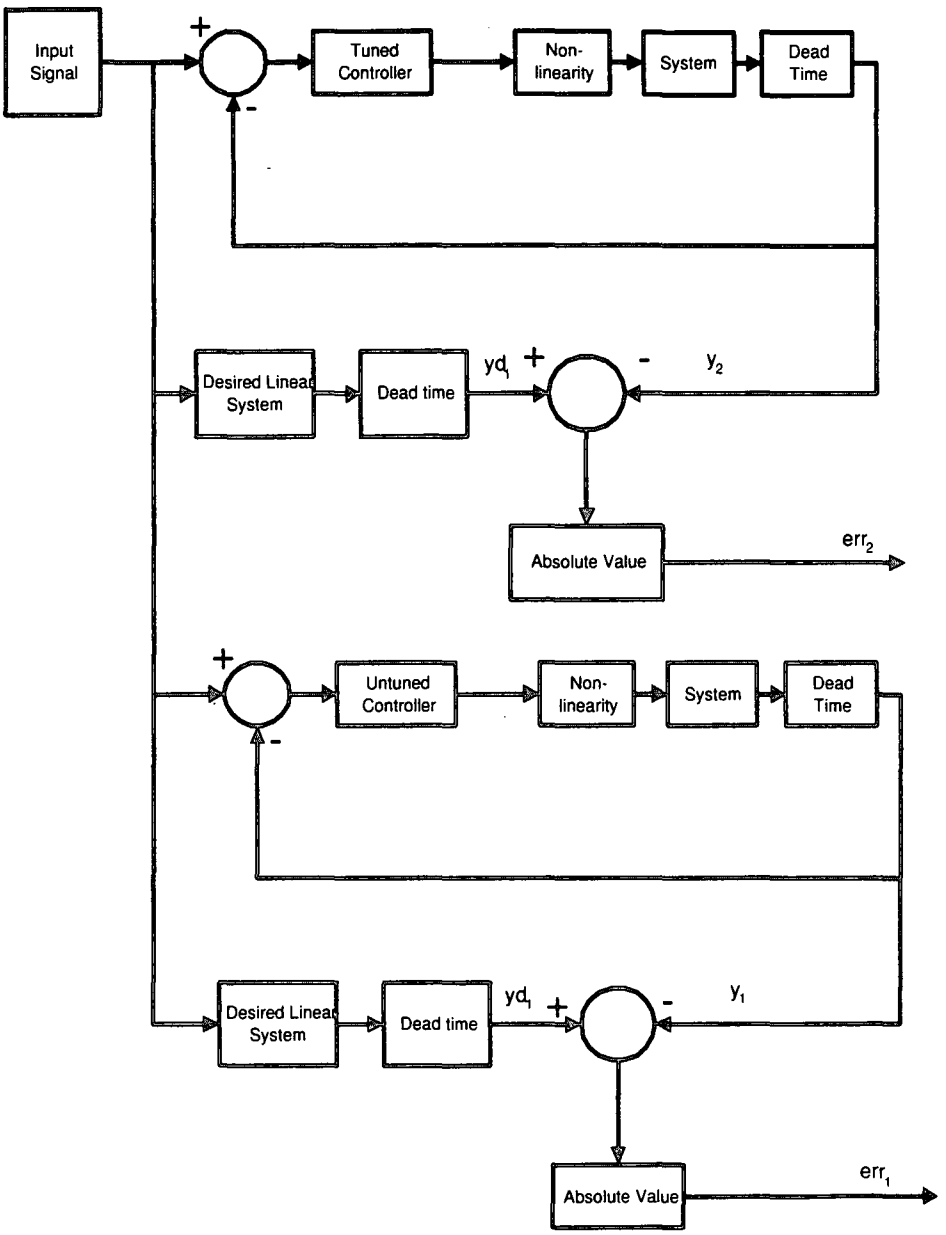

Figure 4-12: System structure used for the generation of error in the time domain

$$err_u(t) = \mid y_2(t) - yd_1(t) \mid,$$

where y_2 is the output of the tuned closed loop system.

The results are plotted on a time versus signal amplitude. These plots give us a good indication of the transient process duration and the quality of the compensation that took place. In addition to the graphs, we give a numerical indication of the errors achieved. This gives useful indication of the approximate amount of compensation.

The errors in the time domain are obtained as

$$err_2 = \sum_{t=0}^{t=final} |y_2(t) - yd_1(t)|,$$

$$err_1 = \sum_{t=0}^{t=final} |y_1(t) - yd_1(t)|,$$

where err_2 is total absolute error between the tuned and the reference system in the time domain; and err_1 is the total absolute error between the untuned and the reference system in the time domain; t_{final} denotes final time of the simulation window sample; y_{d1} is the output of the reference system to a chirp signal input; y_2 is the output of the tuned system. The difference of the two errors, err_2 and err_1, indicate the amount of total compensation that took place in the time domain between the tuned and untuned system using a chirp signal input. These errors are illustrated in Table 4-4.

4.5.2 Results of the test with the chirp signal input

Figures 4.13 - 4.21 illustrate the time behavior of the tuned and untuned system using a chirp signal input. The thick line in the graphs indicates error in the time domain between the tuned system output and the desired system output (err_2). The thin line

indicates the error in the time domain between the untuned system output and the desired system output (err_1). The ideal situation with the tuned system would be if there is zero error between the reference model time domain behavior and the tuned system time domain behavior. This was not achieved simply because our objective was design in the frequency domain, which generally is not equivalent to the time domain design specification. However, a significant reduction of error (with regard to the behavior of the reference system) of the tuned system is noted (compared to the error produced by the untuned system).

For further reference to programs that generated these graphs see Appendix A.

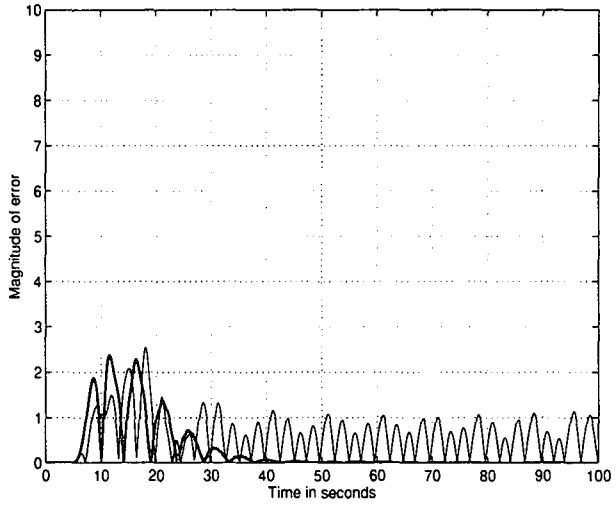

Figure 4-13: Error in time domain for tuned and untuned model No.1 for a chirp signal input

4.5.3 Numerical results of the time domain test with chirp input signal

The integrals of absolute errors for the responses are given for the tuned and untuned compensators in Table 4-4.

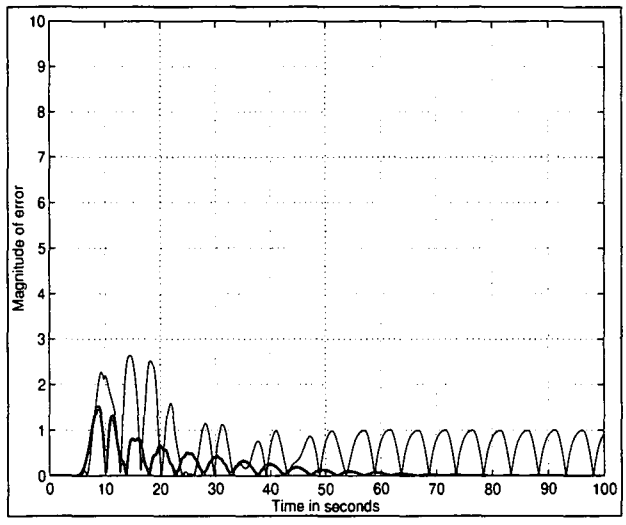

Figure 4-14: Error in time domain for tuned and untuned model No.2 for a chirp signal input

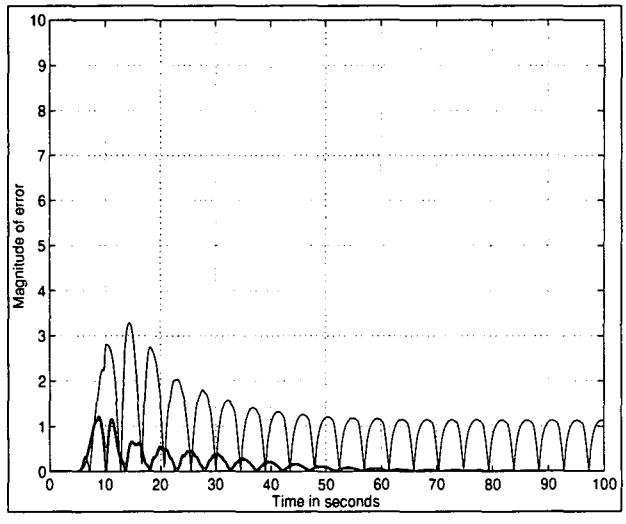

Figure 4-15: Error in time domain for tuned and untuned model No.3 for a chirp signal input

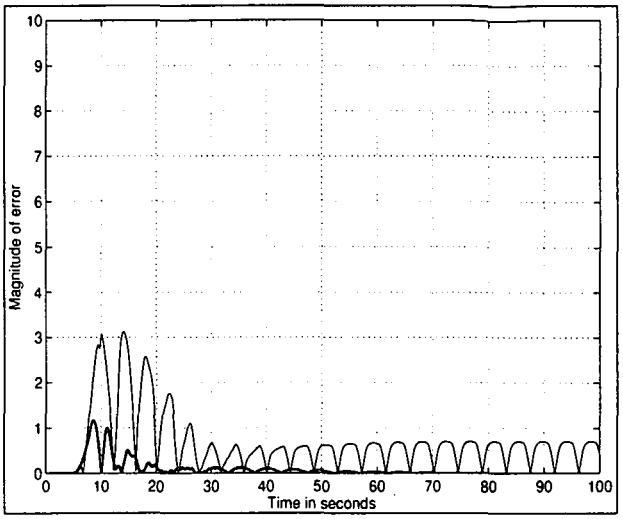

Figure 4-16: Error in time domain for tuned and untuned model No.4 for a chirp signal input

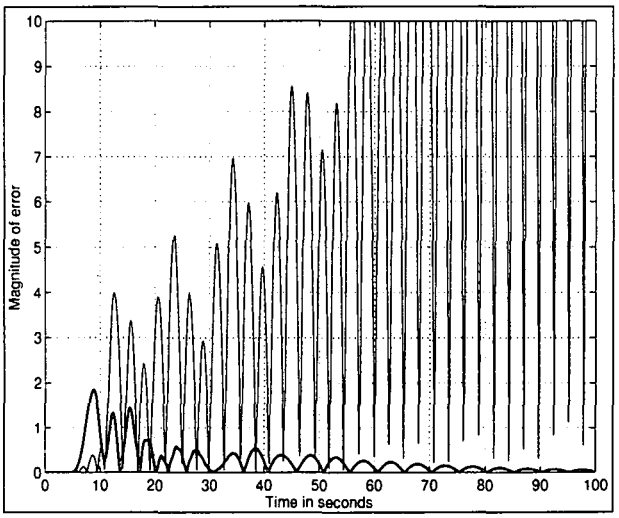

Figure 4-17: Error in time domain for tuned and untuned model No.5 for a chirp signal input

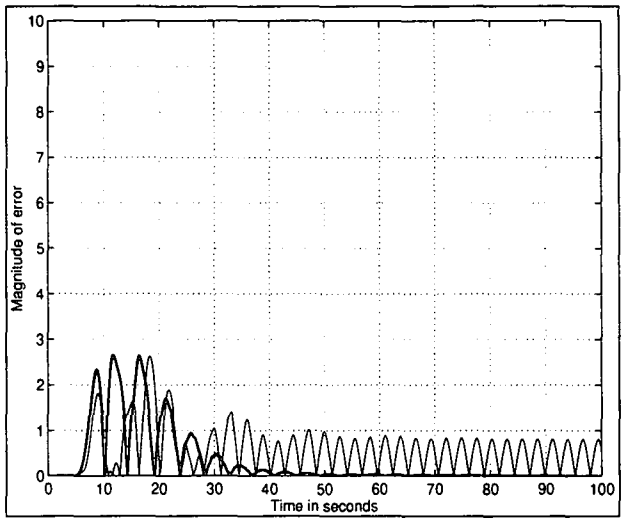

Figure 4-18: Error in time domain for tuned and untuned model No.6 for a chirp signal input

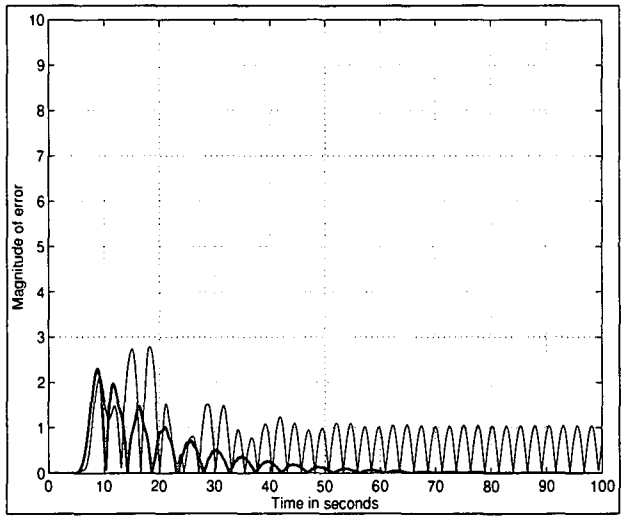

Figure 4-19: Error in time domain for tuned and untuned model No.7 for a chirp signal input

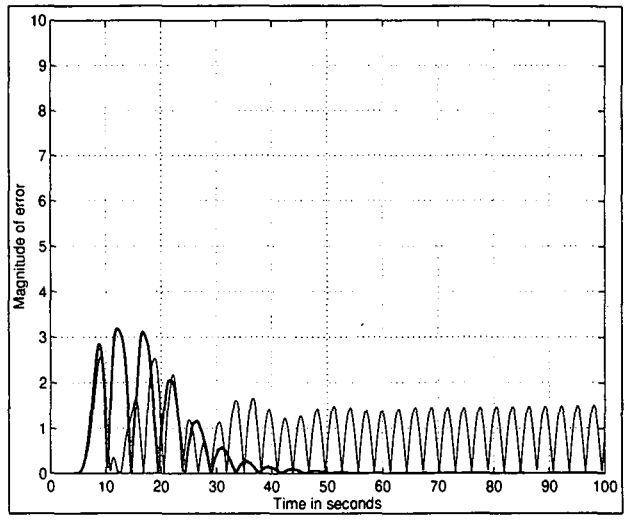

Figure 4-20: Error in time domain for tuned and untuned model No.8 for a chirp signal input

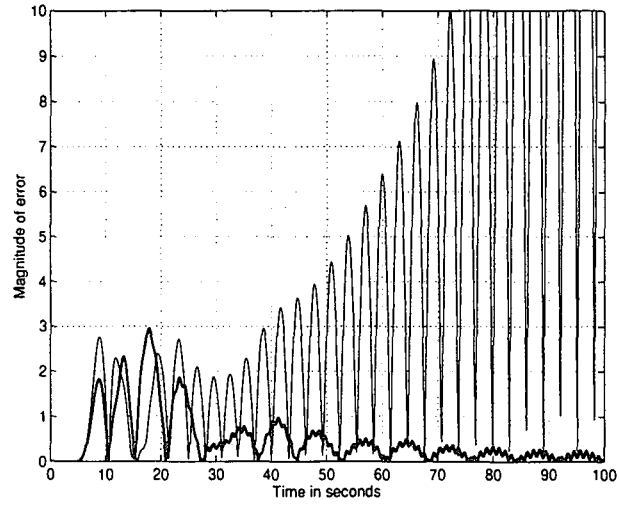

Figure 4-21: Error in time domain for tuned and untuned model No.9 for a chirp signal input

System model No.	uncompensated system	compensated system
1	1032	51
2	580	37
3	768	30
4	499	23
5	2696	50
6	445	63
7	658	50
8	888	393
9	4854	491

Table 4.4: Total error of each of the examined systems in the time domain with chirp input signal

4.5.4 Comments

The behavior of the tuned system in the time domain with the chirp signal is good. The general compensation of the system was achieved. According to the numerical results the error of the tuned system is approximately at the level of 5% compared to that of the untuned system. So, although the aim was not to design compensator for good behavior in the time domain, we notice that with the input signal used in the design process, the time domain behavior is also good.

However, tests of the time domain behavior with the step type signal were also made. Recall that the frequency domain behavior of the tuned system was very good in the frequency domain with the step input signal.

4.6 Time domain tests of system responses to a step input signal

Simulations of the tuned and untuned system with a step input signal were performed. We use the same techniques of comparison as in the case of the chirp signal. The step signal has jump from 0 amplitude to 1 amplitude at the instant of 5 seconds from the start of simulation.

4.6.1 Simulation results with step input signal in time domain.

Figures 4.22 - 4.30 illustrate the behavior of the tuned and untuned system using a step input. The thick line indicates the error between the tuned and the desired system and the thin line indicated the error between the untuned and the desired system.

Generation of all graphs and numerical values of the errors are analogous to the case with the chirp input signals. The same programs have been used.

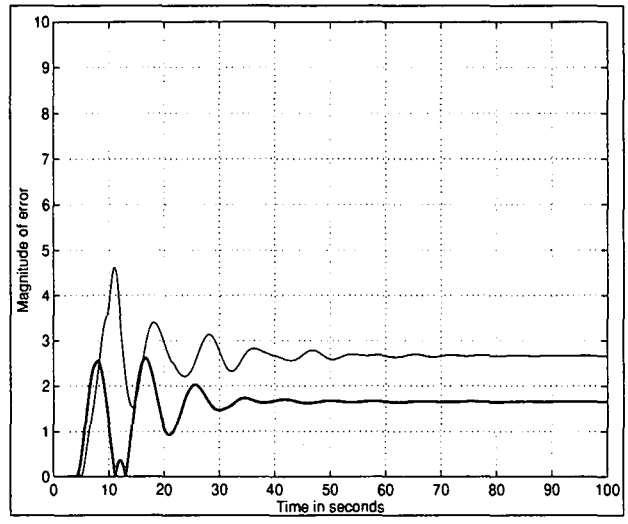

Figure 4-22: Error in time domain for tuned and untuned model No.1 for a step signal input

4.6.2 Total error in the time domain for the step input signal

The results are further expressed numerically for ease of comparison. Table 4-5 reflects the integral absolute error in the time domain of responses of the tuned and untuned system with regard to the reference model response when the input signal was of the step type.

For the Matlab programs generating these graphs and values refer to Appendix A.

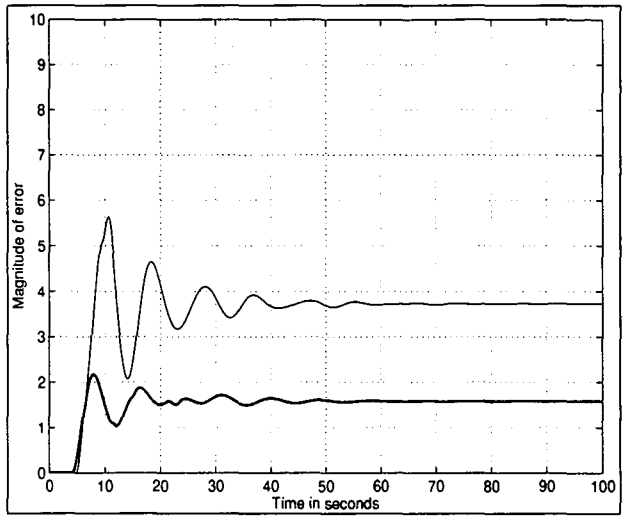

Figure 4-23: Error in time domain for tuned and untuned model No.2 for a step signal input

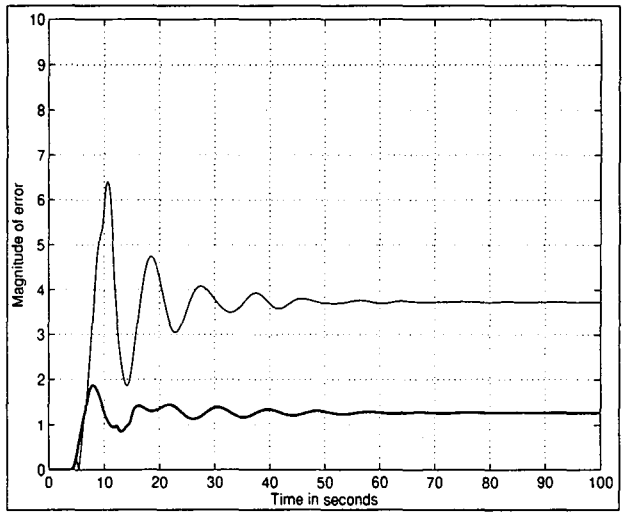

Figure 4-24: Error in time domain for tuned and untuned model No.3 for a step signal input

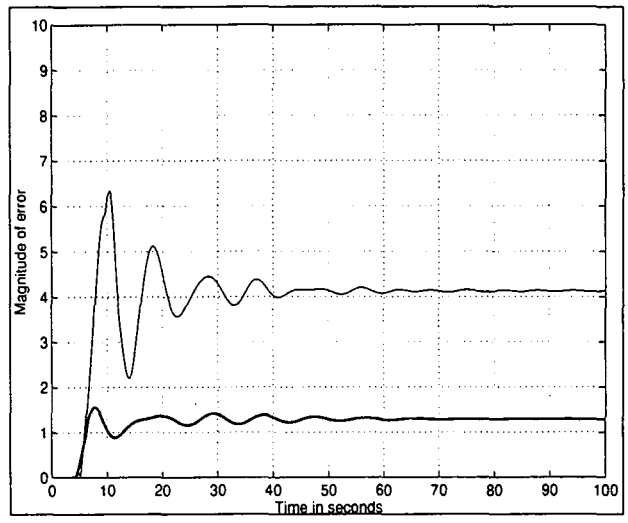

Figure 4-25: Error in time domain for tuned and untuned model No.4 for a step signal input

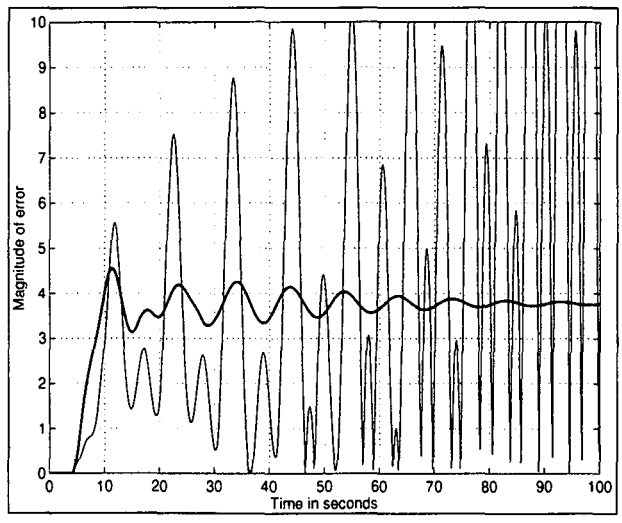

Figure 4-26: Error in time domain for tuned and untuned model No.5 for a step signal input

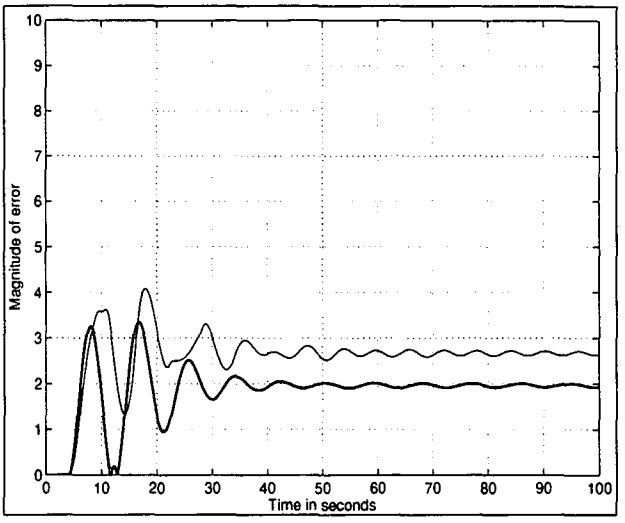

Figure 4-27: Error in time domain for tuned and untuned model No.6 for a step signal input

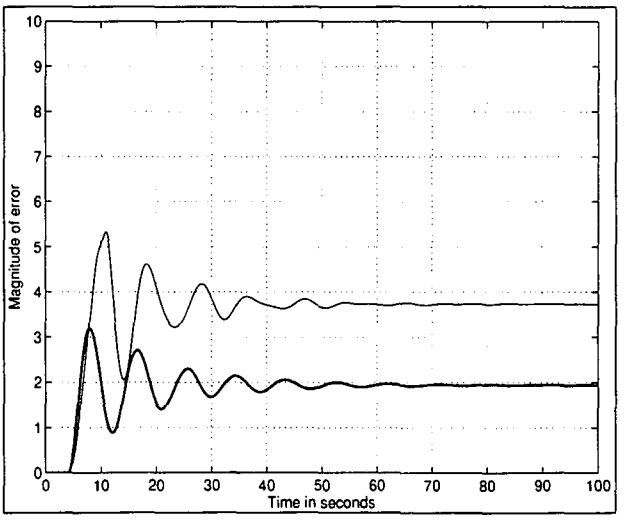

Figure 4-28: Error in time domain for tuned and untuned model No.7 for a step signal input

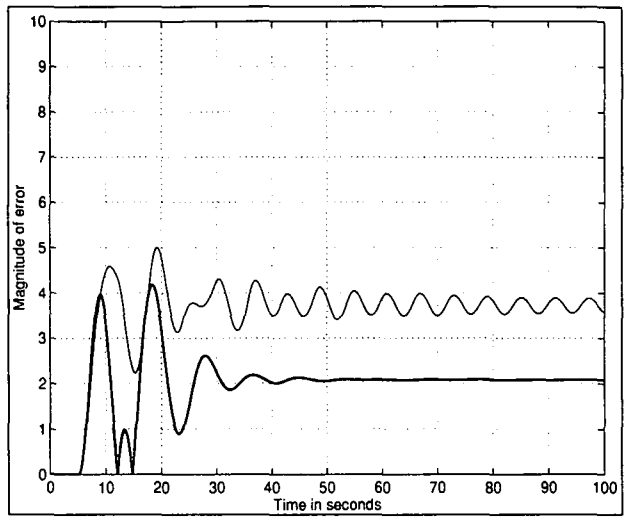

Figure 4-29: Error in time domain for tuned and untuned model No.8 for a step signal input

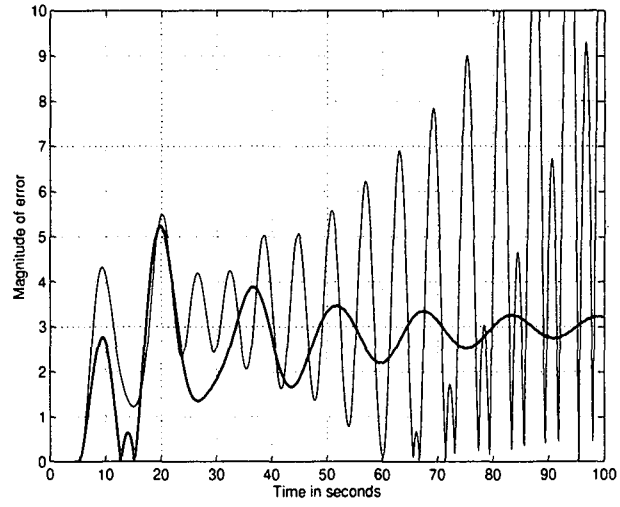

Figure 4-30: Error in time domain for tuned and untuned model No.9 for a step signal input

Model No.	untuned system	tuned system
1	4416	2486
2	4829	2356
3	5070	1979
4	5285	1994
5	8175	5007
6	4108	2853
7	5012	2810
8	3483	1969
9	3907	2541

Table 4.5: Total error obtained for step input signal

4.6.3 Comments

Unfortunately, for the step input the tuned system did not perform in the time domain as well as with the chirp signal. Observing the plots for time versus amplitude there a steady state error. The overall compensation is also affected because of the steady state error (Table 4.5). However, the tuned system has a much better behavior than the untuned. So, the frequency domain design has reflected positively to the time domain behavior in this case too. The reason for the steady state error is that the optimizer attempted to reduce the error for all frequencies in the range in an equal fashion by making the total error as small as possible. Thus, the component with the zero frequency which controls the steady state error for the step signal did not have any preference to other frequency components and its influence to the overall score in the frequency domain was not too significant.

4.7 Time domain response of desired and tuned system

The figures 4-31 to 4-39 illustrate the time domain responses for tuned (thick curve) and desired (thin curve) systems. Compare with figures 4.40-4.48.

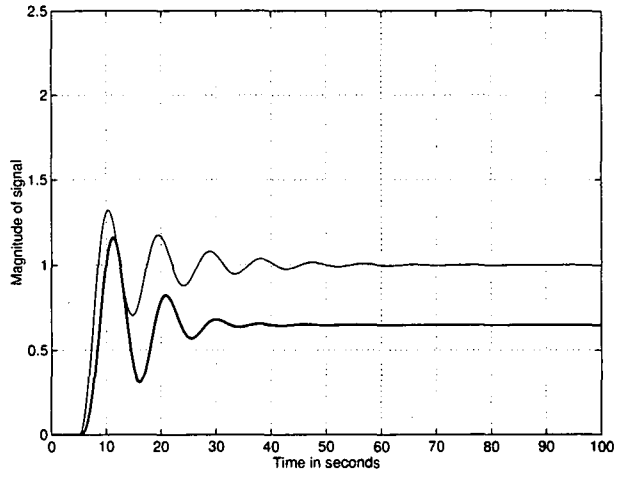

Figure 4-31: Time domain responses for step input signal for model No.1

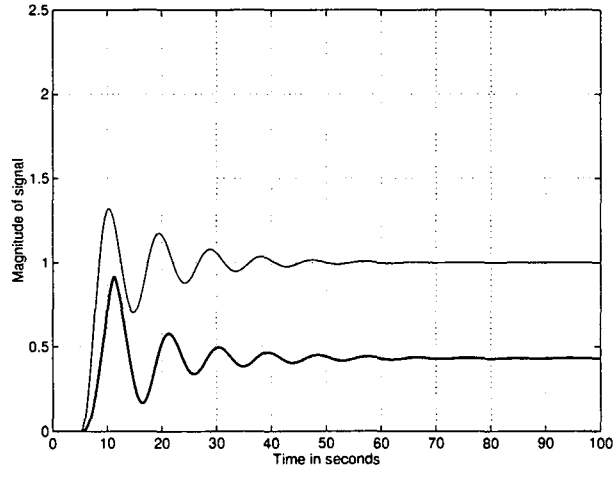

Figure 4-32: Time domain responses for step input signal for model No.2

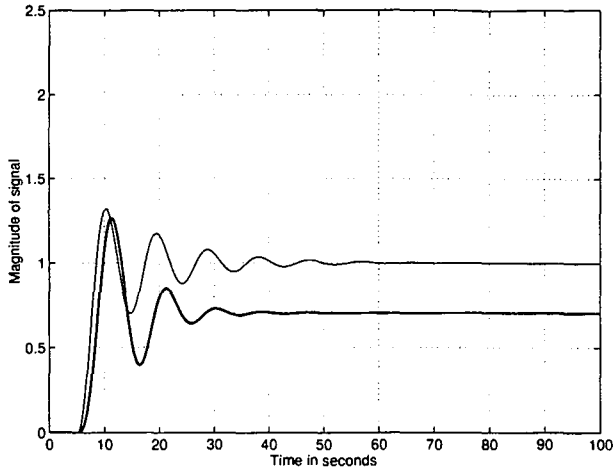

Figure 4-33: Time domain responses for step input signal for model No.3

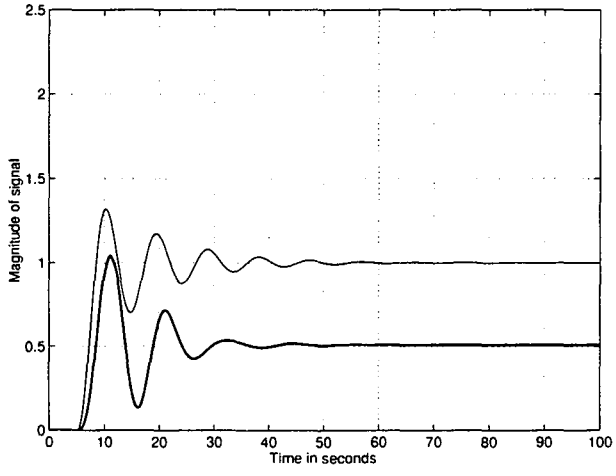

Figure 4-34: Time domain responses for step input signal for model No.4

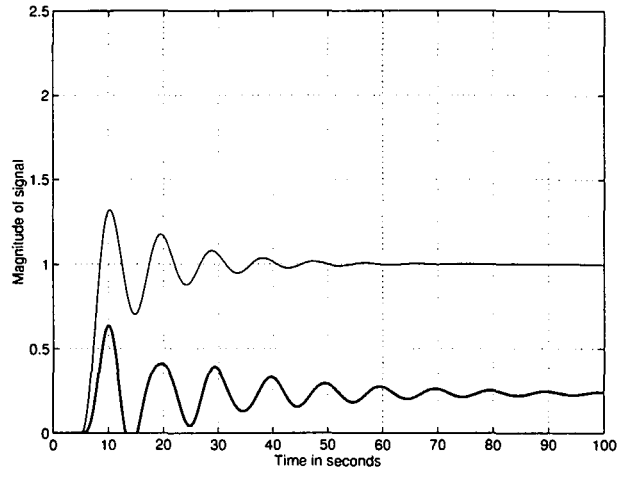

Figure 4-35: Time domain responses for step input signal for model No.5

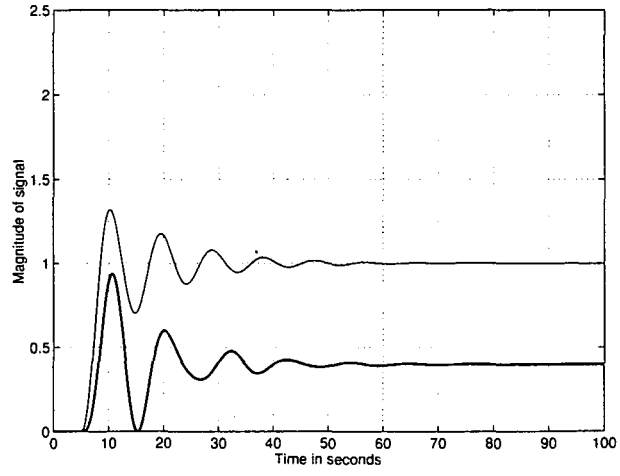

Figure 4-36: Time domain responses for step input signal for model No.6

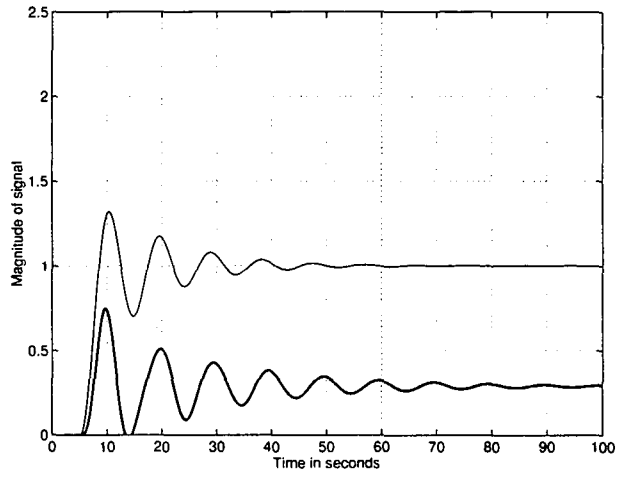

Figure 4-37: Time domain responses for step input signal for model No.7

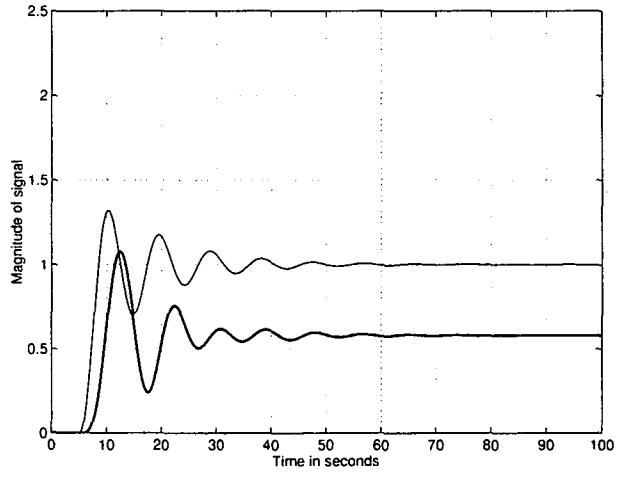

Figure 4-38: Time domain responses for step input signal for model No.8

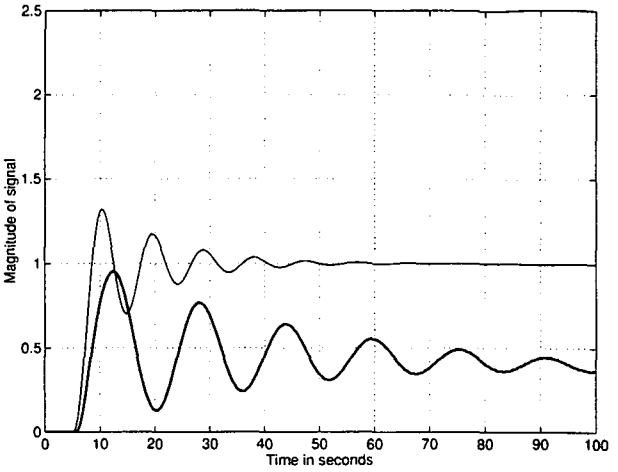

Figure 4-39: Time domain responses for step input signal for model No.9

4.8 Time domain response of desired and untuned systems

The Figs.4.40-4.48 illustrate the time domain responses for untuned (thick curve) and desired (thin curve) systems.

4.9 General conclusions from these experiments

The experiments and analysis of the frequency domain design indicate that considerable compensations of systems was achieved. The frequency characteristics of the compensated system match closely the desired system as indicated in the Fig 4.3 to 4.11. For our purposes the goal was achieved in that the system behaves very close to the desired one in the frequency domain. However, we have taken the analysis a step further and analyzed the experiments with the time domain behavior. The purpose of this was to examine whether the time domain characteristics also match, as was shown for the frequency characteristics.

From the time domain analysis of the system responses, we noticed that good op-

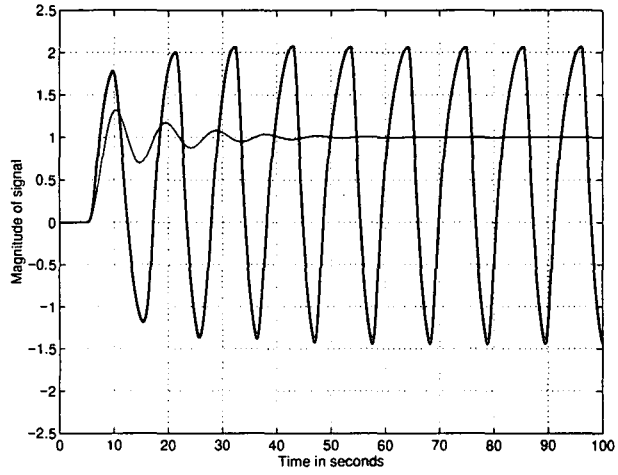

Figure 4-40: Time domain responses for step input signal for untuned model No.1

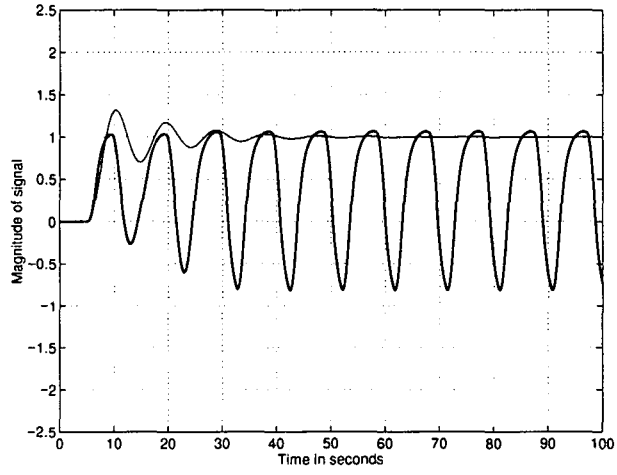

Figure 4-41: Time domain responses for step input signal for untuned model No.2

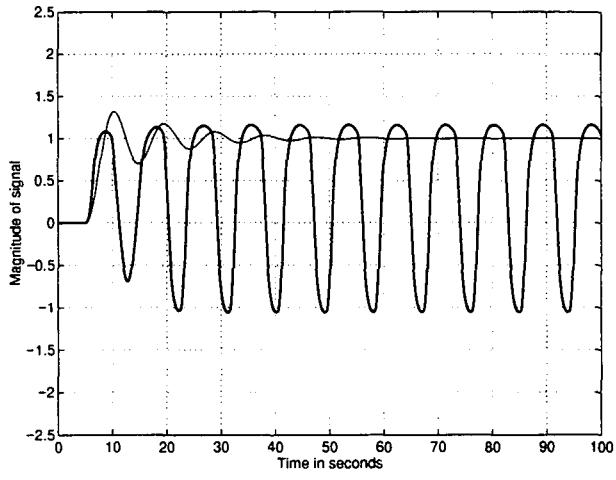

Figure 4-42: Time domain responses for step input signal for untuned model No.3

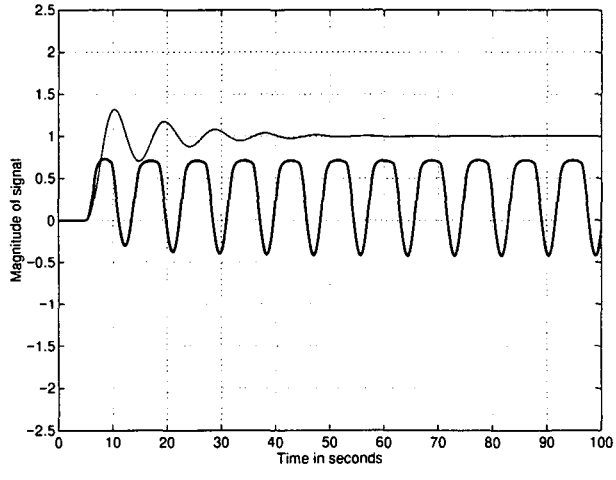

Figure 4-43: Time domain responses for step input signal for untuned model No.4

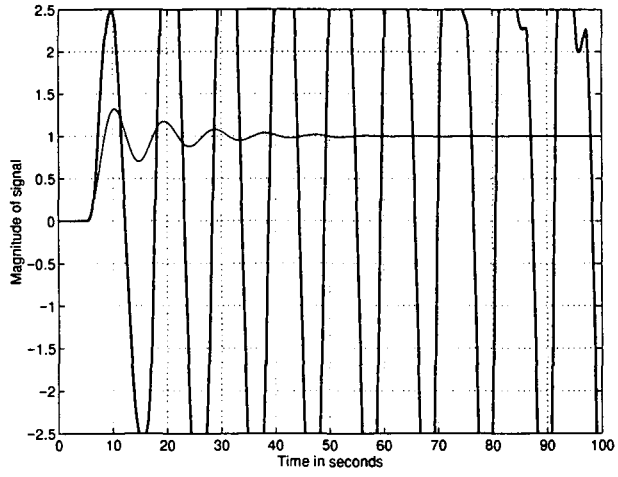

Figure 4-44: Time domain responses for step input signal for untuned model No.5

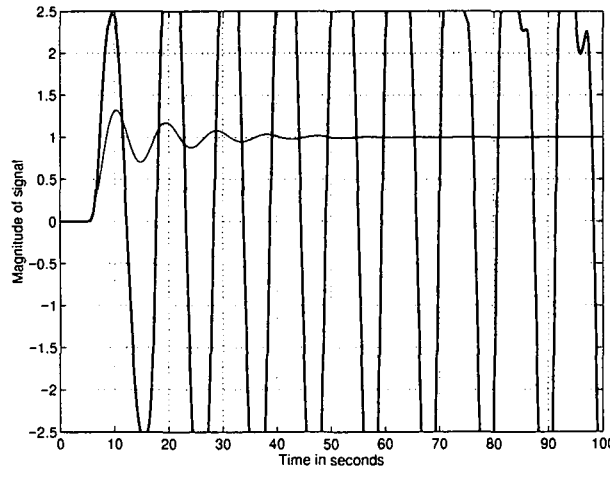

Figure 4-45: Time domain responses for step input signal for untuned model No.6

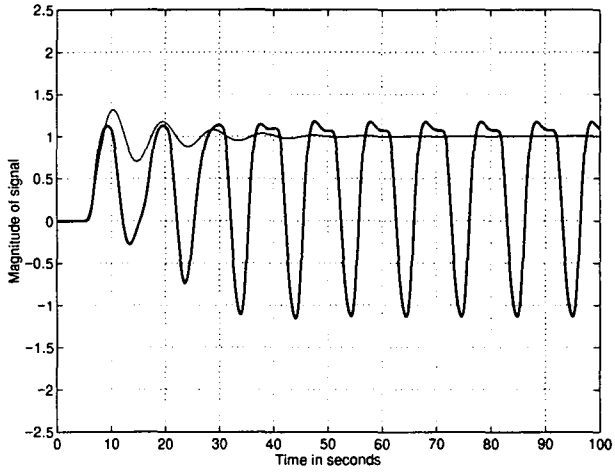

Figure 4-46: Time domain responses for step input signal for untuned model No.7

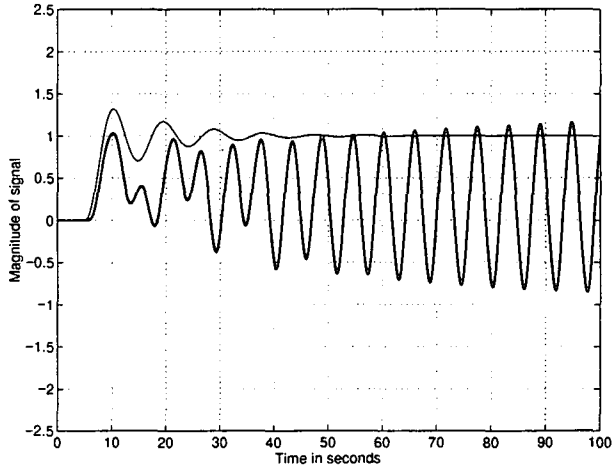

Figure 4-47: Time domain responses for step input signal for untuned model No.8

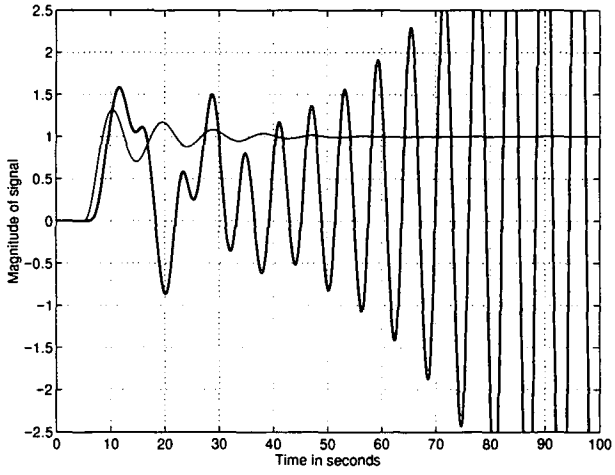

Figure 4-48: Time domain responses for step input signal for untuned model No.9

timization was achieved for the chirp signal input, but not so good for a step input. System responses to a step input produced a steady state; hence there is a significant overall error in these cases. However, these errors are much larger for the uncompensated systems, some of which are also unstable. From the presence of the steady state error in the time domain, it could be concluded that good frequency compensation does not necessary mean good time domain compensation. To reduce this steady state error we can use several approaches. For example, we can use simultaneous time and frequency domain optimization (Bajić and Stavrou 1997). Or, we may give preference to low frequency components in the optimization process. This however goes beyond the scope of this study.

Chapter 5

Conclusions

This study proposed and tested a method for the design of compensators in the frequency domain. The method is developed for a class of nonlinear time invariant systems that have a dead time. The computational tests performed have shown that the method works well and that the frequency characteristics of the original system can be adjusted to match closely to the desired ones. The method needs further development in terms of the improvement of the optimization method, selective control of specific frequency domains in optimization process and generalization to other structures of the system to be compensated.

5.1 Advantages of frequency optimization

The technology of compensator design process in the frequency domain is still in the developmental phase, but the results obtained so far look promising. The computational subroutines developed for the purpose of this study may, in principle, be converted into a single integrated circuit solution. Even though the in-circuit solution is still remote, the programs written may be converted to a single user friendly executable PC program as a general solution of frequency compensation of nonlinear systems with dead time.

This method is applicable for the design of compensators for systems with arbitrary

frequency characteristics. The optimization routines used for determination of compensator parameters do not rely on the structure of the system for which the compensator is to be designed. Also, the structure of the compensator is, in a way, free for selection. This gives the designer flexibility to choose the appropriate structure for the compensator. The system may also remain partially unknown and the design can still be conducted. If the frequency response of the system is sufficiently known, i.e. if it is known for signals of different magnitudes and if they produce sufficiently dense spectra, then the system could be represented sufficiently well in computer modelling for the design process.

Even though compensation of the class of systems considered relates mainly to control applications, this may also be extended to communication problems. A simple example is the optimization of a device that contains an RF amplifier, DSP unit and a transmission line. The RF amplifier may be a high frequency nonlinear transistor circuit, whose output is processed by the DSP unit and transmitted through the transmission line. This DSP processing and transmission results in the introduction of a dead time. Since often some specified frequency input/output characteristics of such systems are needed, the reference model that possesses such characteristics can be defined. Consequently, one can adjust the device to match closely the desired frequency characteristics. Thus, we need the optimal values of an input matching impedance device (compensator) and we can apply the methodology presented in this study.

Also, the proposed optimization technique is applicable to arbitrary control system that contains components that can be adjusted, for example a PID or other types of controllers.

5.2 Limitations of the method

The method is computationally intensive due to requests for transformations from the time to frequency domain and due to the optimization part that requires simulation of systems for different input signals on a significant time interval. Also, the more para-

meters the optimizer has to adjust, the more computations are needed. When too many parameters are subject to tuning, the optimizer may loose its efficiency and give out wrong or inaccurate results.

For some higher order systems the optimization process may not be able to reduce the error sufficiently and the tuned system need not have good behavior in the whole frequency range. For systems that are unstable or on the boundary of stability the optimizer may also not be able to give accurate results.

The optimal structure of the compensator is not investigated. There could be some solutions to this problem. For example, the optimizer could optimize several predefined structures of the compensator and then select the best obtained solution.

The optimization of an unknown nonlinear time invariant SISO systems with dead time can be achieved using the method described above. This however may not be a general solution to all nonlinear systems. The optimization is achieved for a nonlinear function without dynamic elements. The designer who wishes to achieve a time domain optimization for such systems may find that the above solution is not adequate. Thus, this requires further research.

Chapter 6

References

Alkin, O. (1994). *Digital Signal Processing: a laboratory approach using PC-DSP*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey 07632.

Bajić, V. B. (1996). Optimization based design of compensators in frequency and time-frequency domains for nonlinear time delayed systems, Private communication.

Bajić, V. B. and M. Stavrou (1997), Optimization based practical controller design for nonlinear systems in the time-frequency domain, *Mathematical Modelling and Scientific Computing* (11th ICMCM Conference, Washington DC; X. J. R. Avula and Anil Nerode Eds.), Vol.8.

Bode, H. (1945). *Network analysis and feedback amplifier design*, Van Nostrand, New Jersey.

Boonton, Jr. R. C. (1952). Coulomb friction in feedback control systems with statistical inputs, MIT Dynamic Analysis and Control Laboratory, Dept.61, 1 March.

Brayton, R. K., S. W. Director, G. D. Hatchel, and L. Vidigal (1979). A New method for Statistical Circuit Design Based on Quasi-Neuton Methods and Function Splitting, *IEEE Trans. Circuits and Systems*, Vol. CAS-26, pp 784-794.

Brigham, E. O. (1974). *The Fast Fourier Transform*, Prentice-Hall, Engelwood Cliffs.

Coals, J. F. (1956). *Nonlinearities in Control Systems*, University of Cambridge, England, pp.55-91.

Corana, A. *et al.* (1987). Minimizing Multimodal Functions of Continuous Variables with the Simulated Annealing Algorithm, *ACM Transactions on Mathematical Software*, Vol.13, No.3, pp.262-280.

Dennis, Jr. J. E. and R. B. Schnabel (1983). *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Prentice Hall.

Du, D. and P. M. Pardalos (1995). *Minimax and applications*, Kluwer.

Du, D. and J. Sun (1994). *Advances in Optimization and Approximation*, Kluwer.

Fletcher, R. (1988). *Practical Methods of Optimization*, Wiley.

Floudas, C. A. and P. M. Pardalos (1992). *Recent Advances in Global Optimization*, Princeton University Press.

Franklin, G. F. , D. J. Powell and M. L. Workman (1990). *Digital Control of Dynamic Systems*, Second edition, Addison Wesley.

Gill, P. (1982). *Practical Optimization*, Academic Press.

Grace, A (1994). *MATLAB Optimization Toolbox*, The Mathworks Inc., South Natic, MA.

Horowitz, I. M. (1963). *Synthesis of feedback systems*, Academic Press, New York.

Horst, R. and P. M. Pardalos (1995). *Handbook of Global Optimization*", Kluwer.

Horst, R. and H. Tuy (1993). *Global Optimization*, Springer-Verlag.

Horst, R., P. M. Pardalos and N. V. Thoai (1995). *Introduction to global optimization*, Vol. 3,Kluwer.

Ingle, V. K. and J. G. Proakis (1997). *Digital Signal Processing Using MATLAB V.4*, PWC Publishing Company, Boston.

Jackson, L. B. (1991). *Signals, Systems, and Transforms*, Addison-Wesley, Reading, Massachusetts.

Khalil, H. K. (1992). *Nonlinear systems*, Macmillan, Mitchigan State University.

Kochenburg, R. J. (1953). Limiting in feedback control systems, *Trans. AIEE*, Vol.72 , Part 2, pp.119-125.

Kollar, I. (1993).On Frequency Domain Identification of Linear Systems, *IEEE Trans. on Instrumentation and Measurement*, Vol. 42,No. 1, pp. 2-6.

Kraniouskas, P. (1992). *Transforms in Signals and Systems*, Addison-Wesley, Wokingham, England.

Macfarlane, A. G. J. (1970). *Dynamical System Models,* Harrap, UK.

Marquardt, D. (1963). An Algorithm for Least Squares Estimation of Nonlinear Parameters *SIAM J. Appl. Math.*, Vol 11, pp 431-441.

MathWorks Inc, (1997a). *MATLAB, The Language of Technical Computing: Using MATLAB Version 5*, The MathWorks, Inc., South Natic, MA.

MathWorks Inc, (1997b). *SIMULINK, DSP Blockset*, The MathWorks, Inc., South Natic, MA.

More', J. J. and S. J. Wright (1993). *Optimization Software Guide*, SIAM Books.

Narendra, K. S. and L. S. Valavani (1978). Stable adaptive compensator design-direct control, *IEEE Trans. Autom. Control*, Vol. AC-23, No.4, pp.570-583.

Narendra, K. S. and Y. H. Lin (1980). Stable discrete adaptive control, *IEEE Trans. Autom. Control*, Vol. AC-25, No.3, pp.456-461.

Nichols, N. B. (1952). Backlash in velocity lag servomechanism, *Trans. AIEE*, Vol.71, Part 2, June, pp.169-181.

Nyquist, H. (1932). Regeneration Theory, *Bell System Technical Journal*, Vol.11, pp.126-147.

Ogata, K. (1996). *Modern Control Engineering*, Prentice Hall.

Pardalos, P.M. (1987). Generation of large-scale quadratic programs for use as global optimization test problems, *ACM Transactions on Mathematical Software*, Vol.13, No.2, pp.133-137.

Pardalos, P. M. (1991). Construction of test problems in quadratic bivalent programming, *ACM Transactions on Mathematical Software*, Vol.17, No.1, pp.74-87.

Phillips, C. L. and R. D. Harbor (1988). *Feedback control systems*, Prentice-Hall, Englewood Cliffs, N.J.

Powell, M. J. D. (1997). *A Fast Algorithm for Nonlinearly Constrained Optimization Calculations*, Lecture Notes in Mathematics, Springer-Verlag, Vol.630, pp.144-157.

Proakis, J. G. and D. G. Manolakis (1992). *Digital Signal Processing Principles , Algorithms, and Applications*, Second Edition, Macmillan , New York.

Pintelon, R. and P. Guillaume, Y. Rolain and F. Verbeyst (1992). Identification of Linear Systems Captured in a Feedback loop, *IEEE Trans. on Instrumentation and Measurment*, Vol. 41, No. 6, pp. 747-754

Pintelon, R and J. Schoukens (1991). *Identification of Linear systems: a Practical Guide for Accurate Modeling*, Pergamon Press, London

Saunders, W. R., C. L. Phillips and J. M. Maciejowski (1989). *Multivariable feedback design*, Addison Wesley, UK.

Schoen, F. (1993). A Wide Class of Test Functions for Global Optimization, *J. of Global Optimization*, Vol.3, pp.133-137.

Shearer, J. L., A. T. Murphy and H. H. Richardson (1971). *Introduction to System Dynamics*, Addison-Wesley.

Vydiasagar, M. (1993). *Nonlinear Systems Analysis*, Prentice Hall, Englewood Cliffs, New Jersey.

Wellstead, P. E. (1979). *Introduction to System Modelling*, Academic Press.

Wright, M. (1992). *Interior methods for constrained optimization*, Acta Mathematica, Cambridge University Press.

Chapter 7

Appendix A:

7.1 Matlab programs

Please note that all programs were written in Matlab and the simulations were performed with the aid of Simulink. Matlab and Simulink are registered trade marks of MathWorks Inc.

7.1.1 Nonlinear analysis

Graphs were generated for comparison of tuned and untuned systems. The program is called for each individual graph. It relies on the fact that the values y1, y2, y3, were generated using Simulink. The Simulink diagram corresponding to these values is shown below.

{Pre condition: The Matlab has loaded the values of the optimized compensator and the tuned system was simulated.

Post condition: The generation of the error signal in time domain and graph (thickness 1.5mm) form of the nonlinearity in the tuned system}

Error graph generation for tuned system

```
n=length(tt);
```

```
max=1;
ln=1.5;
hndl=semilogy(abs(0.2*y1-0.5*y2)+abs(0.2*y1-2*y3)+abs(0.5*y2-2*y3));
set(hndl,'LineWidth',ln);
axis([0 1000 0.0001 100]);
xlabel('time');ylabel('magnitude');
NUMC=ones(size(NUMC));
DENCC=ones(size(DENCC));
NLVEC1=[1 2 3 4 5 6 7 8 9 10 11 12 13 14 15];
NLVEC = [-(fliplr(NLVEC1)) 0 NLVEC1];
hold on;
sum(abs(0.2*y1-0.5*y2)+abs(0.2*y1-2*y3)+abs(0.5*y2-2*y3))
```

Graph generation for untuned system

{Pre condition: The program above was run in order to initialize the compensator variables to their initial conditions and untuned system was simulated.

Post condition: The generation of the error signal in time domain.}

```
hold on;
ln=0.5;
hndl=semilogy(abs(0.2*y1-0.5*y2)+abs(0.2*y1-2*y3)+abs(0.5*y2-2*y3));
set(hndl,'LineWidth',ln);
grid;
sum(abs(0.2*y1-0.5*y2)+abs(0.2*y1-2*y3)+abs(0.5*y2-2*y3))
hold off;
```

7.1.2 Results in frequency domain for tuned and untuned systems with respect to the desired model

Tuned system results

The analysis used was implemented using Matlab. The programm assumes that the system shown above was simulated with the tuned values pre loaded.

```
n=length(yd1);
max=1;
t=[0:0.1:50];
yproc11=y1;
yproc22=y2;
yproc33=y3;
yd11=yd1;
yd22=yd2;
yd33=yd3;
yyy1 =(fft(yproc11));
yyy2 =(fft(yproc22));
yyy3 =(fft(yproc33));
yds1=(fft(yd11));
yds2=(fft(yd22));
yds3=(fft(yd33));
z=[(abs(yyy1)-abs(yds1))' + (abs(yyy2)-abs(yds2))' +
(abs(yyy3)-abs(yds3))'];
t(n/2:n)=[];
ln=1.5;
hndl=semilogy(t,m);
set(hndl,'LineWidth',ln);
axis([0 35 0.00001 10000]);
```

```
xlabel('frequency');
ylabel('magnitude');
NUMC=ones(size(NUMC));
DENCC=ones(size(DENCC));
NLVEC1=[1 2 3 4 5 6 7 8 9 10 11 12 13 14 15];
NLVEC = [-(fliplr(NLVEC1)) 0 NLVEC1];
hold on;
```

Untuned system results

The following program is a follow up from the program described above. The program must only be run once the above program has been run. The simulink diagram above then must be simulated in order to obtain the necessary input and output variables for the analysis.

```
t=tt*pi*2*(1/step)*(1/time);
yproc11=y1;
yproc22=y2;
yproc33=y3;
yd11=yd1;
yd22=yd2;
yd33=yd3;
yyy1 =(fft(yproc11));
yyy2 =(fft(yproc22));
yyy3 =(fft(yproc33));
yds1=(fft(yd11));
yds2=(fft(yd22));
yds3=(fft(yd33));
z=[(abs(yyy1)-abs(yds1))' + (abs(yyy2)-abs(yds2))' +
(abs(yyy3)-abs(yds3))'];
```

```
t(n/2:n)=[];
z(n/2:n)=[];
m=abs(z);
ln=0.5;
hndl=semilogy(t,m);
set(hndl,'LineWidth',ln);
grid;
hold off;
```

7.1.3 Time domain results of tuned and untuned systems with respect to the reference model

Tuned system time domain results

The program assumes that the compensated values of the compensator were pre loaded and the system was simulated. The input to the system is a chirp signal. The output of the simulated system is a matrix y1. The output of the desired system is a matrix yd1. The output the reference input is matrix r1.

The parameters of the Simulink system were set to:

```
final time = 100 seconds;
initial time =0 seconds;
sampling rate = 0.1 seconds;
method of simulation = range Kutta 45;
The parameters of the chirp signal were set to:
initial frequency =0 Hz;
final frequency = 5Hz;
ln=1.5;
hndl=plot(y1);
set(hndl,'LineWidth',ln);
```

```
axis([0 1000 -10 10]);
xlabel('time');
ylabel('magnitude');
title('Time response of compensated and uncompensated system');
NUMC=ones(size(NUMC));
DENCC=ones(size(DENCC));
NLVEC1=[1 2 3 4 5 6 7 8 9 10 11 12 13 14 15];
NLVEC = [-(fliplr(NLVEC1)) 0 NLVEC1];
hold on;
```

Untuned time domain results

The program is a follow up of the above tuned program. It assumes that the above program was run and that the simulink diagrams were simulated so as to obtain the outputs described above. All other parameters are as described above.

```
ln=0.5;
hndl=plot(y1);
set(hndl,'LineWidth',ln);
grid;
hold off;
```

7.1.4 Results with reference to the desired model

Tuned system

```
ln=1.5;
   hndl=plot(abs(y1-yd1)+abs(y2-yd2)+abs(y3-yd3));
   sum(abs(y1-yd1)+abs(y2-yd2)+abs(y3-yd3))
   set(hndl,'LineWidth',ln);
   axis([0 1000 0 20]);
```

```
xlabel('time');
ylabel('magnitude');
NUMC=ones(size(NUMC));
DENCC=ones(size(DENCC));
NLVEC1=[1 2 3 4 5 6 7 8 9 10 11 12 13 14 15];
NLVEC = [-(fliplr(NLVEC1)) 0 NLVEC1];
hold on;
```

Untuned system

```
hold on;
   ln=0.5;
   hndl=plot(abs(y1-yd1)+abs(y2-yd2)+abs(y3-yd3));
   sum(abs(y1-yd1)+abs(y2-yd2)+abs(y3-yd3)) set(hndl,'LineWidth',ln);
   grid;
   hold off;
```

Plotting time domain signals

```
ln=1.5;
   hndl=plot(y1);
   set(hndl,'LineWidth',ln);
   hold on; ln=0.5;
   hndl=plot(yd1);
   set(hndl,'LineWidth',ln);
   grid; hold off;
   axis([0 1000 0 10]);
   xlabel('time');
   ylabel('magnitude');
   NUMC=ones(size(NUMC));
```

```
DENCC=ones(size(DENCC));
NLVEC1=[1 2 3 4 5 6 7 8 9 10 11 12 13 14 15];
NLVEC = [-(fliplr(NLVEC1)) 0 NLVEC1];
```

Plotting frequency domain signal using 3 various amplitudes and combining them to one normalized value: The case of untuned system

```
t=[0:0.01:5];
ys1=fft(y1);
ys2=fft(y2);
ys3=fft(y3);
yds1=fft(yd1);
yds2=fft(yd2);
yds3=fft(yd3);
m=[abs(abs(ys1)-abs(yds1))' + abs(abs(ys2)-abs(yds2))' +
abs(abs(ys3)-abs(yds3))'];
m(502:1001)=[];
sum(m)
ln=1.5;
hndl=semilogy(t,m);
set(hndl,'LineWidth',ln);
axis([0 5 0.00001 10000]);
xlabel('frequency in Hz');ylabel('magnitude of error');
%title('');
NUMC=ones(size( NUMC)) ;
DENCC=ones(size( DENCC));
NLVEC1=[1 2 3 4 5 6 7 8 9 10 11 12 13 14 15];
NLVEC = [-(fliplr( NLVEC1)) 0 NLVEC1];
hold on;
```

Plotting frequency domain signal using 3 various amplitudes and combining them to one normalized value: The case of tuned system

```
ys1 =(fft(y1));
   ys2 =(fft(y2));
   ys3 =(fft(y3));
   yds1=(fft(yd1));
   yds2=(fft(yd2));
   yds3=(fft(yd3));
   m=[abs(abs(ys1)-abs(yds1))' + abs(abs(ys2)-abs(yds2))' +
   abs(abs(ys3)-abs(yds3))'];
   m(502:1001)=[];
   sum(m)
   ln=0.5;
   hndl=semilogy(t,m);
   set(hndl,'LineWidth',ln);
   grid;
   hold off;
```

7.1.5 Optimization error generation

The programs bellow illustrate the generation of the error signal matrix that is used as the input to the optimizer. The following have been writen as a part of the optimization routines.

Error via transfer fuction method

Note that yproc1, yproc2, yproc3, are the outputs of the systems to 3 sets of input signals that are of the chirp type with 3 different amplitudes.

```
   yyy1 =(fft(yproc1));
```

```
yyy2 =(fft(yproc2));
yyy3 =(fft(yproc3));
%The yd1,yd2,yd3 are the desired outputs of the reference model
yds1=(fft(yd1));
yds2=(fft(yd2));
yds3=(fft(yd3));
%r1,r2,r3 are the reference signal to the desired model and the system.
%The T1,T2,T3 are the frequency transfer functions of system of amplitudes
described above.
T1=abs(yproc1)./abs(r1);
T2=abs(yproc2)./abs(r2);
T3=abs(yproc3)./abs(r3);
%Only the positive frequency components are considered.
T1(500:1001)=[];
T2(500:1001)=[];
T3(500:1001)=[];
%The error is generated
z=[abs(T1-T2)' abs(T2-T3)' abs(T3-T1)'];
err=(z);
```

Error for linearization

yproc1, yproc2, yproc3 are the outputs of the systems to 3 sets of input signals. Signals are of the chirp type with 3 different amplitudes. To normilize the signals, they are multiplied by the inverse of their multipliers. The gains of the signals yproc1, yproc2, yproc3 are 5, 2 and 0.5 respectivelly.

```
yproc1=yproc1*0.2;
yproc2=yproc2*0.5;
yproc3=yproc3*2;
```

```
z=[abs(yproc1-yproc2)' + abs(yproc2-yproc3)' +
abs(yproc1-yproc3)'];
err=(z);
```

7.1.6 Linear dynamic part optimization programs

Error in the frequency domain

```
%Note that yproc1,yproc2,yproc3 are the outputs of the systems.
   yyy1 =(fft(yproc1));
   yyy2 =(fft(yproc2));
   yyy3 =(fft(yproc3));
   %The yd1,yd2,yd3 are the desired outputs of the reference model
   yds1=(fft(yd1));
   yds2=(fft(yd2));
   yds3=(fft(yd3));
   z=[abs(abs(yyy3)-abs(yds3))' abs(abs(yyy2)-abs(yds2))'
   abs(abs(yyy1)-abs(yds1))'];
   err=(z);
```

7.1.7 Program used to simulate the nonlinear systems with dead time in the closed loop

```
function [ret,x0,str,ts,xts]=optpidvb(t,x,u,flag);
   %OPTPIDVB is the M-file description of the SIMULINK system named OPTPIDVB.
   % The block-diagram can be displayed by typing:  OPTPIDVB.
   %
   % SYS=OPTPIDVB(T,X,U,FLAG) returns depending on FLAG certain
   % system values given time point, T, current state vector, X,
   % and input vector, U.
```

```
% FLAG is used to indicate the type of output to be returned in SYS.
%
% Setting FLAG=1 causes OPTPIDVB to return state derivatives, FLAG=2
% discrete states, FLAG=3 system outputs and FLAG=4 next sample
% time.  For more information and other options see SFUNC.
%
% Calling OPTPIDVB with a FLAG of zero:
% [SIZES]=OPTPIDVB([],[],[],0), returns a vector, SIZES, which
% contains the sizes of the state vector and other parameters.
% SIZES(1) number of states
% SIZES(2) number of discrete states
% SIZES(3) number of outputs
% SIZES(4) number of inputs
% SIZES(5) number of roots (currently unsupported)
% SIZES(6) direct feedthrough flag
% SIZES(7) number of sample times
%
% For the definition of other parameters in SIZES, see SFUNC.
% See also, TRIM, LINMOD, LINSIM, EULER, RK23, RK45, ADAMS, GEAR.
% Note:  This M-file is only used for saving graphical information;
% after the model is loaded into memory an internal model
% representation is used.
% the system will take on the name of this mfile:
sys = mfilename;
new_system(sys)
simver(1.3)
if (0 == (nargin + nargout))
 set_param(sys,'Location',[4,42,791,563])
```

```
 open_system(sys)
end;
set_param(sys,'algorithm', 'RK-45')
set_param(sys,'Start time', '0.0')
set_param(sys,'Stop time', '100')
set_param(sys,'Min step size', '0.1')
set_param(sys,'Max step size', '0.1')
set_param(sys,'Relative error','1e-4')
set_param(sys,'Return vars', '')
add_block('built-in/To Workspace',[sys,'/','To Workspace9'])
set_param([sys,'/','To Workspace9'],...
 'mat-name','r3',...
 'buffer','120000',...
 'position',[225,2038,270,2062])
add_block('built-in/Sum',[sys,'/','Sum6'])
set_param([sys,'/','Sum6'],...
 'inputs','+-',...
 'position',[230,1727,255,1778])
add_block('built-in/To Workspace',[sys,'/','To Workspace6'])
set_param([sys,'/','To Workspace6'],...
 'mat-name','y3',...
 'buffer','120000',...
 'position',[930,1633,975,1657])
add_block('built-in/Note',[sys,'/','u - output of compensator3'])
set_param([sys,'/','u - output of compensator3'],...
 'position',[910,1872,915,1877])
add_block('built-in/Gain',[sys,'/','Gain2'])
set_param([sys,'/','Gain2'],...
```

```
 'Gain','4',...
 'position',[105,1718,140,1762])
add_block('built-in/Scope',[sys,'/','Scope5'])
set_param([sys,'/','Scope5'],...
 'Vgain','2.000000',...
 'Hgain','100.000000',...
 'Vmax','4.000000',...
 'Hmax','200.000000',...
 'Window',[420,156,635,305])
open_system([sys,'/','Scope5'])
set_param([sys,'/','Scope5'],...
 'position',[665,1943,695,1977])
add_block('built-in/To Workspace',[sys,'/','To Workspace8'])
set_param([sys,'/','To Workspace8'],...
 'mat-name','yd3',...
 'buffer','120000',...
 'position',[665,2013,710,2037])
add_block('built-in/Outport',[sys,'/','Outport7'])
set_param([sys,'/','Outport7'],...
 'Port','8',...
 'position',[670,1885,700,1915])
add_block('built-in/Transfer Fcn',[sys,'/','Transfer Fcn7'])
set_param([sys,'/','Transfer Fcn7'],...
 'Numerator','NUMC',...
 'Denominator','DENCC',...
 'position',[275,1730,390,1780])
add_block('built-in/Look Up Table',[sys,'/',['Look-Up',13,'Table5']])
set_param([sys,'/',['Look-Up',13,'Table5']],...
```

```
 'Input_Values','[NLI]',...
 'Output_Values','NLVEC',...
 'position',[420,1734,475,1776])
add_block('built-in/Scope',[sys,'/','Scope3'])
set_param([sys,'/','Scope3'],...
 'Vgain','2.040000',...
 'Hgain','100.000000',...
 'Vmax','4.080000',...
 'Hmax','200.000000',...
 'Window',[421,22,636,158])
open_system([sys,'/','Scope3'])
set_param([sys,'/','Scope3'],...
 'position',[925,1493,955,1527])
add_block('built-in/Note',[sys,'/',['compensator3',13,'']])
set_param([sys,'/',['compensator3',13,'']],...
 'position',[320,1695,325,1700])
add_block('built-in/Note',[sys,'/','u3'])
set_param([sys,'/','u3'],...
 'position',[655,1692,660,1697])
add_block('built-in/Outport',[sys,'/','Outport9'])
set_param([sys,'/','Outport9'],...
 'Port','4',...
 'position',[220,1300,250,1330])
add_block('built-in/To Workspace',[sys,'/','To Workspace10'])
set_param([sys,'/','To Workspace10'],...
 'mat-name','r2',...
 'buffer','120000',...
 'position',[215,1363,260,1387])
```

```
add_block('built-in/Sum',[sys,'/','Sum7'])
set_param([sys,'/','Sum7'],...
 'inputs','+-',...
 'position',[220,1052,245,1103])
add_block('built-in/To Workspace',[sys,'/','To Workspace7'])
set_param([sys,'/','To Workspace7'],...
 'mat-name','y2',...
 'buffer','120000',...
 'position',[920,958,965,982])
add_block('built-in/Note',[sys,'/','u - output of compensator4'])
set_param([sys,'/','u - output of compensator4'],...
 'position',[900,1203,905,1208])
add_block('built-in/Gain',[sys,'/','Gain3'])
set_param([sys,'/','Gain3'],...
 'Gain','4',...
 'position',[95,1043,130,1087])
add_block('built-in/Scope',[sys,'/','Scope6'])
set_param([sys,'/','Scope6'],...
 'Vgain','2.000000',...
 'Hgain','100.000000',...
 'Vmax','4.000000',...
 'Hmax','200.000000',...
 'Window',[203,158,422,305])
open_system([sys,'/','Scope6'])
set_param([sys,'/','Scope6'],...
 'position',[655,1268,685,1302])
add_block('built-in/To Workspace',[sys,'/','To Workspace11'])
set_param([sys,'/','To Workspace11'],...
```

```
 'mat-name','yd2',...
 'buffer','120000',...
 'position',[655,1338,700,1362])
add_block('built-in/Transfer Fcn',[sys,'/','Transfer Fcn8'])
set_param([sys,'/','Transfer Fcn8'],...
 'Numerator','NUMC',...
 'Denominator','DENCC',...
 'position',[265,1055,380,1105])
add_block('built-in/Look Up Table',[sys,'/',['Look-Up',13,'Table7']])
set_param([sys,'/',['Look-Up',13,'Table7']],...
 'Input_Values','[NLI]',...
 'Output_Values','NLVEC',...
 'position',[410,1059,465,1101])
add_block('built-in/Scope',[sys,'/','Scope4'])
set_param([sys,'/','Scope4'],...
 'Vgain','2.040000',...
 'Hgain','100.000000',...
 'Vmax','4.080000',...
 'Hmax','200.000000',...
 'Window',[204,20,422,159])
open_system([sys,'/','Scope4'])
set_param([sys,'/','Scope4'],...
 'position',[915,818,945,852])
add_block('built-in/Note',[sys,'/','u2'])
set_param([sys,'/','u2'],...
 'position',[645,1023,650,1028])
add_block('built-in/Note',[sys,'/',['compensator2',13,'']])
set_param([sys,'/',['compensator2',13,'']],...
```

```
 'position',[325,996,330,1001])
add_block('built-in/Outport',[sys,'/','Outport8'])
set_param([sys,'/','Outport8'],...
 'Port','7',...
 'position',[235,1975,265,2005])
add_block('built-in/Look Up Table',[sys,'/',['Look-Up',13,'Table6']])
set_param([sys,'/',['Look-Up',13,'Table6']],...
 'Input_Values','nlin',...
 'Output_Values','nlout',...
 'position',[525,1059,600,1101])
add_block('built-in/Outport',[sys,'/','Outport5'])
set_param([sys,'/','Outport5'],...
 'Port','6',...
 'position',[1010,900,1040,930])
add_block('built-in/Saturation',[sys,'/','Saturation2'])
set_param([sys,'/','Saturation2'],...
 'Lower Limit','-0.5*10^100',...
 'Upper Limit','0.5*10^100',...
 'position',[945,1563,975,1587])
add_block('built-in/Outport',[sys,'/','Outport4'])
set_param([sys,'/','Outport4'],...
 'Port','9',...
 'position',[1020,1560,1050,1590])
add_block('built-in/Saturation',[sys,'/','Saturation1'])
set_param([sys,'/','Saturation1'],...
 'Lower Limit','-2*10^100',...
 'Upper Limit','2*10^100',...
 'position',[930,903,960,927])
```

```
add_block('built-in/Look Up Table',[sys,'/',['Look-Up',13,'Table4']])
set_param([sys,'/',['Look-Up',13,'Table4']],...
 'Input_Values','nlin',...
 'Output_Values','nlout',...
 'position',[530,1734,605,1776])
add_block('built-in/Transfer Fcn',[sys,'/','Transfer Fcn6'])
set_param([sys,'/','Transfer Fcn6'],...
 'Numerator','NUM',...
 'Denominator','DENOM',...
 'position',[655,1730,770,1780])
add_block('built-in/Transport Delay',[sys,'/','L2'])
set_param([sys,'/','L2'],...
 'Delay Time','L',...
 'Buffer Size:','2048',...
 'position',[815,1737,880,1773])
add_block('built-in/Transfer Fcn',[sys,'/','Transfer Fcn11'])
set_param([sys,'/','Transfer Fcn11'],...
 'Numerator','NUM',...
 'Denominator','DENOM',...
 'position',[645,1055,760,1105])
add_block('built-in/Transport Delay',[sys,'/','L3'])
set_param([sys,'/','L3'],...
 'Delay Time','L',...
 'Buffer Size:','2048',...
 'position',[805,1062,870,1098])
add_block('built-in/Transfer Fcn',[sys,'/','Transfer Fcn10'])
set_param([sys,'/','Transfer Fcn10'],...
 'Numerator','[NUMD]',...
```

```
 'Denominator','[DENOMD]',...
 'position',[255,1260,370,1310])
add_block('built-in/Transport Delay',[sys,'/','L6'])
set_param([sys,'/','L6'],...
 'Delay Time','L',...
 'Buffer Size:','2048',...
 'position',[415,1267,480,1303])
add_block('built-in/Outport',[sys,'/','Outport10'])
set_param([sys,'/','Outport10'],...
 'Port','5',...
 'position',[655,1210,685,1240])
add_block('built-in/Transfer Fcn',[sys,'/','Transfer Fcn9'])
set_param([sys,'/','Transfer Fcn9'],...
 'Numerator','[NUMD]',...
 'Denominator','[DENOMD]',...
 'position',[315,1935,430,1985])
add_block('built-in/Transport Delay',[sys,'/','L7'])
set_param([sys,'/','L7'],...
 'Delay Time','L',...
 'Buffer Size:','2048',...
 'position',[485,1942,550,1978])
% Subsystem 'Chirp Signal1'.
new_system([sys,'/','Chirp Signal1'])
set_param([sys,'/','Chirp Signal1'],'Location',[466,243,956,594])
add_block('built-in/Clock',[sys,'/','Chirp Signal1/Clock'])
set_param([sys,'/','Chirp Signal1/Clock'],...
 'position',[50,75,70,95])
add_block('built-in/Fcn',[sys,'/','Chirp Signal1/Fcn'])
```

```
   set_param([sys,'/','Chirp Signal1/Fcn'],...
    'Expr','sin(AA*u*u + BB*u)',...
    'position',[95,60,160,110])
   add_block('built-in/Outport',[sys,'/','Chirp Signal1/out_1'])
   set_param([sys,'/','Chirp Signal1/out_1'],...
    'position',[190,75,210,95])
   add_line([sys,'/','Chirp Signal1'],[75,85;90,85])
   add_line([sys,'/','Chirp Signal1'],[165,85;185,85])
   set_param([sys,'/','Chirp Signal1'],...
    'Mask Display','plot(t,sin(t.*t))',...
    'Mask Type','chirp')
   set_param([sys,'/','Chirp Signal1'],...
    'Mask Dialogue','Chirp Signal.\n(Sine wave with increasing
   frequency)|Initial frequency (Hz):| Target time (secs):| Frequency at target
time (Hz):')
   set_param([sys,'/','Chirp Signal1'],...
    'Mask Translate','BB = 2*pi*@1; AA =pi*(@3 -@1)/(@2);t=[0:.1:5];')
   set_param([sys,'/','Chirp Signal1'],...
    'Mask Help','The chirp signal has linearly increasing frequency with time.
This block can be used for spectral analysis of nonlinear systems.  ',...
    'Mask Entries','0\/100\/5\/')
   % Finished composite block 'Chirp Signal1'.
   set_param([sys,'/','Chirp Signal1'],...
    'position',[35,1722,75,1758])
   % Subsystem 'Chirp Signal'.
   new_system([sys,'/','Chirp Signal'])
   set_param([sys,'/','Chirp Signal'],'Location',[466,243,956,594])
   add_block('built-in/Clock',[sys,'/','Chirp Signal/Clock'])
```

```
   set_param([sys,'/','Chirp Signal/Clock'],...
    'position',[50,75,70,95])
   add_block('built-in/Fcn',[sys,'/','Chirp Signal/Fcn'])
   set_param([sys,'/','Chirp Signal/Fcn'],...
    'Expr','sin(AA*u*u + BB*u)',...
    'position',[95,60,160,110])
   add_block('built-in/Outport',[sys,'/','Chirp Signal/out_1'])
   set_param([sys,'/','Chirp Signal/out_1'],...
    'position',[190,75,210,95])
   add_line([sys,'/','Chirp Signal'],[75,85;90,85])
   add_line([sys,'/','Chirp Signal'],[165,85;185,85])
   set_param([sys,'/','Chirp Signal'],...
    'Mask Display','plot(t,sin(t.*t))',...
    'Mask Type','chirp')
   set_param([sys,'/','Chirp Signal'],...
    'Mask Dialogue','Chirp Signal.\n(Sine wave with increasing
   frequency)|Initial frequency (Hz):| Target time (secs):| Frequency at target
time (Hz):')
   set_param([sys,'/','Chirp Signal'],...
    'Mask Translate','BB = 2*pi*@1; AA =pi*(@3 -@1)/(@2);t=[0:.1:5];')
   set_param([sys,'/','Chirp Signal'],...
    'Mask Help','The chirp signal has linearly increasing frequency with time.
This block can be used for spectral analysis of nonlinear systems.  ',...
    'Mask Entries','0\/100\/0.1\/')
   % Finished composite block 'Chirp Signal'.
   set_param([sys,'/','Chirp Signal'],...
    'position',[10,1047,45,1083])
   add_block('built-in/Sum',[sys,'/','Sum8'])
```

```
set_param([sys,'/','Sum8'],...
 'inputs','+-',...
 'position',[145,137,170,188])
add_block('built-in/Gain',[sys,'/','Gain4'])
set_param([sys,'/','Gain4'],...
 'Gain','5',...
 'position',[80,128,115,172])
add_block('built-in/Transfer Fcn',[sys,'/',['Transfer',13,' Fcn13']])
set_param([sys,'/',['Transfer',13,' Fcn13']],...
 'Numerator','NUMC',...
 'Denominator','DENCC',...
 'position',[190,141,240,189])
add_block('built-in/Note',[sys,'/','u - output of compensator5'])
set_param([sys,'/','u - output of compensator5'],...
 'position',[430,26,435,31])
add_block('built-in/Outport',[sys,'/','Outport6'])
set_param([sys,'/','Outport6'],...
 'Port','3',...
 'position',[645,60,675,90])
add_block('built-in/To Workspace',[sys,'/',' To Workspace13'])
set_param([sys,'/',' To Workspace13'],...
 'mat-name','y1',...
 'buffer','120000',...
 'position',[560,113,605,137])
add_block('built-in/Transport Delay',[sys,'/','L4'])
set_param([sys,'/','L4'],...
 'Delay Time','L',...
 'Buffer Size:','2048',...
```

```
 'position',[500,145,540,185])
add_block('built-in/Saturation',[sys,'/',' Saturation'])
set_param([sys,'/',' Saturation'],...
 'Lower Limit','-5*10^100',...
 'Upper Limit','5*10^100',...
 'position',[560,63,590,87])
add_block('built-in/Look Up Table',[sys,'/',['Look-Up',13,'Table9']])
set_param([sys,'/',['Look-Up',13,'Table9']],...
 'Input_Values','[NLI]',...
 'Output_Values','NLVEC',...
 'position',[270,144,315,186])
add_block('built-in/Step Fcn',[sys,'/','Step Input'])
set_param([sys,'/','Step Input'],...
 'Time','0',...
 'position',[10,132,50,168])
add_block('built-in/Note',[sys,'/',['compensator1',13,'']])
set_param([sys,'/',['compensator1',13,'']],...
 'position',[250,99,255,104])
add_block('built-in/Transfer Fcn',[sys,'/',['Transfer',13,' Fcn14']])
set_param([sys,'/',['Transfer',13,' Fcn14']],...
 'Numerator','NUM',...
 'Denominator','DENOM',...
 'position',[430,139,480,191])
add_block('built-in/Look Up Table',[sys,'/',['Look-Up',13,'Table8']])
set_param([sys,'/',['Look-Up',13,'Table8']],...
 'Input_Values','nlin',...
 'Output_Values','nlout',...
 'position',[365,144,405,186])
```

```
add_block('built-in/Note',[sys,'/',['System',13,'']])
set_param([sys,'/',['System',13,'']],...
 'position',[445,91,450,96])
add_block('built-in/Outport',[sys,'/','Outport11'])
set_param([sys,'/','Outport11'],...
 'position',[135,345,165,375])
add_block('built-in/Outport',[sys,'/',' Outport12'])
set_param([sys,'/',' Outport12'],...
 'Port','2',...
 'position',[495,260,525,290])
add_block('built-in/To Workspace',[sys,'/','To Workspace14'])
set_param([sys,'/','To Workspace14'],...
 'mat-name','yd1',...
 'buffer','120000',...
 'position',[495,368,540,392])
add_block('built-in/Scope',[sys,'/','Scope7'])
set_param([sys,'/','Scope7'],...
 'Vgain','5.000000',...
 'Hgain','100.000000',...
 'Vmax','10.000000',...
 'Hmax','200.000000',...
 'Window',[3,157,201,305])
open_system([sys,'/','Scope7'])
set_param([sys,'/','Scope7'],...
 'position',[505,313,535,347])
add_block('built-in/Transport Delay',[sys,'/','L5'])
set_param([sys,'/','L5'],...
 'Delay Time','L',...
```

```
 'Buffer Size:','2048',...
 'position',[395,312,460,348])
add_block('built-in/Transfer Fcn',[sys,'/','Transfer Fcn12'])
set_param([sys,'/','Transfer Fcn12'],...
 'Numerator','[NUMD]',...
 'Denominator','[DENOMD]',...
 'position',[235,305,350,355])
add_block('built-in/Scope',[sys,'/','Scope8'])
set_param([sys,'/','Scope8'],...
 'Vgain','5.000000',...
 'Hgain','100.000000',...
 'Vmax','10.000000',...
 'Hmax','200.000000',...
 'Window',[4,20,202,161])
open_system([sys,'/','Scope8'])
set_param([sys,'/','Scope8'],...
 'position',[560,8,590,42])
add_block('built-in/To Workspace',[sys,'/','To Workspace12'])
set_param([sys,'/','To Workspace12'],...
 'mat-name','r1',...
 'buffer','120000',...
 'position',[190,253,235,277])
add_block('built-in/Note',[sys,'/',['r - reference',13,' input']])
set_param([sys,'/',['r - reference',13,' input']],...
 'position',[60,76,65,81])
add_line(sys,[885,1755;885,1645;925,1645])
add_line(sys,[885,1755;960,1755;960,1850;215,1850;225,1765])
add_line(sys,[260,1755;270,1755])
```

```
add_line(sys,[145,1740;225,1740])
add_line(sys,[395,1755;415,1755])
add_line(sys,[145,1740;145,1960;310,1960])
add_line(sys,[145,1740;145,2050;220,2050])
add_line(sys,[480,1755;525,1755])
add_line(sys,[145,1740;145,1990;230,1990])
add_line(sys,[885,1755;885,1510;920,1510])
add_line(sys,[875,1080;875,970;915,970])
add_line(sys,[875,1080;950,1080;950,1181;205,1181;215,1090])
add_line(sys,[250,1080;260,1080])
add_line(sys,[135,1065;215,1065])
add_line(sys,[385,1080;405,1080])
add_line(sys,[135,1065;135,1285;250,1285])
add_line(sys,[135,1065;135,1375;210,1375])
add_line(sys,[470,1080;520,1080])
add_line(sys,[135,1065;135,1315;215,1315])
add_line(sys,[875,1080;875,835;910,835])
add_line(sys,[50,1065;90,1065])
add_line(sys,[605,1080;640,1080])
add_line(sys,[765,1080;800,1080])
add_line(sys,[610,1755;650,1755])
add_line(sys,[775,1755;810,1755])
add_line(sys,[885,1755;885,1575;940,1575])
add_line(sys,[980,1575;1015,1575])
add_line(sys,[875,1080;875,915;925,915])
add_line(sys,[965,915;1005,915])
add_line(sys,[375,1285;410,1285])
add_line(sys,[485,1285;565,1285;565,1225;650,1225])
```

```
add_line(sys,[485,1285;560,1285;560,1350;650,1350])
add_line(sys,[485,1285;650,1285])
add_line(sys,[435,1960;480,1960])
add_line(sys,[555,1960;605,1960;605,1900;665,1900])
add_line(sys,[555,1960;600,1960;600,2025;660,2025])
add_line(sys,[555,1960;660,1960])
add_line(sys,[80,1740;100,1740])
add_line(sys,[545,165;555,125])
add_line(sys,[545,165;545,235;130,235;140,175])
add_line(sys,[175,165;185,165])
add_line(sys,[120,150;140,150])
add_line(sys,[245,165;265,165])
add_line(sys,[120,150;120,330;230,330])
add_line(sys,[120,150;120,265;185,265])
add_line(sys,[320,165;360,165])
add_line(sys,[120,150;130,360])
add_line(sys,[545,165;555,25])
add_line(sys,[485,165;495,165])
add_line(sys,[410,165;425,165])
add_line(sys,[545,165;555,75])
add_line(sys,[595,75;640,75])
add_line(sys,[355,330;390,330])
add_line(sys,[465,330;480,330;490,275])
add_line(sys,[465,330;480,330;490,380])
add_line(sys,[465,330;500,330])
add_line(sys,[55,150;75,150])
drawnow
% Return any arguments.
```

```
if (nargin | nargout)
 % Must use feval here to access system in memory
 if (nargin > 3)
 if (flag == 0)
 eval(['[ret,x0,str,ts,xts]=',sys,'(t,x,u,flag);'])
 else
 eval(['ret =', sys,'(t,x,u,flag);'])
 end
 else
 [ret,x0,str,ts,xts] = feval(sys);
 end
else
 drawnow % Flash up the model and execute load callback
end
```

7.1.8 Program used in simulatión of nonlinear system with dead time in the open loop using a chirp signal input. This program was used in the linearization process.

```
function [ret,x0,str,ts,xts]=optpidvb(t,x,u,flag);
  %OPTPIDVB is the M-file description of the SIMULINK system named OPTPIDVB.
  % The block-diagram can be displayed by typing:  OPTPIDVB.
  %
  % SYS=OPTPIDVB(T,X,U,FLAG) returns depending on FLAG certain
  % system values given time point, T, current state vector, X,
  % and input vector, U.
  % FLAG is used to indicate the type of output to be returned in SYS.
  %
```

```
% Setting FLAG=1 causes OPTPIDVB to return state derivatives, FLAG=2
% discrete states, FLAG=3 system outputs and FLAG=4 next sample
% time.  For more information and other options see SFUNC.
%
% Calling OPTPIDVB with a FLAG of zero:
% [SIZES]=OPTPIDVB([],[],[],0), returns a vector, SIZES, which
% contains the sizes of the state vector and other parameters.
% SIZES(1) number of states
% SIZES(2) number of discrete states
% SIZES(3) number of outputs
% SIZES(4) number of inputs
% SIZES(5) number of roots (currently unsupported)
% SIZES(6) direct feedthrough flag
% SIZES(7) number of sample times
%
% For the definition of other parameters in SIZES, see SFUNC.
% See also, TRIM, LINMOD, LINSIM, EULER, RK23, RK45, ADAMS, GEAR.
% Note:  This M-file is only used for saving graphical information;
% after the model is loaded into memory an internal model
% representation is used.
% the system will take on the name of this mfile:
sys = mfilename;
new_system(sys)
simver(1.3)
if (0 == (nargin + nargout))
 set_param(sys,'Location',[428,362,639,486])
 open_system(sys)
end;
```

```
set_param(sys,'algorithm', 'RK-45')
set_param(sys,'Start time', '0.0')
set_param(sys,'Stop time', '100')
set_param(sys,'Min step size', '0.1')
set_param(sys,'Max step size', '0.1')
set_param(sys,'Relative error','1e-4')
set_param(sys,'Return vars', '')
add_block('built-in/To Workspace',[sys,'/','To Workspace9'])
set_param([sys,'/','To Workspace9'],...
 'mat-name','r3',...
 'buffer','120000',...
 'position',[225,2038,270,2062])
add_block('built-in/Sum',[sys,'/','Sum6'])
set_param([sys,'/','Sum6'],...
 'inputs','+-',...
 'position',[230,1727,255,1778])
add_block('built-in/To Workspace',[sys,'/','To Workspace6'])
set_param([sys,'/','To Workspace6'],...
 'mat-name','y3',...
 'buffer','120000',...
 'position',[930,1633,975,1657])
add_block('built-in/Note',[sys,'/','u - output of compensator3'])
set_param([sys,'/','u - output of compensator3'],...
 'position',[910,1872,915,1877])
add_block('built-in/Gain',[sys,'/','Gain2'])
set_param([sys,'/','Gain2'],...
 'Gain','0.5',...
 'position',[105,1718,140,1762])
```

```
add_block('built-in/Scope',[sys,'/','Scope5'])
set_param([sys,'/','Scope5'],...
 'Vgain','2.000000',...
 'Hgain','100.000000',...
 'Vmax','4.000000',...
 'Hmax','200.000000',...
 'Window',[420,156,635,305])
open_system([sys,'/','Scope5'])
set_param([sys,'/','Scope5'],...
 'position',[665,1943,695,1977])
add_block('built-in/To Workspace',[sys,'/','To Workspace8'])
set_param([sys,'/','To Workspace8'],...
 'mat-name','yd3',...
 'buffer','120000',...
 'position',[665,2013,710,2037])
add_block('built-in/Outport',[sys,'/','Outport7'])
set_param([sys,'/','Outport7'],...
 'Port','8',...
 'position',[670,1885,700,1915])
add_block('built-in/Transfer Fcn',[sys,'/','Transfer Fcn7'])
set_param([sys,'/','Transfer Fcn7'],...
 'Numerator','1',...
 'Denominator','1',...
 'position',[275,1730,390,1780])
add_block('built-in/Look Up Table',[sys,'/',['Look-Up',13,'Table5']])
set_param([sys,'/',['Look-Up',13,'Table5']],...
 'Input_Values','[NLI]',...
 'Output_Values','NLVEC',...
```

```
 'position',[420,1734,475,1776])
add_block('built-in/Scope',[sys,'/','Scope3'])
set_param([sys,'/','Scope3'],...
 'Vgain','2.040000',...
 'Hgain','100.000000',...
 'Vmax','4.080000',...
 'Hmax','200.000000',...
 'Window',[421,22,636,158])
open_system([sys,'/','Scope3'])
set_param([sys,'/','Scope3'],...
 'position',[925,1493,955,1527])
add_block('built-in/Note',[sys,'/',['compensator3',13,'']])
set_param([sys,'/',['compensator3',13,'']],...
 'position',[320,1695,325,1700])
add_block('built-in/Note',[sys,'/','u3'])
set_param([sys,'/','u3'],...
 'position',[655,1692,660,1697])
add_block('built-in/Outport',[sys,'/','Outport9'])
set_param([sys,'/','Outport9'],...
 'Port','4',...
 'position',[220,1300,250,1330])
add_block('built-in/To Workspace',[sys,'/','To Workspace10'])
set_param([sys,'/','To Workspace10'],...
 'mat-name','r2',...
 'buffer','120000',...
 'position',[215,1363,260,1387])
add_block('built-in/Sum',[sys,'/','Sum7'])
set_param([sys,'/','Sum7'],...
```

```
 'inputs','+-',...
 'position',[220,1052,245,1103])
add_block('built-in/To Workspace',[sys,'/','To Workspace7'])
set_param([sys,'/','To Workspace7'],...
 'mat-name','y2',...
 'buffer','120000',...
 'position',[920,958,965,982])
add_block('built-in/Note',[sys,'/','u - output of compensator4'])
set_param([sys,'/','u - output of compensator4'],...
 'position',[900,1203,905,1208])
add_block('built-in/Gain',[sys,'/','Gain3'])
set_param([sys,'/','Gain3'],...
 'Gain','5',...
 'position',[95,1043,130,1087])
add_block('built-in/Scope',[sys,'/','Scope6'])
set_param([sys,'/','Scope6'],...
 'Vgain','2.000000',...
 'Hgain','100.000000',...
 'Vmax','4.000000',...
 'Hmax','200.000000',...
 'Window',[203,158,422,305])
open_system([sys,'/','Scope6'])
set_param([sys,'/','Scope6'],...
 'position',[655,1268,685,1302])
add_block('built-in/To Workspace',[sys,'/','To Workspace11'])
set_param([sys,'/','To Workspace11'],...
 'mat-name','yd2',...
 'buffer','120000',...
```

```
 'position',[655,1338,700,1362])
add_block('built-in/Transfer Fcn',[sys,'/','Transfer Fcn8'])
set_param([sys,'/','Transfer Fcn8'],...
 'Numerator','1',...
 'Denominator','1',...
 'position',[265,1055,380,1105])
add_block('built-in/Look Up Table',[sys,'/',['Look-Up',13,'Table7']])
set_param([sys,'/',['Look-Up',13,'Table7']],...
 'Input_Values','[NLI]',...
 'Output_Values','NLVEC',...
 'position',[410,1059,465,1101])
add_block('built-in/Scope',[sys,'/','Scope4'])
set_param([sys,'/','Scope4'],...
 'Vgain','2.040000',...
 'Hgain','100.000000',...
 'Vmax','4.080000',...
 'Hmax','200.000000',...
 'Window',[204,20,422,159])
open_system([sys,'/','Scope4'])
set_param([sys,'/','Scope4'],...
 'position',[915,818,945,852])
add_block('built-in/Note',[sys,'/','u2'])
set_param([sys,'/','u2'],...
 'position',[645,1023,650,1028])
add_block('built-in/Note',[sys,'/',['compensator2',13,'']])
set_param([sys,'/',['compensator2',13,'']],...
 'position',[325,996,330,1001])
add_block('built-in/Outport',[sys,'/','Outport8'])
```

```
set_param([sys,'/','Outport8'],...
 'Port','7',...
 'position',[235,1975,265,2005])
add_block('built-in/Look Up Table',[sys,'/',['Look-Up',13,'Table6']])
set_param([sys,'/',['Look-Up',13,'Table6']],...
 'Input_Values','nlin',...
 'Output_Values','nlout',...
 'position',[525,1059,600,1101])
add_block('built-in/Outport',[sys,'/','Outport5'])
set_param([sys,'/','Outport5'],...
 'Port','6',...
 'position',[1010,900,1040,930])
add_block('built-in/Saturation',[sys,'/','Saturation2'])
set_param([sys,'/','Saturation2'],...
 'Lower Limit','-0.5*10^100',...
 'Upper Limit','0.5*10^100',...
 'position',[945,1563,975,1587])
add_block('built-in/Outport',[sys,'/','Outport4'])
set_param([sys,'/','Outport4'],...
 'Port','9',...
 'position',[1020,1560,1050,1590])
add_block('built-in/Saturation',[sys,'/','Saturation1'])
set_param([sys,'/','Saturation1'],...
 'Lower Limit','-2*10^100',...
 'Upper Limit','2*10^100',...
 'position',[930,903,960,927])
add_block('built-in/Look Up Table',[sys,'/',['Look-Up',13,'Table4']])
set_param([sys,'/',['Look-Up',13,'Table4']],...
```

```
 'Input_Values','nlin',...
 'Output_Values','nlout',...
 'position',[530,1734,605,1776])
add_block('built-in/Transfer Fcn',[sys,'/','Transfer Fcn6'])
set_param([sys,'/','Transfer Fcn6'],...
 'Numerator','NUM',...
 'Denominator','DENOM',...
 'position',[655,1730,770,1780])
add_block('built-in/Transport Delay',[sys,'/','L2'])
set_param([sys,'/','L2'],...
 'Delay Time','L',...
 'Buffer Size:','2048',...
 'position',[815,1737,880,1773])
add_block('built-in/Transfer Fcn',[sys,'/','Transfer Fcn11'])
set_param([sys,'/','Transfer Fcn11'],...
 'Numerator','NUM',...
 'Denominator','DENOM',...
 'position',[645,1055,760,1105])
add_block('built-in/Transport Delay',[sys,'/','L3'])
set_param([sys,'/','L3'],...
 'Delay Time','L',...
 'Buffer Size:','2048',...
 'position',[805,1062,870,1098])
add_block('built-in/Transfer Fcn',[sys,'/','Transfer Fcn10'])
set_param([sys,'/','Transfer Fcn10'],...
 'Numerator','[NUMD]',...
 'Denominator','[DENOMD]',...
 'position',[255,1260,370,1310])
```

```
add_block('built-in/Transport Delay',[sys,'/','L6'])
set_param([sys,'/','L6'],...
 'Delay Time','L',...
 'Buffer Size:','2048',...
 'position',[415,1267,480,1303])
add_block('built-in/Outport',[sys,'/','Outport10'])
set_param([sys,'/','Outport10'],...
 'Port','5',...
 'position',[655,1210,685,1240])
add_block('built-in/Transfer Fcn',[sys,'/','Transfer Fcn9'])
set_param([sys,'/','Transfer Fcn9'],...
 'Numerator','[NUMD]',...
 'Denominator','[DENOMD]',...
 'position',[315,1935,430,1985])
add_block('built-in/Transport Delay',[sys,'/','L7'])
set_param([sys,'/','L7'],...
 'Delay Time','L',...
 'Buffer Size:','2048',...
 'position',[485,1942,550,1978])
add_block('built-in/Sum',[sys,'/','Sum8'])
set_param([sys,'/','Sum8'],...
 'inputs','+-',...
 'position',[145,137,170,188])
add_block('built-in/Transfer Fcn',[sys,'/','Transfer Fcn12'])
set_param([sys,'/','Transfer Fcn12'],...
 'Numerator','[NUMD]',...
 'Denominator','[DENOMD]',...
 'position',[235,275,350,325])
```

```
add_block('built-in/Transport Delay',[sys,'/','L5'])
set_param([sys,'/','L5'],...
 'Delay Time','L',...
 'Buffer Size:','2048',...
 'position',[395,282,460,318])
add_block('built-in/Scope',[sys,'/','Scope7'])
set_param([sys,'/','Scope7'],...
 'Vgain','5.000000',...
 'Hgain','100.000000',...
 'Vmax','10.000000',...
 'Hmax','200.000000',...
 'Window',[3,157,201,305])
open_system([sys,'/','Scope7'])
set_param([sys,'/','Scope7'],...
 'position',[505,283,535,317])
add_block('built-in/To Workspace',[sys,'/','To Workspace14'])
set_param([sys,'/','To Workspace14'],...
 'mat-name','yd1',...
 'buffer','120000',...
 'position',[495,338,540,362])
add_block('built-in/Outport',[sys,'/','Outport12'])
set_param([sys,'/','Outport12'],...
 'Port','2',...
 'position',[495,230,525,260])
add_block('built-in/Outport',[sys,'/','Outport11'])
set_param([sys,'/','Outport11'],...
 'position',[135,315,165,345])
add_block('built-in/To Workspace',[sys,'/','To Workspace12'])
```

```
set_param([sys,'/','To Workspace12'],...
 'mat-name','r1',...
 'buffer','120000',...
 'position',[145,248,190,272])
add_block('built-in/Note',[sys,'/','r - reference input'])
set_param([sys,'/','r - reference input'],...
 'position',[80,111,85,116])
add_block('built-in/Transfer Fcn',[sys,'/',';'])
set_param([sys,'/',';'],...
 'Numerator','1',...
 'Denominator','1',...
 'position',[190,141,240,189])
add_block('built-in/Note',[sys,'/',['compensator1',13,'']])
set_param([sys,'/',['compensator1',13,'']],...
 'position',[250,114,255,119])
add_block('built-in/Note',[sys,'/',['System',13,'']])
set_param([sys,'/',['System',13,'']],...
 'position',[395,116,400,121])
add_block('built-in/Look Up Table',[sys,'/',['Look-Up',13,'Table9']])
set_param([sys,'/',['Look-Up',13,'Table9']],...
 'Input_Values','[NLI]',...
 'Output_Values','NLVEC',...
 'position',[255,144,300,186])
add_block('built-in/Look Up Table',[sys,'/',['Look-Up',13,'Table8']])
set_param([sys,'/',['Look-Up',13,'Table8']],...
 'Input_Values','nlin',...
 'Output_Values','nlout',...
 'position',[325,144,365,186])
```

```
add_block('built-in/Transfer Fcn',[sys,'/','Transfer Fcn14'])
set_param([sys,'/','Transfer Fcn14'],...
 'Numerator','NUM',...
 'Denominator','DENOM',...
 'position',[385,139,435,191])
add_block('built-in/Transport Delay',[sys,'/','L4'])
set_param([sys,'/','L4'],...
 'Delay Time','L',...
 'Buffer Size:','2048',...
 'position',[450,145,490,185])
add_block('built-in/Saturation',[sys,'/','Saturation'])
set_param([sys,'/','Saturation'],...
 'Lower Limit','-5*10^100',...
 'Upper Limit','5*10^100',...
 'position',[510,63,540,87])
add_block('built-in/Outport',[sys,'/','Outport6'])
set_param([sys,'/','Outport6'],...
 'Port','3',...
 'position',[560,60,590,90])
add_block('built-in/To Workspace',[sys,'/','To Workspace13'])
set_param([sys,'/','To Workspace13'],...
 'mat-name','y1',...
 'buffer','120000',...
 'position',[510,113,555,137])
add_block('built-in/Scope',[sys,'/','Scope8'])
set_param([sys,'/','Scope8'],...
 'Vgain','5.000000',...
 'Hgain','100.000000',...
```

```
 'Vmax','10.000000',...
 'Hmax','200.000000',...
 'Window',[4,20,202,161])
open_system([sys,'/','Scope8'])
set_param([sys,'/','Scope8'],...
 'position',[510,13,540,47])
add_block('built-in/Note',[sys,'/','u1'])
set_param([sys,'/','u1'],...
 'position',[570,36,575,41])
add_block('built-in/Note',[sys,'/','u - output of compensator5'])
set_param([sys,'/','u - output of compensator5'],...
 'position',[430,26,435,31])
% Subsystem 'Chirp Signal2'.
new_system([sys,'/','Chirp Signal2'])
set_param([sys,'/','Chirp Signal2'],'Location',[466,243,956,594])
add_block('built-in/Clock',[sys,'/','Chirp Signal2/Clock'])
set_param([sys,'/','Chirp Signal2/Clock'],...
 'position',[50,75,70,95])
add_block('built-in/Fcn',[sys,'/','Chirp Signal2/Fcn'])
set_param([sys,'/','Chirp Signal2/Fcn'],...
 'Expr','sin(AA*u*u + BB*u)',...
 'position',[95,60,160,110])
add_block('built-in/Outport',[sys,'/','Chirp Signal2/out_1'])
set_param([sys,'/','Chirp Signal2/out_1'],...
 'position',[190,75,210,95])
add_line([sys,'/','Chirp Signal2'],[75,85;90,85])
add_line([sys,'/','Chirp Signal2'],[165,85;185,85])
set_param([sys,'/','Chirp Signal2'],...
```

```
    'Mask Display','plot(t,sin(t.*t))',...
    'Mask Type','chirp')
  set_param([sys,'/','Chirp Signal2'],...
    'Mask Dialogue','Chirp Signal.\n(Sine wave with increasing
  frequency)|Initial frequency (Hz):| Target time (secs):| Frequency at target
time (Hz):')
  set_param([sys,'/','Chirp Signal2'],...
    'Mask Translate','BB = 2*pi*@1; AA =pi*(@3 -@1)/(@2);t=[0:.1:5];')
  set_param([sys,'/','Chirp Signal2'],...
    'Mask Help','The chirp signal has linearly increasing frequency with time.
This block can be used for spectral analysis of nonlinear systems.  ',...
    'Mask Entries','0\/100\/1\/')
  % Finished composite block 'Chirp Signal2'.
  set_param([sys,'/','Chirp Signal2'],...
    'position',[0,127,30,163])
  % Subsystem 'Chirp Signal'.
  new_system([sys,'/','Chirp Signal'])
  set_param([sys,'/','Chirp Signal'],'Location',[466,243,956,594])
  add_block('built-in/Clock',[sys,'/','Chirp Signal/Clock'])
  set_param([sys,'/','Chirp Signal/Clock'],...
    'position',[50,75,70,95])
  add_block('built-in/Fcn',[sys,'/','Chirp Signal/Fcn'])
  set_param([sys,'/','Chirp Signal/Fcn'],...
    'Expr','sin(AA*u*u + BB*u)',...
    'position',[95,60,160,110])
  add_block('built-in/Outport',[sys,'/','Chirp Signal/out_1'])
  set_param([sys,'/','Chirp Signal/out_1'],...
    'position',[190,75,210,95])
```

```
   add_line([sys,'/','Chirp Signal'],[75,85;90,85])
   add_line([sys,'/','Chirp Signal'],[165,85;185,85])
   set_param([sys,'/','Chirp Signal'],...
    'Mask Display','plot(t,sin(t.*t))',...
    'Mask Type','chirp')
   set_param([sys,'/','Chirp Signal'],...
    'Mask Dialogue','Chirp Signal.\n(Sine wave with increasing
   frequency)|Initial frequency (Hz):| Target time (secs):| Frequency at target
time (Hz):')
   set_param([sys,'/','Chirp Signal'],...
    'Mask Translate','BB = 2*pi*@1; AA =pi*(@3 -@1)/(@2);t=[0:.1:5];')
   set_param([sys,'/','Chirp Signal'],...
    'Mask Help','The chirp signal has linearly increasing frequency with time.
This block can be used for spectral analysis of nonlinear systems.  ',...
    'Mask Entries','0\/100\/1\/')
   % Finished composite block 'Chirp Signal'.
   set_param([sys,'/','Chirp Signal'],...
    'position',[0,1047,30,1083])
   % Subsystem 'Chirp Signal1'.
   new_system([sys,'/','Chirp Signal1'])
   set_param([sys,'/','Chirp Signal1'],'Location',[466,243,956,594])
   add_block('built-in/Clock',[sys,'/','Chirp Signal1/Clock'])
   set_param([sys,'/','Chirp Signal1/Clock'],...
    'position',[50,75,70,95])
   add_block('built-in/Fcn',[sys,'/','Chirp Signal1/Fcn'])
   set_param([sys,'/','Chirp Signal1/Fcn'],...
    'Expr','sin(AA*u*u + BB*u)',...
    'position',[95,60,160,110])
```

```
add_block('built-in/Outport',[sys,'/','Chirp Signal1/out_1'])
set_param([sys,'/','Chirp Signal1/out_1'],...
 'position',[190,75,210,95])
add_line([sys,'/','Chirp Signal1'],[75,85;90,85])
add_line([sys,'/','Chirp Signal1'],[165,85;185,85])
set_param([sys,'/','Chirp Signal1'],...
 'Mask Display','plot(t,sin(t.*t))',...
 'Mask Type','chirp')
set_param([sys,'/','Chirp Signal1'],...
 'Mask Dialogue','Chirp Signal.\n(Sine wave with increasing
frequency)|Initial frequency (Hz):| Target time (secs):| Frequency at target
time (Hz):')
set_param([sys,'/','Chirp Signal1'],...
 'Mask Translate','BB = 2*pi*@1; AA =pi*(@3 -@1)/(@2);t=[0:.1:5];')
set_param([sys,'/','Chirp Signal1'],...
 'Mask Help','The chirp signal has linearly increasing frequency with time.
This block can be used for spectral analysis of nonlinear systems.  ',...
 'Mask Entries','0\/100\/1\/')
% Finished composite block 'Chirp Signal1'.
set_param([sys,'/','Chirp Signal1'],...
 'position',[0,1722,35,1758])
add_block('built-in/Gain',[sys,'/','Gain4'])
set_param([sys,'/','Gain4'],...
 'Gain','15',...
 'position',[80,123,115,167])
add_line(sys,[885,1755;885,1645;925,1645])
add_line(sys,[260,1755;270,1755])
add_line(sys,[145,1740;225,1740])
```

```
add_line(sys,[395,1755;415,1755])
add_line(sys,[145,1740;145,1960;310,1960])
add_line(sys,[145,1740;145,2050;220,2050])
add_line(sys,[480,1755;525,1755])
add_line(sys,[145,1740;145,1990;230,1990])
add_line(sys,[885,1755;885,1510;920,1510])
add_line(sys,[875,1080;875,970;915,970])
add_line(sys,[250,1080;260,1080])
add_line(sys,[135,1065;215,1065])
add_line(sys,[385,1080;405,1080])
add_line(sys,[135,1065;135,1285;250,1285])
add_line(sys,[135,1065;135,1375;210,1375])
add_line(sys,[470,1080;520,1080])
add_line(sys,[135,1065;135,1315;215,1315])
add_line(sys,[875,1080;875,835;910,835])
add_line(sys,[35,1065;90,1065])
add_line(sys,[605,1080;640,1080])
add_line(sys,[765,1080;800,1080])
add_line(sys,[610,1755;650,1755])
add_line(sys,[775,1755;810,1755])
add_line(sys,[885,1755;885,1575;940,1575])
add_line(sys,[980,1575;1015,1575])
add_line(sys,[875,1080;875,915;925,915])
add_line(sys,[965,915;1005,915])
add_line(sys,[375,1285;410,1285])
add_line(sys,[485,1285;565,1285;565,1225;650,1225])
add_line(sys,[485,1285;560,1285;560,1350;650,1350])
add_line(sys,[485,1285;650,1285])
```

```
add_line(sys,[435,1960;480,1960])
add_line(sys,[555,1960;605,1960;605,1900;665,1900])
add_line(sys,[555,1960;600,1960;600,2025;660,2025])
add_line(sys,[555,1960;660,1960])
add_line(sys,[40,1740;100,1740])
add_line(sys,[495,165;505,125])
add_line(sys,[175,165;185,165])
add_line(sys,[120,145;140,150])
add_line(sys,[245,165;250,165])
add_line(sys,[120,145;120,300;230,300])
add_line(sys,[120,145;120,260;140,260])
add_line(sys,[305,165;320,165])
add_line(sys,[120,145;130,330])
add_line(sys,[495,165;505,30])
add_line(sys,[440,165;445,165])
add_line(sys,[370,165;380,165])
add_line(sys,[495,165;505,75])
add_line(sys,[545,75;555,75])
add_line(sys,[355,300;390,300])
add_line(sys,[465,300;480,300;490,245])
add_line(sys,[465,300;480,300;490,350])
add_line(sys,[465,300;500,300])
add_line(sys,[35,145;75,145])
drawnow
% Return any arguments.
if (nargin | nargout)
 % Must use feval here to access system in memory
 if (nargin > 3)
```

```
 if (flag == 0)
 eval(['[ret,x0,str,ts,xts]=',sys,'(t,x,u,flag);'])
 else
 eval(['ret =', sys,'(t,x,u,flag);'])
 end
 else
 [ret,x0,str,ts,xts] = feval(sys);
 end
else
 drawnow % Flash up the model and execute load callback
end
```