ABSTRACT

MODELING, CONTROL AND SIMULATION OF CONTROL-AFFINE NONLINEAR SYSTEMS WITH STATE-DEPENDENT TRANSFER FUNCTIONS

by
Roger Kobla Kwadzogah

There has been no known research that applies nonlinear transfer function to a nonlinear control problem. The belief is that nonlinear systems have no transfer functions. The Laplace transformation required to define transfer functions is not tractable mathematically when the coefficients of the differential equation are functions of state, output and control variables. In other words, it is not defined for systems that do not obey principles of superposition. Only linear systems obey this principle. Therefore, this dissertation work represents the very first research to demonstrate how transfer functions can be used to represent and design feedback control for nonlinear systems.

Real systems are inherently nonlinear. A few important examples include an aerospace vehicle whose mass parameter is variable because of fuel consumption, artificial pancreas and HIV drug delivery systems in the bio-medical field, robot arm and magnetic levitation systems in the mechanical engineering field and phase-locked-loop in the electrical engineering field. The subject of nonlinear system control, however, is more of an art than science. There is no unified framework for analysis and design. Success of a design usually depends on a designer's experience. All the theory and design tools available, e.g., the whole subject of linear algebra, are based on systems described with linear models, which obey the principle of superposition. Control system design by linearization, which is based on approximated linear time invariant (LTI) system design model, is the closest to a general design framework available for nonlinear systems.

The most important problem in a control system designed by linearization is the problem of design model parameter variation during its operation. Obviously, this problem is the result of assuming a constant parameter or LTI design model for a real system that is actually nonlinear or has variable parameter model. In other words, a real system does not have constant parameters as approximated by its LTI design model. This problem is important enough to have specific design methods such as robust control and Horowitz quantitative feedback theory developed to address it. As the system is operated further and further out of the approximate linear range this problem gets worst. Furthermore, the controller based on design by linearization is not a tracking controller. It is a regulator that usually cannot track a varying reference input.

Investigated in the research presented in this dissertation is a nonlinear transfer function-based control method, i.e., one based on a model represented with varying parameters therefore a natural solution to the model parameter variation problem of design by linearization. The class of applicable nonlinear and time-varying systems are those that are affine in their control input such that they can be described by the central concept of this scheme, a state-dependent transfer function (SDTF). The introduction of this concept of nonlinear transfer function design model and the feedback control scheme based on it are the contributions of the research presented in this dissertation.

MODELING, CONTROL AND SIMULATION OF CONTROL-AFFINE NONLINEAR SYSTEMS WITH STATE-DEPENDENT TRANSFER FUNCTIONS

by
Roger Kobla Kwadzogah

A Dissertation
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy in Electrical Engineering

Department of Electrical and Computer Engineering

May 2014

APPROVAL PAGE

MODELING, CONTROL AND SIMULATION OF CONTROL-AFFINE NONLINEAR SYSTEMS WITH STATE-DEPENDENT TRANSFER FUNCTIONS

Roger Kobla Kwadzogah

Dr. MengChu Zhou, Dissertation Advisor Date
Distinguished Professor of Electrical and Computer Engineering, NJIT

Dr. John Carpinelli, Committee Member Date
Professor of Electrical and Computer Engineering, NJIT

Dr. Edwin Hou, Committee Member Date
Associate Professor of Electrical and Computer Engineering, NJIT

Dr. Abdallah Khresishah, Committee Member Date
Assistant Professor of Electrical and Computer Engineering, NJIT

Dr. Ji Zhiming, Committee Member Date
Associate Professor of Mechanical and Industrial Engineering, NJIT

BIOGRAPHICAL SKETCH

Author:	Roger Kobla Kwadzogah
Degree:	Doctor of Philosophy
Date:	May 2014

Undergraduate and Graduate Education:

- Doctor of Philosophy in Electrical Engineering,
  New Jersey Institute of Technology, Newark, NJ, 2014
- Master of Science in Electrical Engineering,
  New Jersey Institute of Technology, Newark, NJ, 1999
- Bachelor of Science in Electrical Engineering,
  New Jersey Institute of Technology, Newark, NJ, 1997

Major: Electrical Engineering

Presentations and Publications:

R. K. Kwadzogah, M. Zhou, X. Wang, "Pointwise Stabilization and Control of a Class of Nonlinear Systems via State-dependent Transfer Functions," *2014 IEEE International Conference on Automation Science and Engineering, Tapei, Taiwan*, Submitted

R. K. Kwadzogah, M. Zhou, S. Li, "Model Predictive Control of Indoor HVAC Systems-A Review," *Proceedings of the 9th IEEE International Conference on Automation Science and Engineering, Madison, WI*, pp. 442-447, August 2013.

R. K. Kwadzogah, D. Misra, "Simultaneous Identification of Friction and a DC Servo Positioning System via Simulation," *International Journal of Intelligent Control and Systems*, vol. 18, no. 1, pp. 10-16, March 2013.

T. N. Chang, R. K. Kwadzogah, R. J. Caudill, "Vibration Control of Linear Robots Using a Piezoelectric Actuator," *IEEE/ASME Transactions on Mechatronics*, vol. 8, no. 4, pp. 439-445, December 2003.

To my beloved family: Cathy, my wife, Tina and Toni, my daughters, and my two academic heroes: my former mentor and advisor the late Prof. Timothy Chang and my current mentor and advisor the Distinguished Prof. MengChu Zhou.

ACKNOWLEDGMENT

My deepest respect, admiration and gratitude go Dr. MengChu Zhou, my advisor, for his confidence in me, his guidance and his support. I am also extremely grateful to the committee members: Dr. John Carpinelli, Dr. Edwin Hou, Dr. Abdallah Khreishah and Dr. Ji Zhiming. I am grateful for the financial support provided to me through the Gr T/R-Match grant and NSF Grant with No. CMMI-1162482. Thanks to all students of Dr. Zhou's group especially Xao Wang, Sisi Li, Sarker Mosharraf and Haobo Lai. Special thanks to my wife Cathy, and my two children Kristina (Tina) and Antoinette (Toni), who made a lot of sacrifices that make it possible for me come this far.

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

Chapter **Page**

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Objective

Even though real systems are inherently nonlinear there is no general theory and therefore no general method for designing their feedback control. All the available control theories, which are based on principles of superposition, are applicable to only LTI systems. The closest to a general method for designing control for nonlinear systems is design by linearization. This is an approach in which the nonlinear system behavior is approximated, within a narrow range of its variables, to LTI system behavior. The assumption is that within this relatively narrow range, the behavior is approximately linear. As long as the operation of the resulting control system is kept within this narrow range, any linear design method can use this approximated LTI model to design a controller.

Sometimes it is possible to design a custom controller based on a nonlinear model rather than the approximate LTI model in design by linearization, but the large signal processing resources usually required to implement it are the reason the approach may not be cost-effective. Design by linearization, just like the LTI model it is based on, usually produces a controller with constant parameters, i.e., a fixed gain controller, which usually requires relatively small signal processing resources to implement and this may be one of the important appeals of the method.

Thus the availability of linear system theory, the lack of its equivalent for nonlinear systems and the relatively small signal processing resource required to implement a linear controller are the main advantages of design by linearization. But the most important problem associated with the method is the design model parameter variations during control system operations, a problem of a failed design model. The real system parameters are not constant, as assumed in the approximate

LTI design model, the fundamental practical problem of linear control theory; real control systems are inherently nonlinear. This problem becomes worst as the system is operated further and further out of the assumed linear range. There are some available methods to address this problem including robust control and frequency domain solutions like Horowitz quantitative feedback theory (QFT).

Furthermore, design by linearization produces a regulator but not a tracking controller. It cannot track a varying reference input unless it is employed in gain scheduling. The objective of the research presented in this dissertation is to demonstrate a nonlinear transfer function-based design approach that addresses the model parameter variation problem of linear theory-based control of practical control systems. The new approach, in theory, may be described as a combination of functionalities of robust control, gain scheduling and adaptive control. It is applicable to a class of nonlinear and time-varying systems that are affine in their control input. This is a class of nonlinear systems that can be described with the new nonlinear transfer function concept of SDTF. Many practical systems belong to this class. In fact, some of real life systems that can benefit from the new paradigm include an aerospace vehicle whose mass parameter is varying due to fuel consumption and others like artificial pancreas device, [24], an HIV drug delivery control system, [19], robot arm (a motorized simple inverted pendulum), overhead crane and magnetic levitation systems and phase-locked-loop (PLL) system.

The concept of SDTF for a nonlinear system is based on the fact that, a nonlinear differential equation that describes the evolution over time of the output of such a system can be expressed in a linear-like form by choosing appropriate state-dependent coefficients (SDCs) [7] and [12]. One very important implication of this idea is that these systems can be regarded as though they are LTI pointwise. If the state of such a system can be measured or estimated at points along its state trajectory, then its differential equation at each such point is linear or linear over infinitesimal intervals between two points, i.e., the SDCs are evaluated from the measured state of the system at each point. At each such point or over each such infinitesimal interval, the Laplace transform can be applied to convert the equation into an equivalent transfer function. This is the basis of an SDTF paradigm. This new perspective immediately highlights what the distinction is between nonlinear and LTI systems. An LTI system is one characterized by a ”constant” transfer function, i.e., a transfer function with a gain factor, zeros and poles that are all constant. A nonlinear system, on the other hand, is one with a ”variable” transfer function, i.e., a transfer function in which at least one of the three characteristic parameters, i.e., the gain factor, the zeros and the poles, vary with time or state of the system which explains the concept of SDTF for nonlinear systems. Another simple way to put this: the parameters of an LTI system are stationary while those of a nonlinear system are non–stationary. This new paradigm of a nonlinear transfer function and the nonlinear feedback control system based on it, that we propose to solve model parameter variations of a real control system designed based on fixed parameters or pure linear model, are the main contributions of the research presented in this dissertation.

1.2 Dissertation Organization

Chapter 2 is a literature survey of nonlinear systems and control methods. Chapter 3 introduces the concept of SDTF quantitatively and illustrates its utility with simulated examples by using a friction-free simple inverted pendulum. Chapter 4 examines properties of the SDTF and how they relate to stability, controllability and equilibrium of a nonlinear system. Chapter 5 presents the design of an SDTF-based feedback controller known as a pointwise stabilizing controller (PSC). The design is illustrated with two simple nonlinear system examples and finally generically with a system of an arbitrary order. Before presenting a complete design that can be tested experimentally, a model parameter identification scheme that can be used to determine the parameters of a real system is presented in Chapter 6. Then in Chapter 7 a realistic digital design is simulated for a dc motorized inverted pendulum. This system includes components such as a pulse width modulator (PWM), analog-to-digital converter (ADC) and high power amplifier and so on. Finally, Chapter 8 is a summary of the work and also suggestions for future work.

CHAPTER 2

REVIEW OF NONLINEAR CONTROL MODELS AND METHODS

This chapter is a brief review of nonlinear control models and methods. The powerful analysis and design framework available for linear systems is founded on the superposition principle. This principle does not apply to nonlinear systems. While differential equations of a linear system have constant coefficients those of a nonlinear system have non-constant coefficients. As such there are a number of phenomena observed in nonlinear systems that are not present in linear systems [20]. One such phenomenon is multiple isolated equilibrium points. The implication of this difference is that the state of a nonlinear system unlike that of a linear system can converge to one of possible equilibrium points depending on the initial state of the system.

2.1 Review of Design Methods

The variety of design methods available for nonlinear control is a true testimony to the lack of a general design framework.

2.1.1 Linearization about an Equilibrium Operating Point

Linearization about an equilibrium point is a general method applicable to nonlinear systems. It is based on the realization that the behavior of a nonlinear system near one of its equilibrium points can be determined via linearization with respect to that point. If the involved nonlinearities are smooth, then a control system may be reasonably approximated by this linearized system whose dynamics is described by linear differential equations [27] and [20]. The linearized model is valid for small deviation around that equilibrium point. Any linear system control design method can be used to design the controllers for such models. Should the states of the system stray too far beyond the operating point, the control system might become unstable.

2.1.2 Feedback Linearization

In design by feedback linearization, a nonlinear system is transformed fully or partially into a linear system by using part of the control input. Once again the aim is to use the existing linear system control design theory for the transformed linear systems. Effectively the nonlinearities present in the system are canceled out. The method, however, is only applicable to a limited class of nonlinear systems. These are systems that are of minimum phase or input-state linearizable [27] and [20]. The approach does not guarantee robustness to model parameter variations. It also requires full state measurement for its implementation. It has, however, been successfully, used in the control of helicopters, high performance aircraft, industrial robots and bio medical devices [22]. Bao et al.,[3], have used the approach to successfully control a three-phase photovoltaic inverter with a filter.

2.1.3 Adaptive Control

An adaptive controller differs from a fixed gain controller in that its parameters are variable and there is a mechanism for adjusting these parameters on-line based on signals [21] [27]. There are two versions of the method: model reference adaptive control and self-tuning adaptive control. In the former, the value of an unknown system parameter is adjusted rather than estimated such that the system output can track that of a reference model. Stability and tracking error are usually guaranteed. In the latter, the unknown model parameters are estimated until they are found to fit available input-output data from the real process. Popular estimation algorithms like least squares and its various extensions are used. Proportional integral derivative (PID), pole placement and linear quadratic regulators are then used. The issues are: 1) convergence and stability are difficult to guarantee and 2) signals must be sufficiently rich to allow the estimated parameters to converge to their true values.

2.1.4 State Dependent Ricatti Equation Model

The idea that a nonlinear differential equation of a control-affine nonlinear system can be expressed in a linear form, with coefficients that are functions of the state of the system is first recognized in [12] [8]. But the present form of the control scheme based on this idea, known as state-dependent Ricatti equation (SDRE), is due to the work in [8]. A complete review of this work is available in [6]. In [8], an SDRE scheme was successfully used for the optimal control of aerospace vehicles in 1996. The work in [12] also hints at the idea in what is called extended linearization, i.e., the possibility of using any linear system design method to design control for control-affine nonlinear systems provided the equation can be re-written in terms of appropriate set of SDCs. A similar idea in [11] is used except that the SDCs are formulated as functions of time rather than system state.

Even though the SDRE scheme has been empirically demonstrated successfully in a number of applications, it has no theoretical foundation. It is simply based on intuition thus its guarantee of stability has not been proven. An extension of the method based on closed-loop pole assignment rather than the linear quadratic regulator control law has successfully been demonstrated in [30]. The work shows that the resulting nonlinear closed-loop system reduces to a linear transfer function with assigned closed-loop poles and concludes that if controllability and observability of the nonlinear system can be guaranteed at each point along a given trajectory, stability of the closed-loop system is guaranteed by the closed-loop pole assignment control law.

2.2 Nonlinear System Model

Transfer functions are one of the most powerful control system design tools. It is the main input to such design methods as Bode, Root locus, Nyquist, Nichols and Horowitz QFT. It is used to describe a system's input-output behavior in Laplace and Fourier transform domains for LTI systems. These are systems that can be described with differential equations having constant coefficients. It is the coefficients that are used to define the corresponding transfer function. The change from the differential equations representation to transfer function representation is achieved by Laplace transform. Nonlinear systems, on the other hand, are described with nonlinear differential equations, i.e., equations with coefficients that are not constant can be expressed as functions of the state of the system in general. Why are nonlinear systems not usually described via transfer functions? The problem is that we cannot derive Laplace transform in a closed-form of input output equation with state-dependent and non-constant coefficients. The solution proposed in this research as the basis of the SDTF concept is instead of a closed-form Laplace transform description. The transformation is performed from point to point or pointwise. Why should this method work? Because the varying parameters in design by linearization are effectively the gain factor, the poles and zeros of a nonlinear system hence, the natural way to solve the problem is to describe the nonlinear system in terms of these parameters. A group of applicable nonlinear systems, as represented by the following equation, are said to be affine in their control input.

$$\dot{x} = f(x) + g(x)u \tag{2.1}$$

where $x \in R^n, u \in R, f(x)$ and $g(x)$ are smooth $g(x) \neq 0$

2.2.1 Transfer Functions of a Nonlinear System

The idea of transfer function for a nonlinear system at first may sound inappropriate. This is because the belief is that transfer functions are defined only for linear systems and not nonlinear systems. Despite this, there have been attempts by researchers to characterize nonlinear systems in terms of transfer functions. The first known research in [34] formulates nonlinear system transfer function based on so-called non-commutative polynomial ring expressed in the differentiation operator as mapping between signal spaces, a pseudo-linear algebra method [25], [4] [10]. Then are the studies [15] and [14] that extend this concept to discrete time systems. Each one concludes that a nonlinear system can characterized by a unique transfer function that is not only a function of the system input and output variables but also their differentials. Its dependency on the differentials of input and output variables of a nonlinear system makes it unsuitable as a control design model. This is because the objective of these early researches is not to use it as a design model but to investigate the controllability and observability properties of nonlinear systems. Thus Zheng and Cao[34] use the concept to investigate a nonlinear system's structural controllability. They also use the concept to investigate in nonlinear systems the notion of equivalence systems and transmission zeros.

CHAPTER 3

STATE-DEPENDENT TRANSFER FUNCTIONS

The concept of an SDTF for a nonlinear system is formally introduced and quantitatively defined in this chapter. It is illustrated via a simple inverted pendulum. What this model for the simple inverted pendulum takes on a continuum of varying LTI systems as the pendulum moves freely in absence of control is simulated to provide the clear understanding of a nonlinear transfer function as a pointwise concept. The evolving poles of the system at a few sampling points along the free motion are tabulated to illustrate the non-stationary parameters of this nonlinear system.

3.1 The SDTF Model

Consider a single-input-single-output (SISO) control-affined nonlinear system of order n described by the following equation:

$$\begin{aligned} \dot{x} &= f(x) + b(x)u \\ y &= h(x) \end{aligned} \tag{3.1}$$

where $x \in R^n, u \in R, f(x), b(x)$ and $h(x)$ are smooth $b(x) \neq 0$ and $h(x) \neq 0$ and the origin of the state space is an equilibrium point, i.e., $f(0) = 0$

Since the origin of the state space is an equilibrium point of this system Eq.(3.1) can also be re-written as follows:

$$\begin{aligned} \dot{x} &= A(x)x + b(x)u \\ y &= c(x)x \end{aligned} \tag{3.2}$$

where $f(x) = A(x)x, h(x) = c(x)x$ and $A(x) \in R^{n \times n}, c(x) \in R^n$

This explicitly shows, all the possible equilibrium points present in this system, including the one at the origin. These are points where the control input equals to zero and can be found by solving the following algebraic equation:

$$A(x)x = 0 \tag{3.3}$$

The solution is obtained when either the state vector x is zero, i.e., $x = 0$ or when the state-dependent dynamic matrix $A(x)$, is zero, i.e., $A(x) = 0$. Their respective solution is the equilibrium at the origin required and the additional equilibrium points that are not at the origin.

The *nth* order system in Eq. (3.2), can be expanded into a total of n first order equations by choosing the following SDCs such that the resulting representation looks like a controllable canonical form-dependent (CCFD) state equation:

$$\begin{aligned}
&\dot{x}_1 = a_1(x)x_2 \\
&\dot{x}_2 = a_2(x)x_3 \\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\
&\dot{x}_{n-1} = a_{n-1}(x)x_{n-2} \\
&\dot{x}_n = q_1(x)x_1 + q_2(x)x_2 + ... + q_n(x)x_n + b_n(x)u \\
&y = c_1(x)x_1
\end{aligned} \tag{3.4}$$

The chosen CCFD SDC are the following matrices and vectors:

$$A(x) = \begin{pmatrix} 0 & a_1(x) & 0 & 0.... & 0 \\ 0 & 0 & a_2(x) & 0.... & 0 \\ ... & ... & ... & .... & ... \\ 0 & 0 & 0 & 0.... & a_{n-1}(x) \\ q_1(x) & q_2(x) & q_3(x) & q_4(x).... & q_n(x) \end{pmatrix} \quad b(x) = \begin{pmatrix} 0 \\ 0 \\ ... \\ 0 \\ b_n(x) \end{pmatrix}$$

$$c(x) = \begin{pmatrix} c_1(x) & 0 & 0 & 0.... & 0 \end{pmatrix}$$

Hence, the equilibrium points that are not at the origin of this system are found by solving the following system of algebraic equations:

$$\begin{aligned} &a_1(x) = 0 \\ &a_2(x) = 0 \\ &\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ &a_{n-1}(x) = 0 \\ &q_1(x) + q_2(x) + ... + q_n(x) = 0 \end{aligned} \tag{3.5}$$

The SDTF of this *nth* order control-affine nonlinear system described in Eq. (3.1) is determined by the following formula:

$$G(x,s) = c(x)(sI - A(x))^{-1}b(x) \tag{3.6}$$

where I is an $n \times n$ identity matrix and s the complex frequency variable.

3.1.1 SDTF of a Simple Inverted Pendulum

To illustrate the SDTF concept presented in the previous section, consider a simple friction-free inverted pendulum described by the following second order nonlinear vector-matrix differential equation:

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -g/L\sin(x_1) + u \\ y &= x_1 \end{aligned} \tag{3.7}$$

where g is the acceleration due to gravity and L is the length of the pendulum.

The CCFD SDC required to define its SDTF are given as:

$$A(x) = \begin{pmatrix} 0 & 1 \\ -\frac{g}{L}\frac{\sin x_1}{x_1} & 0 \end{pmatrix}$$

$$b(x) = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

$$c(x) = \begin{pmatrix} 1 & 0 \end{pmatrix}$$

The resulting SDTF of the system, according to the formula in (3.6), is:

$$G(x, s) = \frac{1}{s^2 + \frac{g}{L}\frac{\sin x_1}{x_1}} \tag{3.8}$$

Figure 3.1 is a MATLAB/SIMULINK simulation block of this system with the external control input set to zero. It is the model of this system. Using it to simulate the free motion of the system shown in Figure 3.2, we start at an initial position of $x_1 = \frac{\pi}{2}$, (90 degrees) and runs for two cycles. As can be seen, the pendulum position oscillates about its marginally stable equilibrium point, at the vertically down position, the origin, $x_1 = 0$. The SDTF of the simple inverted pendulum without friction is a function of only its position and time and not its velocity. The trajectory, as shown in Figure 3.2 is periodic. This result is used to calculate the SDTF and its poles, at eight equally spaced time points tabulated in Table 3.1.

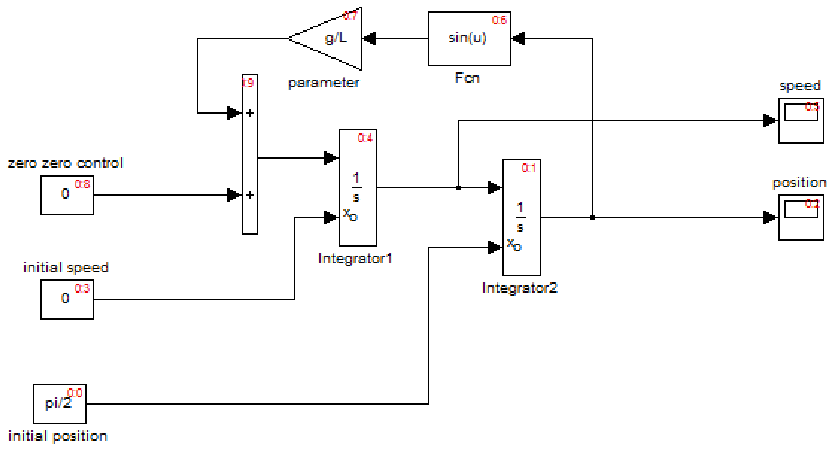

Figure 3.1 SIMULINK Model of the friction-free simple inverted pendulum..

This illustration shows that the SDTF of the friction-free simple inverted pendulum has only poles that are state-dependent, it has no zeros and its gain factor

Table 3.1 The Pendulum SDTF and Poles at Some Points Along its Trajectory

	Time in seconds	*SDTF*	*Poles*
(i)	0	$\frac{1}{s^2+0.63662}$	$p_1 = -j0.79788$ $p_2 = +j0.79788$
(ii)	1	$\frac{1}{s^2+0.82019}$	$p_1 = -j0.90565$ $p_2 = +j0.90565$
(iii)	2	$\frac{1}{s^2+0.99308}$	$p_1 = -j0.99653$ $p_2 = +j0.99653$
(iv)	3	$\frac{1}{s^2+0.73766}$	$p_1 = -j0.85887$ $p_2 = +j0.85887$
(v)	4	$\frac{1}{s^2+0.65653}$	$p_1 = -j0.81026$ $p_2 = +j0.81026$
(vi)	5	$\frac{1}{s^2+0.90845}$	$p_1 = -j0.95313$ $p_2 = +j0.95313$
(vii)	6	$\frac{1}{s^2+0.94232}$	$p_1 = -j0.97073$ $p_2 = +j0.97073$
(viii)	7	$\frac{1}{s^2+0.67395}$	$p_1 = -j0.82095$ $p_2 = +j0.82095$
(ix)	8	$\frac{1}{s^2+0.70572}$	$p_1 = -j0.84007$ $p_2 = +j0.84007$

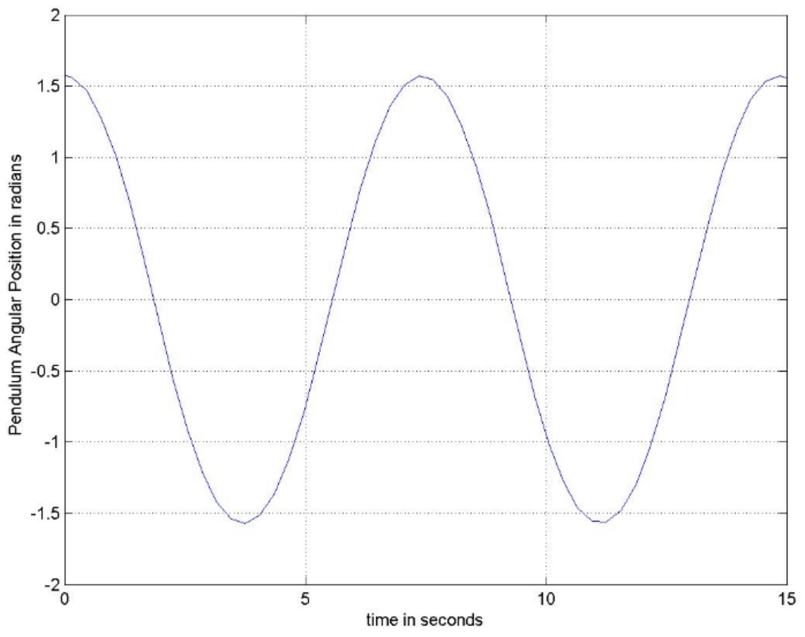

Figure 3.2 Trajectory of friction-free simple inverted pendulum.

is constant. The two state-dependent poles, as shown at a few points in Table 3.1, vary along the periodic trajectory shown in Figure 3.2. Each cycle of the trajectory has two equilibrium points with one at the origin and the second away from the origin at $x_1 = \pi$. At the origin, $x_1 = 0$, the SDTF of the pendulum assumes the transfer function of a marginally stable harmonic oscillator given as:

$$G(x, s) = \frac{1}{s^2 + g/L} \tag{3.9}$$

At the nonzero equilibrium point, $x_1 = \pi$, on the other hand, the SDTF of the pendulum has transfer function of an unstable double integrator given by:

$$G(x, s) = \frac{1}{s^2} \tag{3.10}$$

Thus, as demonstrated by these results, a control-affine nonlinear system assumes a continuum of LTI systems, and the transfer function of these continuum which is equivalent to that of the nonlinear system is described by the concept of an SDTF.

CHAPTER 4

PROPERTIES OF SDTF

In this chapter, properties of the SDTF of a nonlinear system and their relation to the equilibrium, stability and controllability of the system are examined.

4.1 Nonlinear System Equilibrium and Stability

Practically, global stability of a dynamic system, linear or nonlinear, is a property of its equilibrium point at the origin of the state space. While the stability of a linear system is always global, that of a nonlinear system is global only if its equilibrium point is at the origin only. It is the additional equilibrium points away from the origin in case of a nonlinear system that complicates its stability and stabilizability. Thus, a nonlinear system with additional equilibrium points away from the origin can only have local but not global stability. Clearly, a globally stable (asymptotic) control system, linear or nonlinear, implies that when initialized away from the origin in absence of its external control input, it will return to the origin within a finite time. Otherwise the system is unstable. The system is stable if the eigenvalues of its dynamics matrix (state model) or poles of its transfer function have negative real parts, and otherwise the system is unstable. The control input of an unstable system can, however, be configured via its state or output in a feedback to make it stable provided it is controllable or stabilizable.

An equilibrium point of a dynamic system is any point at which the system can remain indefinitely in absence of external inputs or a point where it is at a steady state. If an equilibrium point at the origin or away from the origin, in case of a nonlinear system, is stable, then when a system initialized away from it, it will eventually return to it in absence of its external control, and otherwise the equilibrium point is unstable.

A linear system has only one equilibrium point at the origin of the state space. Thus, a stable linear system is one with a stable equilibrium point. An unstable linear system has unstable equilibrium. A stable linear system is said to be bounded-input bounded-output (BIBO) stable as a bounded input produces a bounded output, and otherwise the output diverges out of control.

The stability situation is different for nonlinear systems generally because unlike linear systems they can have additional equilibrium points away from the origin of the state space. The problem with this is since an equilibrium point is a point where the unforced system converges to a steady state, if there are multiple points of them what is the rule that determines which one the system should converge to? The answer to this question is provided by Lyapunov's stability concepts. The most stable equilibrium point is where the total energy stored in the system is the minimum. Assume the energy stored in the system is a function of its state. Clearly its minimum being zero occurs at the origin. Therefore, the equilibrium at the origin is the most stable. In nonlinear systems, unlike linear ones, the concept of global stability is associated with the equilibrium at the origin while non-zero ones away from the origin are associated with concept of local stability.

Properties of SDTF must be the same as those of the transfer function of LTI system except it is a pointwise property of the corresponding. By stating that a given SDTF is BIBO stable at point, we mean that the equivalent LTI system represented by the same transfer function is stable. The stability of an SDTF changes from point-to-point. An SDTF is globally stable if the transfer functions of all the possible LTI systems it assumes along the trajectory of the nonlinear system are stable. This requires that: 1) the origin of the state space must be an equilibrium point and 2) the state-dependent poles of the SDTF at every point along the trajectory must occur in the left half s-plane. Just as in a linear system case an unstable equilibrium at the origin can be stabilized by feedback, an unstable one in a nonlinear system can also be stabilized through feedback. This shows the importance of equilibrium requirement at the origin in a nonlinear system. The SDTF of simple inverted pendulum in Table 3.1 shows that it is marginally stable (the two poles are located on the imaginary axis of the s-plane) at every point along the state trajectory except at the equilibrium point at $x_1 = \pi$ where it assumes the dynamics of a double integrator (the two poles are located at the origin of the s-plane).

4.1.1 Controllability and Stability

While stability is a very important property of a control system, controllability is another equally important property. Generally a system, linear or nonlinear, is uncontrollable if no control path exists to influence its dynamics from its input to its output, i.e., if its control input cannot affect its output. Thus, for an SISO nonlinear system described by SDTF to be controllable at any point along its trajectory, its gain factor of the SDTF, which defines that control path between the input and output, cannot be zero. In other words, the gain factor of an SDTF defines controllability of an SISO nonlinear system. Consequently a very important conclusion can be drawn from this: a nonlinear system represented by SDTF is globally controllable if and only if its gain factor is independent of the state of the system. To demonstrate this, consider the system described by the following equation:

$$\begin{aligned} \dot{x}_1 &= 3x_2 - x_1x_2 \\ \dot{x}_2 &= -2x_1 - 3x_2 + u \\ y &= x_1 \end{aligned} \tag{4.1}$$

The choice of SDC required to obtain a CCFD representation of this system is given by:

$$A(x) = \begin{pmatrix} 0 & 3 - x_1 \\ -2 & -3 \end{pmatrix}$$

$$b(x) = \begin{pmatrix} 0 \\ x_1 \end{pmatrix}$$

$$c(x) = \begin{pmatrix} 1 & 0 \end{pmatrix}$$

The SDTF of this system, according to the formula in 3.6, is:

$$G(x, s) = \frac{3 - x_1}{s^2 + 3s - 2(3 - x_1)} \tag{4.2}$$

This system has two equilibria. One is a point at the origin while another is a line along $x_1 = 3$. Thus, since the origin is not the only equilibria, this system is not globally stabilizable asymptotically. As can also be seen in (4.2), the gain factor of its SDTF is a function of its state. This also confirms that this system is not globally controllable. Because along the lines through the origin $x_1 = 0$ and $x_1 = 3$, the gain factor of the SDTF becomes zero leading to the loss of its controllability. If the poles of the system are unstable at the point where its controllability is lost, then bounded input at that point will produce a diverging output of instability at the point. The stability of this system, at any point, is determined by the two poles of the SDTF which are the roots of the state-dependent polynomial $s^2 + 3s - 2(3 - x_1)$, as long as controllability at the point does not vanish, i.e., as long as the point stays off the two no-controllability lines $x_1 = 0$ and $x_1 = 3$.

4.1.2 Controllability and Equilibrium

Three parameters required to define a transfer function of a dynamic system: a gain factor, zeros and poles of the system. For an LTI system, all three of these parameters are constant. Thus, the system is nonlinear when at least one of the them is variable or state-dependent. In the example with the simple inverted pendulum only the poles are state-dependent while the gain factor and zeros are constant. In the system of Eq. (4.1) both the poles and gain factor are state-dependent and the zeros are constant.

The state space representation of a dynamic system is not unique but its transfer function is. For this reason, it is clear that not all the possible SDCs that can be used to represent the nonlinear system described in Eq. (3.1). This is a very important point that was missed by the original SDRE research in 1996 until the work [16] shows that some choices of SDCs can actually mask or misrepresent the controllability property of the original system. The choice of SDC should be based on preserving a system's controllability. Therefore, a valid representation is of a CCFD form. A CCFD is a representation that allows the control input to enter the system through a chain of integrators as shown by the generic example in Eq. (3.4). It guarantees that the control is able to move every state. It therefore, reflects if the original system is controllable or uncontrollable. To demonstrate this, consider the following famous example in from [16]:

$$\begin{aligned} \dot{x}_1 &= x_2 - x_1 x_2 \\ \dot{x}_2 &= u \\ y &= x_1 \end{aligned} \tag{4.3}$$

Following the recommendations by the SDRE control scheme, the best choice for SDCs to describe this system are:

$$A(x) = \begin{pmatrix} x_2 & 1 \\ 0 & 0 \end{pmatrix}$$

$$b(x) = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

$$c(x) = \begin{pmatrix} 1 & 0 \end{pmatrix}$$

The SDTF based on this choice, according to the formula in (3.6), is:

$$G(x, s) = \frac{1}{s^2 + x_1 s} \tag{4.4}$$

SDTF shows that the original system in (4.3) is globally controllable and therefore globally stabilizable since the gain factor of the SDTF is constant but not state-dependent. The calculation result of the controllability matrix based on this choice of SDCs also confirms this. But the simulation of the original system shows that this is not the case. Instead the system is uncontrollable along the line $x_1 = -1$. To demonstrate this, consider the following CCFD SDCs:

$$A(x) = \begin{pmatrix} 0 & x_1 + 1 \\ 0 & 0 \end{pmatrix}$$

$$b(x) = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

$$c(x) = \begin{pmatrix} 1 & 0 \end{pmatrix}$$

The SDTF based on this choice, according to the formula in (3.6), is:

$$G(x, s) = \frac{x_1 + 1}{s^2} \tag{4.5}$$

As can be seen, the CCFD SDCs show a gain factor that is state-dependent confirming that original system losses controllability along the line $x_1 = -1$ and therefore, the system is locally but not globally controllable and stabilizable i.e., it is only controllable and therefore, stabilizable only in the region $x_1 > -1$ of the state space.

There is some relationship between controllability and non-zero equilibrium points in an SISO control-affine nonlinear system. To examine this, consider the general control-affine SISO nonlinear system described in Eq. (3.4). Its CCFD representations are:

$$A(x) = \begin{pmatrix} 0 & a_1(x) & 0 & 0.... & 0 \\ 0 & 0 & a_2(x) & 0.... & 0 \\ ... & ... & ... & .... & ... \\ 0 & 0 & 0 & 0.... & a_{n-1}(x) \\ q_1(x) & q_2(x) & q_3(x) & q_4(x).... & q_n(x) \end{pmatrix} b(x) = \begin{pmatrix} 0 \\ 0 \\ ... \\ 0 \\ b_n(x) \end{pmatrix}$$

$$c(x) = \begin{pmatrix} c_1(x) & 0 & 0 & 0.... & 0 \end{pmatrix}$$

where its non-zero equilibrium points are solutions to the following system of equations:

$$\begin{aligned} & a_1(x) = 0 \\ & a_2(x) = 0 \\ & .... \\ & a_{n-1}(x) = 0 \\ & q_1(x) + q_2(x) + ... + q_n(x) = 0 \end{aligned} \tag{4.6}$$

Symbolic toolbox in MATLAB can be used to compute the resulting SDTF according to the Eq. (3.6). The result is given as follows:

$$G(x, s) = \frac{c_1(x) b_n a_1(x) a_2(x)...a_n(x)}{s^n + q_n(x)s^{n-1} + q_{n-1}(x)s^{n-2} + ... + q_2(x)s + q_1(x)} \tag{4.7}$$

A close examination of the result in Eq. (4.7) reveals that its gain factor, the numerator coefficient will attain the value of zero at any of the system non-zero equilibrium points. Clearly, the relation is that an SISO control-affine nonlinear system has no controllability or is uncontrollable at any of its non-zero equilibrium points.

CHAPTER 5

THE POINTWISE STABILIZING CONTROL

This chapter discusses the design of a pointwise stabilizing controller required for feedback control of an SISO control-affine nonlinear system described with an SDTF model. A number of design examples including the friction-free simple inverted pendulum are used to illustrate the proposed methodology.

5.1 Output Feedback Control System

Because SDTF is a frequency domain model, the method employed in this work for design of its system controller is frequency compensation as depicted in Figure 5.1. As a pointwise model, an SDTF used in design must also result in a pointwise or

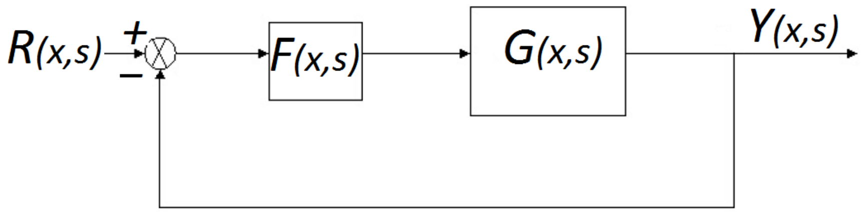

Figure 5.1 PSC implementation.

a state-dependent controller. As the SDTF changes from one point to another, its feedback controller must similarly change to be able to keep it stable. It is an SISO system model and the controller is for an output feedback but not a state feedback. The resultant controller is known as the pointwise stabilizing controller (PSC). It is a discrete-time controller to be designed and implemented only online as the control system is being operated. Even though nonlinear systems are continuous in nature and hard to meaningfully digitize, their control performance can be as good as desired as long as the sampling frequency is fast enough. This controller will, however, require more signal processing resources than a fixed gain controller in a linear system design.

However, with abundance of inexpensive high speed microprocessors today, the design is completely feasible and cost-effective.

The main objective of PSC is to keep the evolving nonlinear system, whose dynamics is described by its SDTF, stable at every sampling point. To achieve this via an output feedback controller, an evolving compensating lead, lag or lag-lead filter is required at each sampling point. The filter parameters are determined not only by the evaluated SDTF of the open-loop system but also by the desired closed-loop performance of the control system. The filter therefore, must be one that will produce a stable closed-loop system based on the evaluated SDTF model at the current sampling instant. To guarantee a stable closed-loop system at each sampling instant, the required filter is designed based on a pole placement algorithm.

Assume $G(x,s)$ is the SDTF evaluated for a given system at a given sampling instant t and that $F(x,s)$ is transfer function of the filter required to compensate $G(x,s)$ such that the resulting closed-loop transfer function denoted by$H(x,s)$ is stable and also meets desired performance requirement. Then the following relationship exists among all of three transfer functions at the sampling instant:

$$H(x,s) = \frac{F(x,s)G(x,s)}{1+F(x,s)G(x,s)} \tag{5.1}$$

This closed-loop SDTF, at every sampling instant, is a function of the filter parameters and system state. It is actually the coefficients of its characteristic polynomial that are the functions of state and filter parameters. For it to be stable, this characteristic polynomial must be a Hurwitz type, i.e., its roots, the poles of the closed-loop SDTF, must all be located in the left half of the s-plane. Therefore, any desired closed-loop poles can be used to synthesize a desired Hurwitz polynomial whose coefficients can then be set equal to those of the state and filter parameter-dependent one of $H(x,s)$. This will then generate a set of algebraic equations with state-dependent coefficients in the filter parameters. To be able to have as many algebraic equations as the total number of unknown filter parameter, a condition important for a unique solution for filter parameters, the filter transfer function must be improper rational and one order less than the order of the open-loop SDTF.

At any sampling instant, the state of the system required to evaluate the coefficients of the filter parameter equations are sampled. The filter parameter equations are then solved. If a solution cannot be found, the filter from the previous sampling point is used. This ability to assign stable closed-loop poles to the nonlinear control system designed based on the SDTF model guarantees its pointwise stability.

To make design procedure clearer, consider the following second order mass-spring-damper system. The spring parameter is not constant instead a nonlinear function, $f_1(x_1)$, of its displacement, x_1. The damping factor parameter not constant either. It is also a nonlinear function, $f_2(x_2)$, of its velocity, (x_2). The mass of the spring is m:

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -\frac{f_1(x_1)}{m} - \frac{f_2(x_2)}{m} + \frac{u}{m} \\ y &= x_1 \end{aligned} \tag{5.2}$$

To obtain the SDTF design model, the following CCFD SDCs are chosen:

$$A(x) = \begin{pmatrix} 0 & 1 \\ 0 & \left(-\frac{f_1(x_1)}{mx_1} - \frac{f_2(x_2)}{mx_2}\right) \end{pmatrix}$$

$$b(x) = \begin{pmatrix} 0 \\ \frac{1}{m} \end{pmatrix}$$

$$c(x) = \begin{pmatrix} 1 & 0 \end{pmatrix}$$

which leads to the following SDTF model:

$$G(x,s) = \frac{\frac{1}{m}}{s^2 + \frac{f_2(x_2)}{mx_2}s + \frac{f_1(x_1)}{mx_1}} \tag{5.3}$$

For this second order system, the PSC design algorithm requires a compensating filter $F(x,s)$ with an improper rational transfer function of first order given as follows:

$$F(x,s) = \frac{d_0 s + d_1}{s + e_1} \tag{5.4}$$

The compensated closed-loop system according the Eq. (5.1) is given as:

$$H(x,s) = \frac{\frac{d_0}{m}s + \frac{d_1}{m}}{s^3 + (\frac{f_2(x_2)}{mx_2} + e_1)s^2 + (e_1\frac{f_2(x_2)}{mx_2} + \frac{f_1(x_1)}{mx_1} + \frac{d_0}{m})s + (e_1\frac{f_1(x_1)}{mx_1} + \frac{d_1}{m})} \tag{5.5}$$

The denominator polynomial of the closed-loop system is a function of the compensating filter parameters which determines the closed-loop system response. Assume that the desired closed-loop control system performance requires closed-loop poles with the following Hurwitz characteristic polynomial:
$s^3 + \alpha_1 s^2 + \alpha_2 s + \alpha_3$
Then filter parameters are solutions to the following set of state-dependent of algebraic equations:

$$\begin{aligned} &\frac{f_2(x_2)}{mx_2} + e_1 = \alpha_1 \\ &e_1\frac{f_1(x_2)}{mx_2} + \frac{f_1(x_1)}{mx_1} + \frac{d_0}{m} = \alpha_2 \\ &e_1\frac{f_1(x_1)}{mx_1} + \frac{d_1}{m} = \alpha_3 \end{aligned} \tag{5.6}$$

A simpler numerical example of the simple inverted pendulum is now illustrated with simulation as follows. The SDTF of this system is given by Eq. (3.8), so its closed-loop SDTF is:

$$H(x,s) = \frac{d_0 s + d_1}{s^3 + e_1 s^2 + (d_0 + \frac{\sin(x_1)}{x_1})s + d_1 + e_1\frac{\sin(x_1)}{x_1}} \tag{5.7}$$

with characteristic polynomial $H_c(x, s)$ given as:

$$H_c(x, s) = s^3 + e_1 s^2 + (d_0 + \frac{\sin(x_1)}{x_1})s + d_1 + e_1 \frac{\sin(x_1)}{x_1} \tag{5.8}$$

Choosing the following three stable closed-loop poles: $s_1, s_2 = -2 \pm j1$ and $s_3 = -4.5$, we have the closed-loop Hurwitz characteristic polynomial is:

$$H_u(s) = s^3 + 8.472s^2 + 22.89s + 22.36 \tag{5.9}$$

Finally setting coefficients of Eq. (5.8) equal to those of Eq.(5.9) leads to the following system of algebraic equations containing the filter parameters:

$$\begin{aligned} &e_1 = 8.472 \\ &(d_0 + \frac{\sin(x_1)}{x_1}) = 22.89 \\ &d_1 + e_1 \frac{\sin(x_1)}{x_1} = 22.36 \end{aligned} \tag{5.10}$$

Once the output, x_1, of this system is sampled at each point, the compensator filter parameters are solved for. MATLAB/SIMULINK design and simulation model of the control system is shown in Figure 5.2.

Reference pulse tracking performance of the PSC for this system at low and higher bandwidths are shown Figures 5.3 and 5.4, respectively.

The pulse command goes between the vertically down position and 90 degrees. As both both simulations shown, the pendulum arm follows, however, with a large overshoot. In the simulation with the lower closed-loop bandwidth, shown in Figure 5.3, the response has a larger steady state error. There is no steady state error in the second simulation with the higher bandwidth. But the reduction in overshoot is not as remarkable. This can be explained by the very small or no damping in the open loop system since there is no friction. Such a system is usually difficult to control since all the damping has to come from the controller. This explains the excessive overshoot in these responses. This why friction is said to have a stabilizing effect in

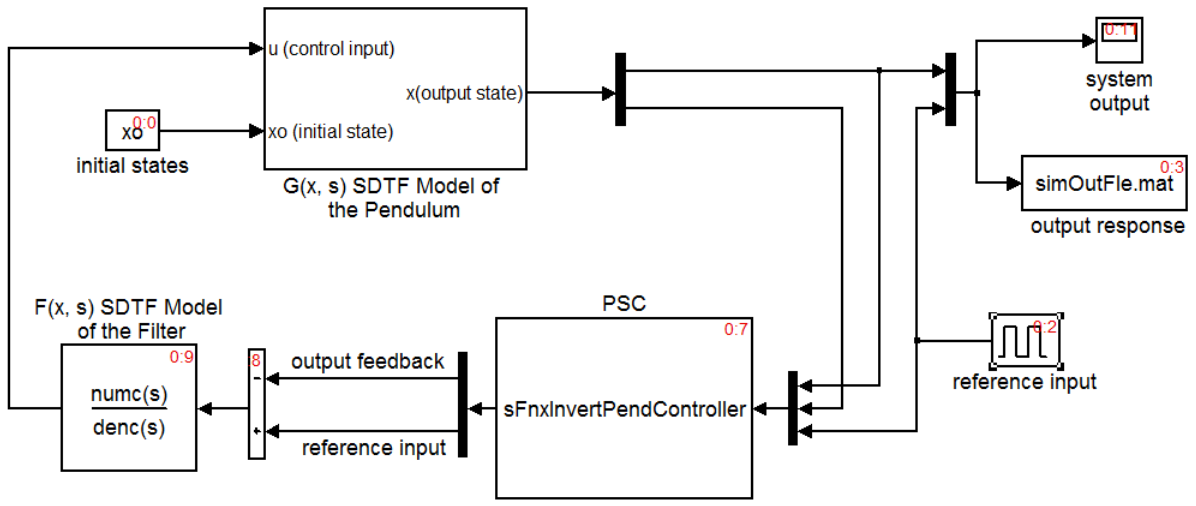

Figure 5.2 PSC control of the simple pendulum.

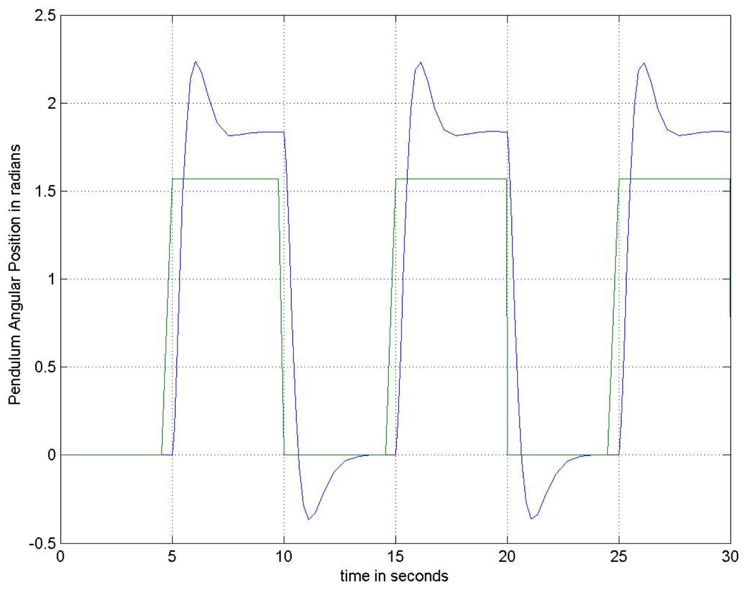

Figure 5.3 PSC pulse reference tracking at low closed-loop bandwidth.

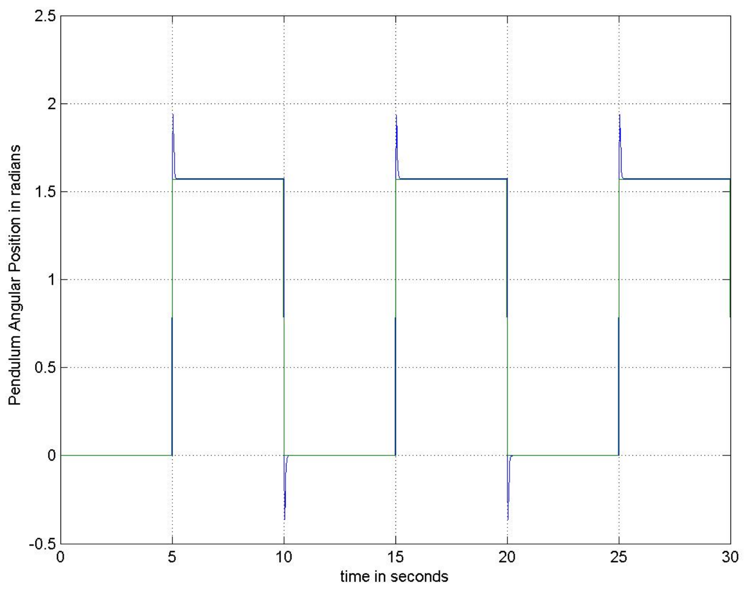

Figure 5.4 PSC pulse reference tracking at high closed-loop bandwidth.

a dynamic system. i.e., it always removes energy from the system, thus driving it to a steady state.

5.1.1 General Design Procedure for PSC

To apply the PSC design procedure to SISO control-affine nonlinear systems of any order, consider the *nth* nonlinear system described by Eq. (5.12). As a PSC design requirement, the transfer function of the compensating filter must be improper and of order $n-1$. Hence, we have the general form of this filter:

$$F(x,s) = \frac{d_0 s^{n-1} + d_1 s^{n-2} + ... + d_{n-1}}{s^{n-1} + e_1 s^{n-2} + ... + e_{n-1}} \tag{5.11}$$

The resulting closed-loop SDTF of the control system, at any given sampling instant, is therefore, given as:

$$H(x,s) = \frac{d_0 s^{n-1} + d_1 s^{n-2} + ... + d_{n-1}}{(s^{n-1} + e_1 s^{n-2} + ... + e_{n-1})(s^n + q_n(x)s^{n-1} + q_{n-1}(x)s^{n-2} + ... + q_2(x)s + q_1(x))} \tag{5.12}$$

The coefficients of its characteristic polynomial, an $(2n-1)$th degree polynomial, are functions of a state and the desired filter parameters d_i and e_j where $i = 0, 1, ...n-1$ and $j = 1, 2, ...n-1$. This polynomial can be be expanded as follows:

$$P_c(x,s) = s^{2n-1}+f_1(x,d_i,e_j)s^{2n-2}+f_2(x,d_i,e_j)s^{2n-3}+...+f_{2n-2}(x,d_i,e_j)s+f_{2n-1}(x,d_i,e_j) \tag{5.13}$$

To force the closed-loop SDTF to be stable, we can force this characteristic polynomial to be a Hurwitz type of our choice. If we choose the following Hurwitz polynomial through closed-loop poles placement.

$$P_h(s) = s^{2n-1} + \alpha_1 s^{2n-2} + \alpha_2 s^{2n-3} + ... + \alpha_{2n-2}s + \alpha_{2n-1} \tag{5.14}$$

Then by setting its coefficients to those of the state-dependent ones in (5.12), the following set of $2n-1$ algebraic state-dependent equations containing the filter parameters, at each sampling instant, using the sampled state of the system:

$$\begin{aligned} &f_1(x,d_i,e_j) = \alpha_1 \\ &f_2(x,d_i,e_j) = \alpha_2 \\ &\text{........................} \\ &f_{2n-2}(x,d_i,e_j) = \alpha_{2n-2} \\ &f_{2n-1}(x,d_i,e_j) = \alpha_{2n-1} \end{aligned} \tag{5.15}$$

CHAPTER 6

DC MOTOR AND FRICTION PARAMETER IDENTIFICATION USING TEST DATA IN SIMULATION

To experimentally test the method presented in this work, a robot arm or a motorized simple inverted pendulum is to be employed. The dc motor component is modeled as an LTI system so as to limit the nonlinear dynamics of the system to that in the arm, the simple inverted pendulum. To make the design as realistic as possible, the model of friction in the system is an important factor to consider. Friction in a dc servo motor system is the main challenge to its accurate positioning. If the friction can be accurately estimated, then positioning accuracy can be enhanced by including the friction model. For this reason, before proceeding on with robot arm design in the next chapter, a dc motor and friction parameter identification method is presented in this chapter.

Extensive research literature exists on friction modeling and estimation, the method presented in this chapter is based on the fact that the significant effect of friction in motion systems is to increase damping. The higher the friction in the system, the more damped the system motion. Thus, it is believed that there is some correlation between friction and damping parameters of the system. For example, the extent of friction present in a dc servo positioning system is reflected in the percentage of overshoot and damped frequency of oscillation of its step response. Therefore, by experimentally measuring the under-damped step response of such a system, the percentage of overshoot and damped frequency of oscillation can be estimated. Since these parameters are related to the natural frequency of the system in which the friction exists, they can be used not only to extract the friction but also the transfer function of the system in simulation.

To do this, an experimentally measured step response of the motor in a closed-loop position control system is required. This control system must be established with only a proportional control gain which must be chosen high enough to have the response be under-damped. The response is then used in simulation in which the model parameters of friction are varied until the simulated response fits a real response. When this happens, the simulation model of friction and actual friction whose effect is present in the experimentally measured step response used in the simulation, are matched. Hence, the model represents a true estimate of the friction in the real system of the experiment. The damped frequency and the percentage of overshoot of the simulated response at this point can be used to calculate the damping factor and the natural frequency of the system from which the dc motor parameters can be calculated.

Although the classical model of friction is employed in the scheme presented in this research, the scheme is applicable to any other type of friction models. In fact, the accuracy of the end application is determined by that of the friction model. Hence, a more accurate friction model must be employed in higher precision positioning systems

6.1 Brief Survey of Friction Models, Estimation and Model Identification

Different models of friction were proposed in prior research. In the classical model, friction is defined as a static mapping between velocity and two forces that oppose the velocity. One of these forces is known as coulomb friction and is constant. The second force is known as viscous friction and is proportional to the velocity. One of the earliest friction models is proposed in [9]. This model is essentially the coulomb component plus a second velocity-dependent component with a delay when the velocity direction changes. Then the so-called LuGre model is developed in [2]. It is a model that includes dynamic and static behaviors. The static behavior also known as the Stribeck behavior represents hysteresis. The work in [29] modifies the LuGre model to enhance the static sub-model. In [13], two different friction models known as the bristle and reset integrator models are developed. The former as a more accurate model is based on the macroscopic behavior at the sliding surfaces expressed as snapping spring force elasticity. The reset integrator model is based on the macroscopic level behavior but at the contacting surfaces. Input to the model is the relative velocity between the two contacting surfaces. Its output is compared to an output of a nonlinear block that has the effect of turning-off the input to the integrator under certain conditions. The reset–integrator model uses less computing time than the bristle model. In [18], a method to estimate the coulomb and viscous components of friction in vibrating systems is presented. This approach is based on the observation that, for larger. amplitude of vibrations, the viscous friction component is dominant while the coulomb component is dominant for smaller amplitudes. In [1], a model defined by a total of seven different parameters as required to capture details including pre-sliding and sliding friction effects is presented. In [17], a static friction model that uses the hysteresis and creep behavior to model the pre-sliding behavior of friction as a set of differential equations with elements of elastic behavior, plastic behavior, hysteresis with memory and viscous damping is developed. In [5], the model described in [17] is used to compensate for the tracking error of a gantry stage that is caused by nonlinear friction.

6.2 Brief Survey of dc Servo Motor Parameter Identification

There is an extensive research literature on dc servo motor system identification methods. Most of these studies are based on an open-loop velocity control system configuration because the system tends to be unstable in an open-loop position control configuration. In [31], an impulse response data with voltage as the input and angular speed as output obtained from experimental measurement is used with an autoregressive moving average (ARMA) model with a steepest decent algorithm to minimize the error between the test and the modeled velocities. The model identified is used to predict the angular speed of the original motor system for several input signals with excellent accuracy. The study in [33] uses a Taylor series expansion of the motor constant speed response based on the relationship between electrical time constant, mechanical time constant, Back-emf, motor friction, and the coefficients in the expanded series. Then by sampling the motor speed under constant voltage and fitting the samples to obtain coefficients of the power terms in Taylor series, the electrical time constant, mechanical time constant, back-emf and motor friction are estimated. In [26], Kalman filtering is employed to estimate the dc motor parameters in a experiment by using the discrete measurement of an integrated dynamometer. The work in [23] implements a discrete-time algebraic parameter identification method for dc motor system identification in frequency domain without loss of generality on time domain setting. In [28], a closed-loop velocity-controlled servomechanism using a data acquisition system based on microcontroller for dc servo motor identification is presented.

One important disadvantage of these referenced model identification investigations is that they are carried out for dc motors in a velocity control rather than positioning control systems. This requires that these motors be driven at high speed so as to cover a wide range of speeds needed for valid results. These speed tests are not only less safe and more difficult to conduct than positioning-based tests, but also more expensive to conduct. A motor velocity sensor or tachometer is much more costly than a position sensor. A simple analog potentiometer can be used for the position sensing. The method in this work is for positioning end applications that do not have to use the usual velocity mode identification. Though the work is based on the classical friction model, other model types used in the references are applicable.

In fact, the accuracy of the positioning applications is determined by the fidelity of the friction model employed.

6.3 Theory Behind the Method

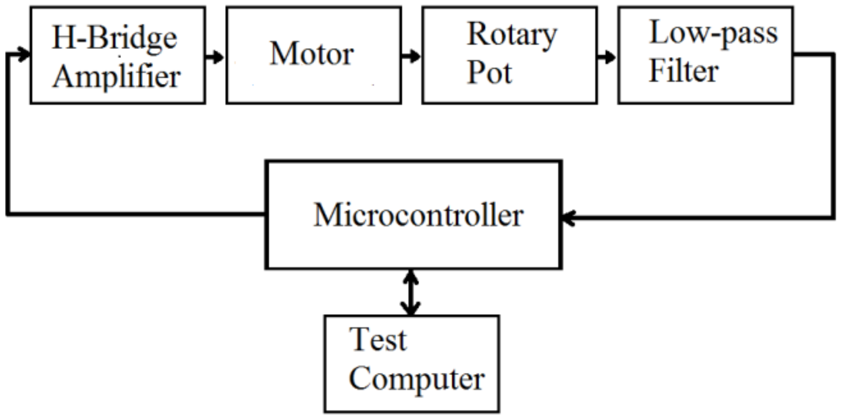

Figure 6.1 DC motor parameter and friction identification system.

A dc servomotor positioning system depicted in Figure 6.1 is a 3rd order dynamic system with a pole at zero and an electrical and mechanical pole. The electrical pole is a function of coil resistance and inductance. The mechanical pole is determined by the inertia of the rotor and load on the motor, coil resistance, torque and back-emf constants. Hence, to define the open-loop transfer function of such a system requires two non-zero poles and a dc gain factor as follows:

$$G(s) = \frac{A_0}{s(s+\omega_m)(s+\omega_e)} \tag{6.1}$$

where A_0 is the dc gain factor, ω_m is the mechanical pole and ω_e is the electrical pole. This system is unstable in open-loop and thus is stabilized with a simple proportional gain K_0 resulting in closed-loop system with the following transfer function:

$$H_x(s) = \frac{K_0A_0}{s(s+\omega_m)(s+\omega_e)+K_0A_0} \tag{6.2}$$

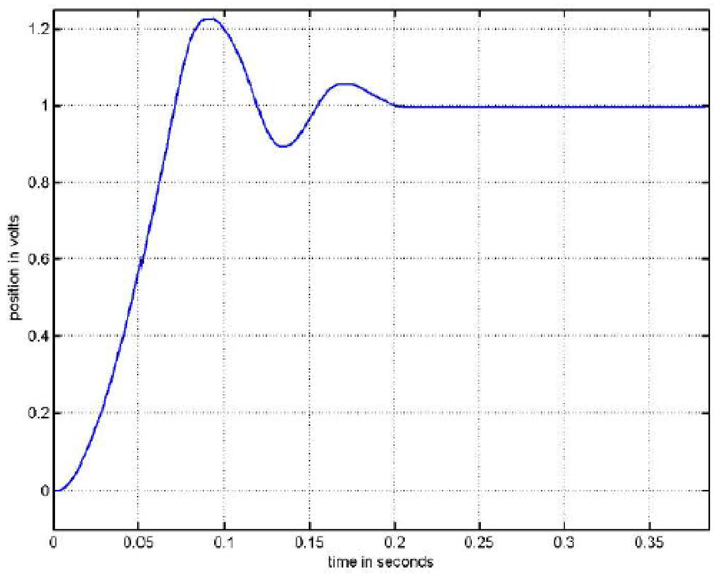

Figure 6.2 Under-damped step response measured for a real test.

In an experiment performed with a digital controller implemented on the system in Figure 6.1, a large proportional gain K_0 is chosen to produce the under-damped step response shown in Figure 6.2. The transfer function of the closed-loop system that generates this response can be represented as follows:

$$H_m(s) = \frac{\omega_0 \omega_n^2}{(s+\omega_0)(s^2 + 2\zeta\omega_n s + \omega_0\omega_n^2)} \tag{6.3}$$

where ζ is the damping factor, ω_n is the natural frequency of oscillation related to this closed-loop system ω_0, is the analog low-pass anti-aliasing filter bandwidth chosen to implement the proportional controller digitally such that $\omega_0 > \omega_n$ and $\omega_0 > \omega_e$. ω_0 and K_0 are known parameters used in the experiment. ω_n and ζ, on the other hand, can be calculated from the measured step response in Figure 6.2. The rest of the parameters required to identify the transfer function of this closed-loop system described with Eq. (6.3) can be calculated according to the following equations:

$$A_0 = \omega_0 \omega_n^2 \tag{6.4}$$

$$\omega_m^2 - (2\zeta\omega_n + \omega_0)\omega_m + (2\zeta\omega_n\omega_0 + \omega_n^2) = 0 \tag{6.5}$$

$$\omega_e = 2\zeta\omega_n + \omega_0 - \omega_m \tag{6.6}$$

The percentage of overshoot, R_t and the damped frequency of oscillation, ω_t, of the this response are easily calculated, which can then be used to calculate the natural frequency and damping factor characteristics of closed-loop system described with Eq. (6.3) according to the following formulas:

$$\zeta = \frac{\log R_t}{\sqrt{(\log R_t)^2 + \pi}} \tag{6.7}$$

$$\omega_n = \frac{\omega_t}{\sqrt{(1 - \zeta^2}} \tag{6.8}$$

The MATLAB script for this is available in Appendix A. Clearly, these relationships shown in Eqs. (6.7) and (6.8) demonstrate that friction in a motorized system partly determines its natural frequency and damping factor.

6.4 Extracting Friction and Motor Parameters from Test Data via Simulation

Figure 6.3 is the simulation model implemented in MATLAB/SIMULINK for the closed-loop system of the system described with Eq. (6.3). The friction model employed is the classical one. The purpose of the simulation to fit these parameters by iteration so as to imitate the real experiment. Hence, we use the same proportional gain control, $K0$, as used in the experiment and the resulting closed-loop system is simulated for its step response by iterating the unknown parameters until the simulated step response fits the experimentally measured one as shown in Figure 6.4.

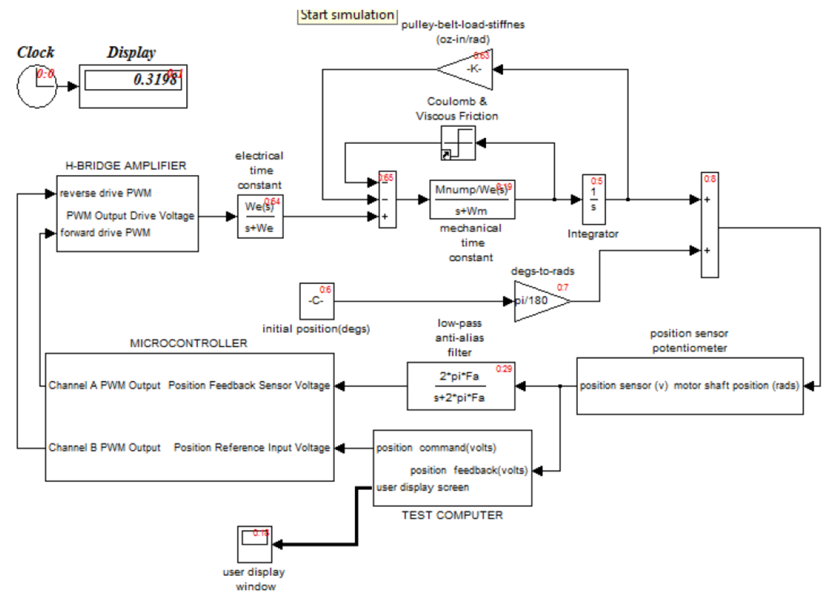

Figure 6.3 SIMULINK model for dc motor and friction identification.

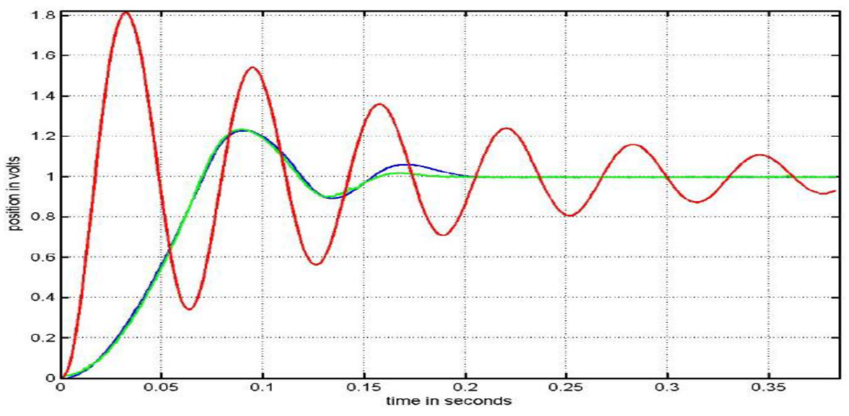

Figure 6.4 Real and simulated step response for real with and without friction.

There are three responses shown in this figure. The highly oscillatory one is the first iteration of the simulation. This simulated response is what the real response would have been if there were no friction in the system; thus in its simulation both the coulomb and viscous parameters of the friction model are set to zero, i.e., $\alpha = 0$ and $\beta = 0$. None of the damping seen in this response is due to friction as described by the classical model. The percent of overshoot and damped frequency calculated from this response respectively are $R_t = 81.52$ and $\omega_n = 100.3$ rad/sec. One of the remaining two responses in the figure is the experimentally measured one. The last response in the figure is the simulated fit for the experimentally measured one. Its percentage of overshoot and damped frequency calculated respectively are $R_t = 23.29$ percent and $\omega_n = 66.84$ rad/sec. It occurs when the friction model fits the real friction in the system at the following values $\alpha = 5.5$ oz.in for the coulomb component and $\beta = 0.005$oz.in/rad/sec for the viscous component. These are estimates of the parameters of the real friction of the system during the experiment. With these values determined the open-loop transfer function parameters of the dc servo motor is calculated according to (6.1), (6.5) and (6.6) to obtain $A_0 = 734820$, $\omega_m = 22.31$ rad/sec and $\omega_e = 1102.9$ rad/sec, respectively.

6.5 Validating Identified Friction and Motor Parameters

If the identified friction and motor parameter are correct, then a control system based on them should achieve desired performance in simulation and real experimental test. To show that this is the case, a lead compensating filter based on these parameters and resulting design is adopted and designed. The step response of the closed-loop control system to be designed must meet the following requirements: 1) Rise time of no more than 100 milliseconds, 2) Percent of over-shoot not to exceed 1 percent and 3) Steady state error of no more than 1 percent.

6.5.1 Designing a Lead Compensating Controller

The 100 millisecond rise time requirement on the step response is translated to the closed-loop system bandwidth of ω_b rad/sec and a unity gain crossover frequency of ω_b rad/sec which are related as follows [32]:

$$\omega_x = 0.635\omega_b \tag{6.9}$$

Note that this closed-loop bandwidth specification is based on the ideal LTI dc motor model with parameters identified in the closed-loop without friction. Thus, by considering the effect of the friction identified, we find that the bandwidth falls short of the desired value leading to a longer rise time. This loss of bandwidth is an effect of additional damping introduced into the system by the friction. Therefore, the desired bandwidth or rise time specification must include a margin to the extent of friction in the system to offset the damping effect of that friction. This can be determined in simulation with the identified friction model. The desired closed-loop bandwidth must be specified to be a bit higher or equivalently the rise time for its step response must be specified to be a bit lower than the actual one used in compensator design. The same is true when the desired percentage of overshoot and the damping factor are handled.

A lead compensation control design is a frequency response-based design scheme in which the open-loop frequency response of the system must be compensated by adding in series with it a lead filter such that overall gain magnitude of the combination reaches unity or zero dB at a phase angle that differs from 180 degrees by a desired phase margin. Hence, the purpose of the lead filter is to provide some additional gain magnitude and phase angle required so that the compensated open-loop transfer function can have a gain of unity or zero dB and desired phase margin at crossover frequency; frequency at which the gain magnitude crosses the unity gain or zero dB line. Consider a first-order lead filter required in this design which has the following transfer function:

$$F(s) = \frac{D_0 \omega_p (s + \omega_z)}{\omega_z (s + \omega_p)} \tag{6.10}$$

where D_0 is the dc gain contribution of the lead filter so as to meet a net gain of unity at the desired crossover frequency of ω_x and ω_p is the pole of the filter pole and ω_z is the zero of the filter. The gain contribution is computed as:

$$F(s) = \frac{1}{G(j\omega_x)} \tag{6.11}$$

where $G(j\omega_x)$ is the magnitude of frequency response of the dc motor, as described by its transfer function in (6.1), evaluated at the unity crossover frequency ω_x Similarly, the phase angle contribution required for system to meet the desired phase margin at this crossover frequency is:

$$\phi = 180 + \theta(\omega_x) - \gamma - 5 \tag{6.12}$$

where $\theta(\omega_x)$ is the net phase angle of the open-loop frequency response of the system in (6.1), at this crossover frequency and γ is the desired phase margin. The required pole and zero of the lead filter are calculated as follows:

$$\omega_p = \omega_x \sqrt{\frac{1 + \sin(\frac{\pi\phi}{180})}{1 - \sin(\frac{\pi\phi}{180})}} \tag{6.13}$$

$$\omega_z = \frac{\omega_x^2}{\omega_p} \tag{6.14}$$

With the identified parameters for the motor and friction, the following are required values of the lead compensating filter necessary to meet the specified performance requirements: $D_0 = 26.93$, $\omega_p = 348.3$ rad/sec and $\omega_z = 38.4$ rad/sec which are translated to a continuous-time lead filter transfer function given as;

$$F(s) = \frac{244.3(s + 38.4)}{s + 348.3} \tag{6.15}$$

With a zero-order-hold sampler (ZOH) at one millisecond sampling period, the equivalent discrete-time filter has the following z-transform transfer function:

$$F(z) = \frac{244.1z - 236.1}{z - 0.7059} \tag{6.16}$$

The resulting digital filter or difference equation has the following coefficients: $a_1 = 244.1$, $a_2 = -236.1$, and $b_1 = -0.7059$

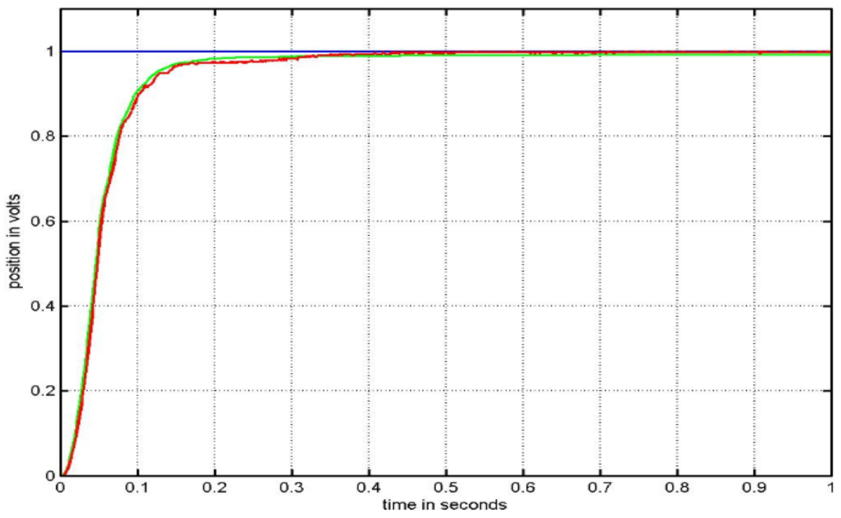

Figure 6.5 Real and simulated response of the closed-loop system.

Figure 6.5 shows a step reference input, its simulated and the real experimental responses for the designed closed-loop dc motor position control system based on the parameters identified for the motor and its friction torque. Clearly, both the simulated and real responses which are almost indistinguishable and meet the 100 millisecond rise time, the 1 percent steady state error and the 1 percent overshoot performance requirements. Thus, the identified parameters of the motor and its friction torque are validated. Refer to MATLAB source code used for the validation in Appendix A.

CHAPTER 7

PSC DESIGN AND SIMULATION OF A ROBOT ARM

The objective in this chapter is to design and simulate the PSC controller for control of a robot arm system. Details of the design are presented after the system can be built and experimentally tested. A robot arm is actually a motorized simple inverted pendulum. The SDTF model of the simple inverted pendulum has already been defined and analyzed in the previous chapters. A scheme to acquire an LTI model of the dc motor and its friction has also been presented in Chapter 6. What is left to define the robot arm system is putting the two main model components together. Figure 7.1 is a SIMULINK model of the complete system. The PSC block is the

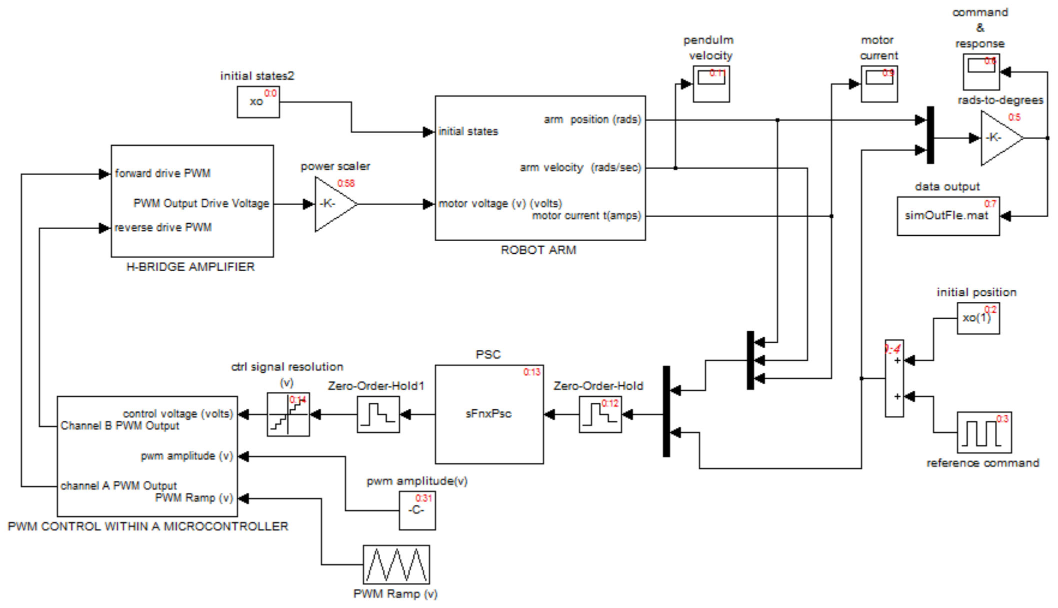

Figure 7.1 SIMULINK Model of robot arm control with PSC.

SDTF-based controller that is implemented digitally inside a microprocessor. Its implementation code is available in Appendix B. Its output is the control signal that drives the PWM generator block input during every sampling period. The output of the PWM generator, in turn, drives the H-bridge and finally the H-bridge power amplifier output after additional gain is used to drive the motor within the robot arm block to position the arm as desired. The ZOH and analog-to-digital Converter (ADC) blocks at the input and output of the PSC block provide periodic sampling or data conversion such that the PSC can only run periodically at the start of every sampling period as required by its design. In other words, only the PSC block in Figure 7.1 is implemented in software and runs periodically or in discrete-time while the rest of blocks are implemented in hardware or they run in continuous-time as required in a real system.

The main system component of Figure 7.1 is the robot arm block which is shown separately in Figure 7.2. It is the nonlinear system under investigation. It consists of a simple inverted pendulum whose dynamics is nonlinear. The pendulum is coupled at 90 degrees to the shaft of a dc motor which is modeled with LTI dynamics as shown in the model in Figure 7.2. The mass of the pendulum, or its load, is usually represented physically by a bob attached to its free end. In this model, a five-turn rotary potentiometer/tachometer is assumed to be attached to the other end of the motor shaft such that the arm of the pendulum is suspended vertically between them. These are sensors required to measure the arm position/velocity, two of the state variables of the system required for evaluating its SDTF. The bandwidth of these sensors is much higher than that of the motor and the pendulum. For this reason, they are modeled as pure gains therefore, not explicitly shown in the model in Figure 7.2. At the coupling joint between the motor shaft and pendulum arm and between the shaft and potentiometer/tachometer sensors are static and viscous friction which is described with the classical model. Part of the viscous friction is from air in which the arm swings.

The H-bridge component in Figure 7.1 is the power amplifier required to amplify the lower power control voltage to drive the motor. An H-bridge rather than a linear amplifier is required because the lower power control voltage from the microprocessor is pulse-width modulated. There are two complementary channels A and B of voltage to drive the motor forward or counterclockwise and backward or clockwise respectively. The input voltages are pulse-width modulated at about 15 kilohertz and at amplitude of five volts. Thus, the two complimentary output voltages are also at the same frequency but at much higher voltage as required to provide the control torque needed to drive the motor forward or backward. The motor is loaded by its own inertia, friction and inertia of the arm. Clearly the higher these loads on the motor the larger the required control torque. Internal implementation details of the amplifier are shown in Figure 7.3. The amplifier is represented as a pure gain in the overall differential equation of the system because its bandwidth is far higher than those of the robot arm main system. Clearly, the representation in Figure 7.3 is a realistic model required for realistic simulation results.

The PWM generator component in Figure 7.1 is function implemented internal to the microprocessor where the PSC controller is implemented. It is the control voltage from the PSC. The implementation details as shown in Figure 7.4 are required to obtain a realistic simulation because that is how the device operates in a real system. It consists of a comparator that compares an input voltage to a triangular pulse of amplitude equal to the maximum level of the control voltage. One of its output channel's voltage level will switch from low to high level whenever the triangular ramp level reaches the control input voltage level. Simultaneously, the opposite or complimentary output is generated on the other channel. The two complimentary outputs become inputs to the H-bridge block where they are amplified to drive the motor in the robot arm block.

The PSC block component in Figure 7.1 is the digital output feedback control required and designed based on the SDTF model to control the robot arm. It is a software routine that runs at the start of each sampling period. The state of the system, pendulum position, velocity and motor current are sampled into this at the start of the sampling period. These sampled data are then used to evaluate the current SDTF of the system. The desired closed-loop pole locations of the system are determined in advance. These together with the evaluated SDTF are passed to a computation routine to solve for parameters of the required compensating filter. Once the filter parameter solutions are returned they are used to calculate the current control output which is then appropriately scaled and sent out into the PWM generator where the complimentary PWM drive voltages at lower power levels are generated.

7.1 System Description and Simulation Model

Figure 7.1 is the simulation model for the robot arm. Accurate of design based on this model is determined by details included in the model. A common source that affects dynamic system accuracy is disturbance inputs like friction. The model shown Figure 7.1 includes friction within the motorized pendulum block. The complete system consists of the robot arm component within which are the dc motor and the inverted pendulum arm. The sub- components of this block are shown in Figure 7.2. Connected to terminals of the motor is an H-bridge power amplifier. Inside it are the component details shown in Figure 7.3. Connected to the H-bridge block is the PWM control block inside a microprocessor. The implementation sub-components of this block are shown in Figure 7.4

7.2 Design Model of the Robot Arm

The robot arm system shown in Figure 7.1 is a third order nonlinear dynamic system. The dc motor is an actuator that provides control torque through its input voltage to move the arm to desired positions. Therefore, the torque developed in the motor at any time must overcome the following friction, torque and moment of inertia:

1) Static friction from F_m and F_p,

2) Viscous friction from B_m and B_p,

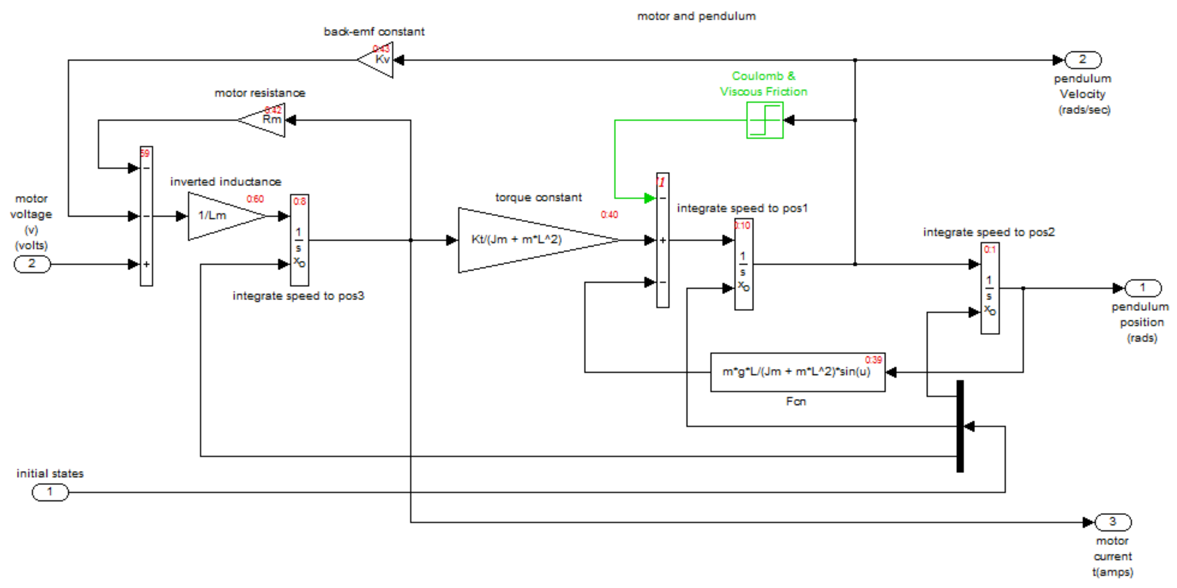

Figure 7.2 SIMULINK model of the motorized arm.

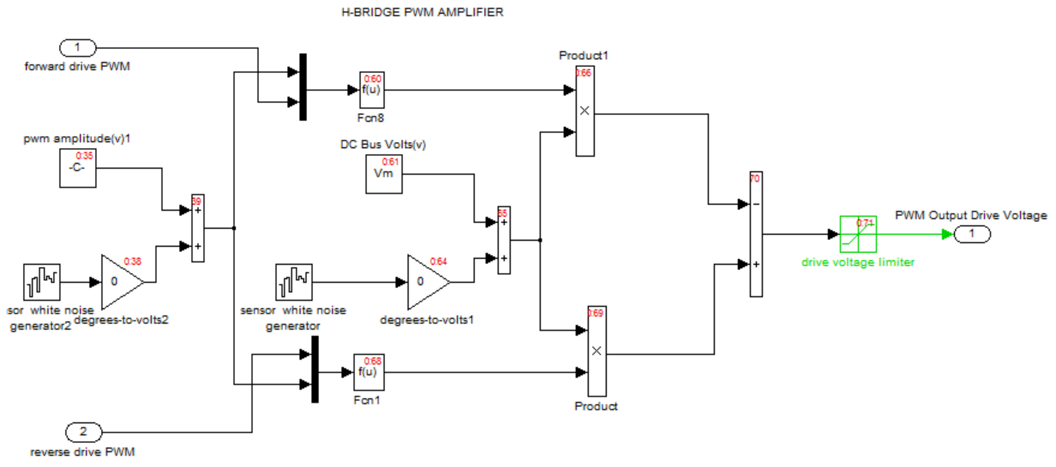

Figure 7.3 SIMULINK model of H-bridge power amplifier.

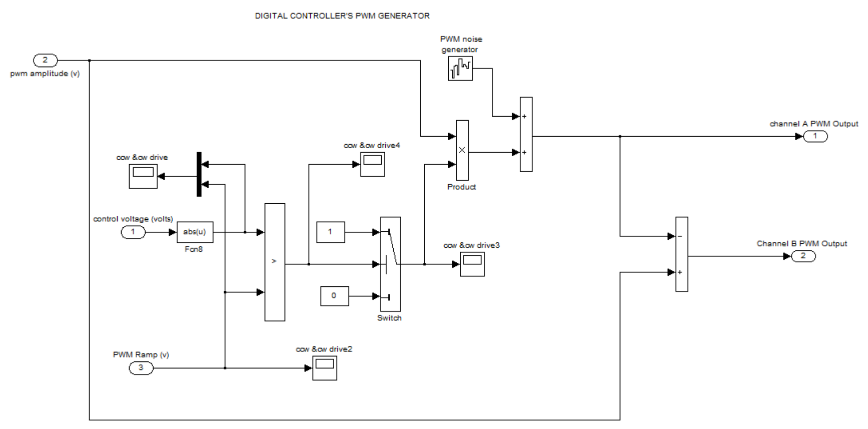

Figure 7.4 SIMULINK model of the PWM generator within a microcontroller.

3) The torque acting on the mass, m due to gravity that can be described by the following differential and

4) moment of inertia of the combined motor and arm around the pivot.

The result is the following SISO control-affine nonlinear equation:

$$(J_m + mL^2)\frac{d^2y}{dt^2} = K_t i - (B_m + B_p)\frac{dy}{dt} - mgL\sin(y) - F_m - F_p \tag{7.1}$$

where i is the motor current and y is the angular position of the arm, L is the length of the arm, g is the acceleration due to gravity, J_m is the motor inertia about the pivot, m is the mass of the robot arm, B_p and B_m are the viscous friction coefficients of the arm and the motor respectively, F_p and F_m are coefficient of the arm and motor static friction respectively and K_t is the motor torque constant. The motor current is related to its resistance R_m, inductance L_m and input voltage u as follows:

$$u = R_m i + L_m\frac{di}{dt} + K_v\frac{dy}{dt} \tag{7.2}$$

With the arm position, velocity and the motor current chosen as the state variables of this system as follows: $x_1 = y$, $x_2 = \frac{dy}{dt}$ and $x_3 = i$ such that $\dot{x_1} = \frac{dy}{dt} = x_2$

and $\dot{x_2} = \frac{d^2y}{dt^2}$ and $\dot{x_3} = \frac{di}{dt}$, the following is the state model representation the of system:

$$\begin{aligned}
\dot{x}_1 &= x_2 \\
\dot{x}_2 &= -\frac{mgL}{J_m + mL^2}\sin(x_1) - \frac{B_m + B_p}{J_m + mL^2}x_2 + \frac{K_t}{J_m + mL^2}x_3 \\
\dot{x}_3 &= -\frac{K_v}{L_m}x_2 - \frac{R_m}{L_m}x_3 + \frac{u}{L_m} \\
y &= x_1
\end{aligned} \tag{7.3}$$

The above model ignores static friction and other hard nonlinearities. As a result the final design must be tweaked through simulation. The SDTF design model, based on the following CCFD SDC is:

$$A(x) = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & \frac{K_t}{Jm+m*L^2} - \frac{\frac{mgL}{Jm+m*L^2}\sin x_1 + \frac{Bm+Bp}{Jm+m*L^2}x_2}{x_3} \\ 0 & \frac{-K_v}{Lm} & \frac{-R_m}{Lm} \end{pmatrix}$$

$$b(x) = \begin{pmatrix} 0 \\ 0 \\ \frac{G}{L_m} \end{pmatrix}$$

$$c(x) = \begin{pmatrix} 1 & 0 & 0 \end{pmatrix}$$

which leads to the following SDTF model:

$$G(x,s) = \frac{\frac{W(K_t x_3 - (B_m+B_p)x_2 - mgL\sin x_1)}{L_m(mL^2+J_m)}}{s^3 + \frac{R_m}{L_m}s^2 + \frac{(K_t K_v x_3 - (B_m+B_p)K_v x_2 - mgLK_v \sin x_1)}{L_m(mL^2+J_m)x_3}s} \tag{7.4}$$

where W is total gain and scale factor from the output position to the output of the H-bridge power PWM amplifier

The robot arm system as represented by SDTF has no zero. Because the system has additional equilibrium points that are not at the origin, its gain factor is state-dependent. Its poles are also state-dependent. The non-zero equilibrium points of this system are the roots of the state-dependent gain factor: $\frac{W(K_t x_3 - (B_m + B_p)x_2 - mgL \sin x_1)}{L_m(mL^2 + J_m)}$

which has the general form: $a_3x_3 + a_2x_2 + a_1 \sin(x_1)$

with its coefficients a_1, a_2 and a_3 as a function of the system parameters:

$m, W, K_t, K_v, R_m, L_m, B_m, J_m, L$ and B_p.

The robot arm is therefore, locally and not globally controllable. It must be kept away from the uncontrollable equilibrium points. If these parameters are chosen such that the non-zero equilibrium point occurs closed to the one at the origin, and then control of the robot arm becomes impossible as it is easier to get into these points near the origin. Thus, the non-zero equilibrium points where this gain factor becomes zero and where the robot arm system losses controllability are the solutions of the equation:

$$a_3x_3 + a_2x_2 + a_1 \sin(x_1) = 0 \tag{7.5}$$

where $a_3 = \frac{WK_t}{Lm(mL^2+J_m)}, a_2 = -\frac{(B_p+B_m)}{Lm(mL^2+J_m)}$ and $-\frac{mgL}{Lm(mL^2+J_m)}$

7.2.1 PSC Design

The PSC design is driven by desired closed-loop system performance. It's step response must meet:

1) Rise time of no more than 1 second,

2) Per cent of over-shoot not to exceed 1 percent and

3) Steady state error of no more than 1 percent

The complete PSC design as implemented using MATLAB script is available in Appendix B.

Because this is a third order system, the compensating filter required in the PSC should be of second order resulting in a fifth order closed-loop system. Thus, five closed-loop poles have to be placed during the PSC implementation. These requirements need to be converted into closed-loop system design parameters such as response speed or bandwidth and phase margin which are related to the closed-loop pole location. The desired speed of response is related to the closed-loop bandwidth ω_b. The desired overshoot, M_p in the step response of the closed-loop system is related to the damping factor, ζ as follows:

$$\zeta = \frac{0.01M_p}{\pi}\sqrt{\frac{1}{1+(\frac{\log 0.01M_p}{\pi})^2}} \tag{7.6}$$

For one percent overshoot specification, the damping factor is about $\zeta = 0.82609$. With a closed-loop bandwidth specification ω_b, the unity-crossover frequency ω_x required for the compensated open-loop system to achieve this according to [32] is given by (6.9). The desired natural frequency, ω_n, of the closed-loop system is related to the crossover frequency ω_x according to:

$$\omega_n = \frac{\omega_x}{\sqrt{\sqrt{4\zeta^4+1}-2\zeta^2}} \tag{7.7}$$

The damped frequency of oscillation ω_d of the closed-loop step response is related to its natural frequency, under assumption of an equivalent second system response, as:

$$\omega_d = \omega_n\sqrt{1-\zeta^2} \tag{7.8}$$

Hence, for the closed-loop system to satisfy the desired performance specifications in terms of this natural frequency, ω_n, damped frequency of oscillation ω_d, and damping factor, ζ, two of its poles must be placed as dominant pair meet to this second order performance requirement. That is the remaining three poles are to be placed far away such that their influence on the closed-loop response is negligible as follows:

$$\begin{aligned}
p_1 &= -\zeta\omega_n + j\omega_d \\
p_2 &= -\zeta\omega_n - j\omega_d \\
p_3 &= -10\zeta\omega_n + j10\omega_d \\
p_4 &= -10\zeta\omega_n - j10\omega_d \\
p_5 &= -10\zeta\omega_n - j\omega_d
\end{aligned} \tag{7.9}$$

These desired closed-loop poles are then passed to the compensating filter design routine listed in Appendix B. This routine implements a compensating filter as described by PSC design procedure.

7.2.2 Simulation and Results

Hard nonlinearities, like actuator saturation (current limits etc.) and stiction, backlash and sensor deadzone are present in a nonlinear regardless of the design model. If it were not for these nonlinearities, one would meet performance requirements as designed without the need for simulation provided the design model is accurate. Because of these nonlinearities, the design must be iterated with simulation until the simulated performance matches the desired one.

The SIMULINK model of the simulation is shown in Figure 7.1. The simulation is run by using the MATLAB script in Appendix B. Starting with a lower closed-loop bandwidth, the PSC is designed and simulation is run for the design. If the simulated response performance does not meet the desired performance, then the closed-loop bandwidth is increased a bit and a new PSC is designed and the simulation repeated. PSC design followed by simulation of its response for increased closed-loop bandwidth is iterated until the simulated step response meets the desired performance. The entire design-and-simulate process until desired performance is met is automated with MATLAB script in Appendix is Appendix B.

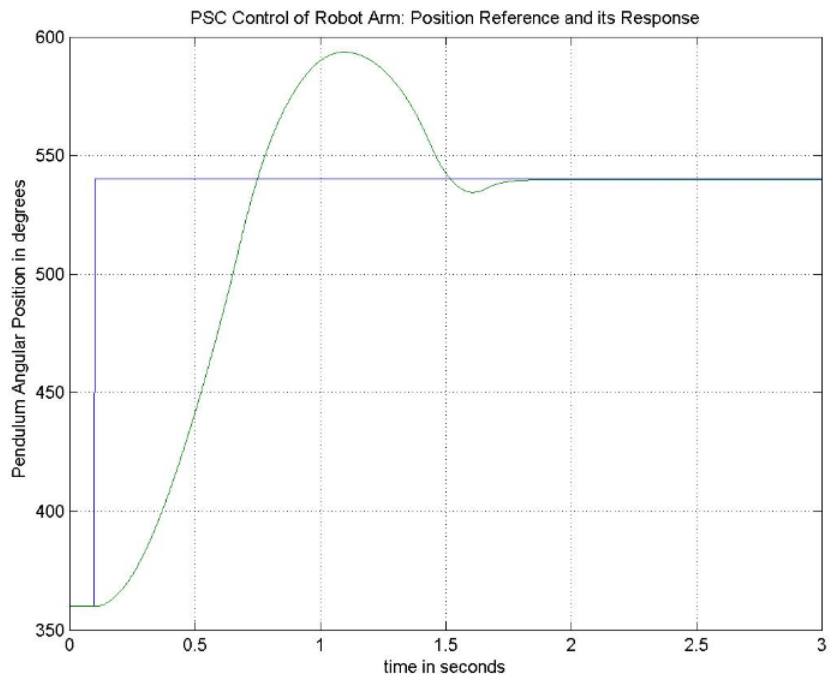

Figure 7.5 Step response of the robot arm with PSC.

The simulation result shown in Figure 7.5 is for the five closed-loop system poles placed at: $-576.25 \pm j393.11, -2881.3e \pm j1965.6$ and -2881.3. The arm is moved from a stable vertically down position to an unstable vertically up position in approximately one second, with a large overshoot and almost no steady state error. Compare this response to the one in Figure 7.6 where closed-loop poles are placed at fives times higher the frequencies as: $-2881.3e+003 \pm j1965.6, -14406 \pm j9827.8$ and -14406. This response clearly has no overshoot. The higher closed-loop poles produce more damping to prevent excessive overshoot. The PSC design can thus achieve a wide range of performances. Note that while the PSC design for the response in Figure 7.6 meets all the performance requirements of one percent over shoot, one second rise-time and less than one percent steady state error, the design for response in Figure7.6, on the other hand, meets all but the overshoot requirement.

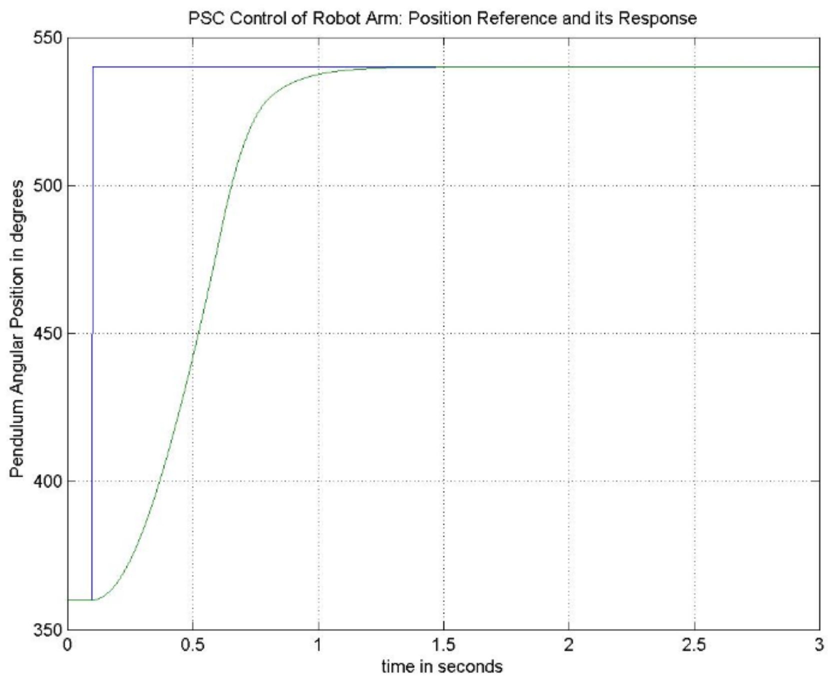

Figure 7.6 Step response of the robot arm with PSC.

CHAPTER 8

CONCLUSION

State-dependent transfer function (SDTF), a new approach to describing control system model for a single-input single-output (SISO) control-affine nonlinear system, has been introduced. This new modeling paradigm is an extension to the state-dependent Ricatti equation (SDRE) approach to feedback control of this class of nonlinear systems. A concept which states that a control-affine differential equation describing a nonlinear system can be expressed in a linear form with state-dependent coefficients provided the origin is an equilibrium point. These coefficients therefore, vary from point to point in the state space as the system evolves along a given trajectory. If the state of a system can be measured or estimated at each point along the trajectory, the system can be described by a continuum of linear differential equations along its trajectory. In other words, the control-affine nonlinear system is linear pointwise and hence, it can be described with this variable parameter model, a pointwise transfer function or an SDTF.

The implication of this new perspective is that, the gain factor, the zeros and the poles the fixed or constant parameters used to define the transfer function of LTI dynamic systems for a control-affine nonlinear system at least one of these parameters will be variable or state-dependent. In other words, LTI systems have stationary parameters, while control-affine nonlinear systems have non-stationary parameters.

It has been demonstrated that whether design of control is SDRE-based or SDTF-based, the CCFD representation is the most appropriate way to express the nonlinear control-affine differential equations. This is because this representation is explicit in terms of controllability of a nonlinear system, i.e., it does not hide the original system's controllability. It has also be shown that control-affine nonlinear system are uncontrollable, i.e., controllability weight, the gain factor of the SDTF, is zero at non-zero equilibrium points.

The SDTF concept has been applied to the pointwise stabilizing controller (PSC). This is an output feedback controller that is implemented digitally, online, while the system is running in real time. The steps involved in the design and implementation of this controller on-the-fly are first, to evaluate the SDTF of the system at each sampling instant by using the measured or estimated values of the state of the system. Then search for a compensating filter based on the evaluated open-loop SDTF of the system. The filter search algorithm is based on stable closed-loop pole assignment as determined from desired control system performance specification. This algorithm guarantees that the resulting closed-loop control system, at each sampling instant, is stable. However, in cases where a compensating filter cannot be found at a given sampling instant, the filter designed from the previous instant can be used.

The SDTF model and the PSC based on it have been validated via simulation on robot arm control. This system consists of dc motor actuator that has been modeled as an LTI sub-system. To make this system as realistic as possible, a model of friction has also been included. An identification method has therefore, also been proposed and tested experimentally on how parameters of the motor and its friction can be determined for use in the PSC design and simulation for the robot arm. The identification method is general enough to be applied to any dc servo system with friction. The control simulation results for the robot arm have successfully validated the new SDTF model and its PSC. The approach, however, has a few limitations. It takes more signal processing resources to implement it than methods based on fixed parameter models as in design by linearization. The design takes place online and in real-time, and is repeated at the start of every sampling period which requires high speed processing. Also the addition of a compensating filter means that the closed-loop system is of much higher order than the open-loop system. The processing resource requirement however, may not be major problem nowadays, due to proliferation of inexpensive high speed microprocessors. Another limitation has to do with state estimation. If the system is not observable for estimation then its state vector be measured to evaluate its SDTF. Measuring all members of the state can make the implementation more expensive.

Contributions of this Research:

1. An invention of a nonlinear system state-dependent controller based on the new nonlinear transfer function, a variable parameter model. This feedback control scheme, PSC, is very suitable to system with varying parameters or even varying loads. With this method, the problem of model parameter variations during real control systems operation is addressed;

2. An invention of a nonlinear frequency domain analysis tool, the first ever, that shows the link between nonlinear system stability and its controllability and also for investigating relationship between its controllability and its nonzero equilibrium points;

3. Answering why not every state-dependent coefficients, SDCs, can be chosen as parameters for defining SDTF and SDRE models and why the controllable canonical form-dependent SDCs preserve controllability property of the original nonlinear system; and

4. An introduction of a simulation-based method for identifying parameters of a dc motor and its friction.

Suggestions for Future Research:

1. Build and test experimentally the robot arm control system designed and simulated in this research;

2. Apply SDTF model and PSC experimentally on a toy aircraft control;

3. Investigate possibility of extending the new method to more general nonlinear systems in which the control input;

4. Investigate possibility of extending the approach to multiple-input multiple-output nonlinear systems based on state-dependent state feedback control through pointwise pole placement.

APPENDIX A

DC MOTOR AND FRICTION IDENTIFICATION

A.1 MATLAB Script for dc Motor and Friction Identification from Real Test Data

The following MATLAB code runs the SIMULINK model of the closed loop dc motor positioning system to fit the step response of a real system. For each simulation iteration, the friction model parameters are adjusted until the simulated step response fits the real response. At that point the simulation model for both the friction and the motor are estimates of their real ones captured in the experimental response. The real response is read from the file "response20.txt". It is an under-damped step response of the closed loop dc servo motor positioning control system modeled as by a 3rd order transfer function. It is scaled and used to compute its percent of overshoot and damped period of oscillation from which the closed loop transfer function of the real system is fit. A step response of the fit transfer function is computed analytically for comparison with the real one as validation. The validated closed loop transfer function is then used in simulation to estimate friction in the real system. It is also used to compute the open loop transfer function of the dc motor.

```
format short e;
%global Ts Acf Bcf Ccf Dcf
global POSITION_ACTUATOR_VOLTS_PER_COUNTS
global POSITION_ACTUATOR_VOLTS_PER_COUNTS
global POSITION_SENSOR_VOLTS_PER_COUNTS
global POSITION_ACTUATOR_SIGNAL_RANGE_COUNTS
global PGAIN

% Friction is the most significant factor
% a DC servomotor position control system will
% exhibit non-linear behavior. The non-linear
% effect of friction in behavior of otherwise
% a linear behavior of an LTI DC servomotor
% position control system can be measured by
% two parameters defined in terms the damped
% period and the percent overshoot of the step
% response of the system within which the
% friction is present respectively

% The first factor is the friction damped period
% factor, defined as the ratio of damped periods
% of the step responses of the LTI system without
% friction and the one that includes friction. The
% more friction present in the system the lower
% this ratio and vice versa.
FRICTION_DAMPING_FREQUECY_REDUCTION_COEFFICIENT = 1.5;

% The first factor is the friction overshoot factor,
% defined as the ratio of percent of overshoots of
% the step responses of the LTI system without
% friction and the one that includes friction.
% The more friction present in the system the
% higher this ratio and vice versa.
FRICTION_OVERSHOOT_INCREASE_COEFFICIENT = 3.5;

% Thus by measuring the damped period and overshoot
% in the real step response of a system with friction,
% iterating friction models can be added to the LTI
% system during step response simulations until
% the simulated step response and real response for
% the system that includes the friction correlates at
% which point the resulting friction model will be
% the true model of the friction present in the system.
% The simulation can then be run one more time without
% the friction model to determine the damped period
% and percent overshoot so as to characterize the
% friction present by computing these 2 ratios.
% Dynamic, motor viscous damping constant
% (oz.in/rad/sec)
% decreasing coeffViscousFriction decreases rise time
% (fast or raises bandwidth) increasing increases
% rise time (slow or lowers bandwidth)
coeffViscousFriction  = 0.004;

% Static motor friction, (oz.in)
coulombFriction = 5.5;

% Dynamic motor torque load stiffness, (oz.in/rad)
dynamicLoadStiffness = 0.001;

% Motor shaft-to-sensor backlash (amount of play)
% in degrees
Bkd = 3;

% Because there exist experimental step response
% of the closed loop system over time interval of 512
% millisecond we shall run simulation during each PID
% designs iteration over 512 millisecond for compare
SimTime = 0.383;

% PWM frequency and period
Fpwm = 15625;
Tpwm = inv(Fpwm);
Wpwm = 2*pi*Fpwm;
```

```
% Sampling frequency used for the plant's model
% acquisition
Fs = 1000;       % Hz or samples/sec
Ws = 2*pi*Fs;     % rads/sec
Ts = inv(Fs);     % sec

Vm = 24;
% Antialiasing filter cutoff frequency that is more
% than 10 times the desired closed loop bandwidth is
% used in (aid.c) so as to insure no interaction between
% the higher frequency dynamics of the filter and the
% lower frequency dynamics of the closed loop plant
Fa = 177;          % Hz
Wa = 2*pi*Fa;     % rads/sec

INITIAL_POSITION_COMMAND_DEGREES     = 900;
POSITION_COMMAND_DEGREES              = 1900;
% experimental value used
% Total motor displacement per each sensor voltage
% supply swing
POSITION_SENSOR_SIGNAL_RANGE_DEGREES  = 3600;
POSITION_SENSOR_SIGNAL_RANGE_VOLTS    = 5;
POSITION_SENSOR_SIGNAL_RANGE_COUNTS   = 1024;
% PIC18Fxxx micro's input analog-to-digital resolution

POSITION_ACTUATOR_SIGNAL_RANGE_VOLTS  =
POSITION_SENSOR_SIGNAL_RANGE_VOLTS;
POSITION_ACTUATOR_SIGNAL_RANGE_COUNTS = 768;
% PIC18Fxxx micro's, output PWM resolution for 5 volts

POSITION_SENSOR_VOLTS_PER_DEGREE =
     POSITION_SENSOR_SIGNAL_RANGE_VOLTS/
                  POSITION_SENSOR_SIGNAL_RANGE_DEGREES;

POSITION_SENSOR_VOLTS_PER_COUNTS =
   POSITION_SENSOR_SIGNAL_RANGE_VOLTS/
                  POSITION_SENSOR_SIGNAL_RANGE_COUNTS;

POSITION_ACTUATOR_VOLTS_PER_COUNTS =
    POSITION_ACTUATOR_SIGNAL_RANGE_VOLTS/
                   POSITION_ACTUATOR_SIGNAL_RANGE_COUNTS;

% Built-in or hardware component of total proportional
% gain. he motor input has in series power PWM amplifier
% of gain (PMW_GAIN) volts/volt and the shaft The power
% amplifier converts the low control voltage from the
% controller to high voltage required to drive the motor
% with a voltage gain of (PWM_GAIN)volts/volt given as
PWM_GAIN = 2*Vm/POSITION_ACTUATOR_SIGNAL_RANGE_VOLTS;

% The motor position output has in series potentiometer
% of gain factor  (POS_SENSOR_GAIN) volts/rads, that
% converts radians of position to volts. Here is how the
% conversion is determined: the motor output comes out as
% radians which is then to degrees by multiplier the
% (180/pi). Next the position sensor, (the potentiometer)
% has a total of (SENSOR_SUPPLY_VOLTS_RANGE) volts applied
% across it for a total of (SENSOR_POSITION_DEGREES_RANGE)
% degrees of its entire span, giving it volts/degree gain
% (SENSOR_SUPPLY_VOLTS_RANGE/SENSOR_POSITION_DEGREES_RANGE)
% which is a second multiplier finally translating motor
% output from radians to its volts analog.

% THIS SENSOR GAIN STEP IS VERY IMPORTANT STEP WHICH MUST
% ALWAYS BE DONE AND DONE CORRECTLY IN ORDER CONVERT THE
% OPEN LOOP TRANSFER FUNCTION INTO AN ELECTRICAL SYTEM
% DESIGN MODEL.
% If this step is missed or not done properly erroneous
% source of gain or scale factor may exists in the design
% model as a results of which design results will differ
% from simulation results.
POS_SENSOR_GAIN=(180/pi)*POSITION_SENSOR_VOLTS_PER_DEGREE;

% Read the response from file
load('\response20.txt');
yData = response20;
PGAIN  = 20;
clear response20;
% Convert measured response from digital counts to volts
y = yData*(POSITION_SENSOR_SIGNAL_RANGE_COUNTS/
      POSITION_SENSOR_SIGNAL_RANGE_DEGREES)*
             POSITION_ACTUATOR_VOLTS_PER_COUNTS;
% Software component of the proportional gain used in
% control loop software(in aid.c)

% Total number response samples taken
totalSamps = length(y);

% The vector of the exact sampling time points
t = Ts*[0:(totalSamps - 1)];

% Get the maximum sample or overshoot value
maxY = max( y);
% Get the minimum sample or overshoot value
minY = min( y);

% Determine output response delay
DELAY = 0;
for ( n = 1:totalSamps )
    % Scan for the first -ve to +ve zero crossing
    if ( y(n) == minY & y(n + 1) > minY)
        DELAY = n;
        break;
     end
end

% Set all sample during the delay to the minimum
for ( n = 1:DELAY )
    y(n) = minY;
end
% Shift sample reference to the minimum and scale
% the sample response for unity step input response
% which also means the DC gain is equal to unity
y = y - minY;
y = y/minY;
y = y/y(totalSamps);
yy = y;
finaY = y(totalSamps);
```

```
CLOSED_LOOP_DCGAIN = finaY;
pack;

% Get the maximum sample or overshoot value
maxY = max( y);
% Get the minimum sample or overshoot value
minY = min( y);

% Scan the samples for the overshoot(maximum sample)
% time
for ( k = 1:totalSamps )
   if ( y(k) == maxY)
      overShootTime = k;
      break;
   end
end

% Scan the response for the 2 time points where
% it crosses its final value level to begin
% overshooting and to begin undershooting
% respectively. The interval between these
% 2 time points is half the damped oscillation
% period
done = 0;
k = 1;
while ( done == 0 )
   if ( y(k) > finaY)
      overShootTime = k - 1;
      for ( m = k:totalSamps )
          if ( y(k) < finaY)
         underShootTime = m - 1;
         done = 1;
         break;
          end
          k = k + 1;
      end
   end
   k = k + 1;
end

% Compute the period of the damped oscillation seconds
Td = 2*abs( Ts*underShootTime - Ts*overShootTime );

% In absence of friction, we have theoretical LTI
% design system model with less damping so that the
% damped oscillation's period will be lower ( that is the
% frequency will  be higher) say by a factor of
% "FRICTION_DAMPING_PERIOD_CORRECTION"

% The true period of the theoretical LTI system in
% absence of friction is
Td = Td/FRICTION_DAMPING_FREQUECY_REDUCTION_COEFFICIENT;
Fd = inv(Td);                              % Hertz
Wd = 2*pi*Fd;                              % radians/sec
% NOTE: A decrease in phase angle as in discretization
% using ZOH will also produce less damping

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%COMPUTE THE PERCENT OF OVERSHOOT AND DAMPING FACTOR%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Mp = (maxY - finaY)/finaY;

% Again, in absence of friction, we have theoretical
% closed loop LTI design system model with less damping
% so that the overshoot percent will actually be higher
% by factor of say
%"FRICTION_PERCENT_OVERSHOOT_CORRECTION"

% So the true percent overshoot of the theoretical
% closed loop LTI system's step response in absence
% of friction is
Mp = Mp*FRICTION_OVERSHOOT_INCREASE_COEFFICIENT;

% So the true damping factor of the theoretical closed
% loop LTI system's step response in absence of friction
% is
%Ze = sqrt((log(Mp)/pi)^2/(1 + (log(Mp)/pi)^2));
Ze = abs(log(Mp)/abs(sqrt( (log(Mp)^2 + pi^2) )));

% So the un-damped frequency of the theoretical closed
% loop LTI system's step response in absence of friction
% is
Wn = Wd/abs(sqrt(1 - Ze^2));
Fn = Wn/2/pi;
if ( Fn > 0.1*Fa)
    disp('THE CLOSED LOOP BANDWIDTH TOO HIGH');
end


% So the time constant reciprocal of the theoretical
% closed loop LTI system is
Ap = Ze*Wn;

% So the dominant complex conjugate pole pair of the
% theoretical closed loop LTI system is
W1 = - Ap + Wd*i;
W2 = - Ap - Wd*i;

% The third pole is assumed to be the antialias
% filter pole if it is greater than the mechanical pole,
% Wm, but less than the electrical pole, We
Wo = 2*pi*Fa;

% We can now compute the theoretical LTI closed loop
% system transfer function, that is, one without the
% effect of friction disturbance, as
% (Gc(s) = cnumcx/cdencx) where
cnumcx = Wo*CLOSED_LOOP_DCGAIN*Wn^2;
cdencx = conv( [1, Wo],[ 1, 2*Ze*Wn, Wn^2 ]);
syscx = tf(cnumcx, cdencx);

% We can now discretize this closed loop transfer
% function as
sysdx = c2d(syscx, Ts);
[dnumcx, ddencx] = tfdata( sysdx, 'v');

% We can now compute the step response of the
% theoretical LTI closed loop system, without friction
% effect
[ydx, dx] = dlsim(dnumcx, ddencx, ones(1, totalSamps));
clear dx;
pack;

% We can now compute the parameters of the theorectical
% LTI open loop system transfer function from the
% correspoding closed loop system parameters as
WeTimesWm = 2*Ze*Wn*Wo + Wn*Wn;
WePlusWm = 2*Ze*Wn + Wo;

a = 1;
b = - (2*Ze*Wn + Wo);
c = 2*Ze*Wn*Wo + Wn*Wn;

We = (- b + sqrt(b^2 - 4*a*c))/(2*a);
Wm = WePlusWm - We;
Ao = Wo*Wn^2/PGAIN/PWM_GAIN/POS_SENSOR_GAIN;
Tm = abs(1/Wm);
Te = abs(1/We);
Fm = abs(Wm/2/pi);
Fe = abs(We/2/pi);

% Extract the DC motor transfer function from the
% computed open loop system transfer function
Mnump = Ao;
Mdenp = conv([1, 0], conv([1, Wm], [1, We]));

% Now run the closed loop simulink simulation for
% overall or the real system model that includes the
% computed motor transfer function, the friction model
% and other necessary non-linear components or effects
sim('sblkDigital_Py')

% Load the simulated step response from file back
% into the workspace
load('simOutFle.mat');
% u(1) = time
% u(2) = position command
% u(3) = position feedback
sim_time  = sim_p_Out(1, :);
sim_cmnd  = sim_p_Out(2, :);
sim_data = sim_p_Out(3, :);

% compute the unit step command used as input to
% simulate the step response
unitStepCommand =
(sim_cmnd - INITIAL_POSITION_COMMAND_DEGREES)./
(sim_cmnd(SimTime/Ts) - INITIAL_POSITION_COMMAND_DEGREES);

% compute the unit step response to the simualted
% unit step input
unitStepResponse =
(sim_data - sim_data(1)) ./(sim_cmnd(SimTime/Ts)
                                    - sim_data(1));

%y = (yData - yData(1)) ./
% (sim_cmnd(SimTime/Ts) - yData(1));
%figure(2);
```

```
%plot(sim_time, unitStepResponse,'b', sim_time,...
% yy, 'g',sim_time,yRho,'r');
%grid;title( 'Real Test in Green, Simulation in Blue');
%xlabel( 'time in seconds');
%ylabel(' position in volts');axis([0, 0.385, 0, 1.95]);

figure(2), plot( sim_time, unitStepResponse);
grid;title( 'Measured Step Response for...'
       'Proportional Gain Control of 4');
xlabel( 'time in sec'); ylabel('position v');
axis([0, 0.385, -0.10, 1.25]);
print( 2, '-djpeg', 'foo')
cnump = Mnump*PWM_GAIN*POS_SENSOR_GAIN;
cdenp = Mdenp;

Nx = length(cdenp) - 1;
% print( 2, '-djpeg', 'test_response')
```

A.2 MATLAB Script for Design to Validate Identified Motor and Friction Parameters.

The following MATLAB code runs the SIMULINK model that designs and simulates the lead compensating filter required in order to realize closed loop control dc motor position control by using the designed model based on the identified motor and friction parameters. The code starts with the desired closed loop control performance. Using that it calculates the required closed loop bandwidth and phase margin. Then it designs a lead filter based on the identified transfer function of the dc motor. It then implements the filter digitally in the output feedback control loop with the SIMULINK model of the identified dc motor and friction models. It captures the closed loop step response and checks to see if it meets the desired performance requirements. If the response does not meet the performance requirement, the desired closed loop bandwidth and the corresponding phase margin are recalculated for use to design a new lead filter to repeat the simulation. This design followed by simulation is iterated until the desired step response performance requirements are met. The lead compensating filter is then used in a real test to obtain real step response to compare with the simulated one to validate the identification scheme.

```
clear all;
clc;
format short e;

global Nc Fs Ts Ap1 Ap2 Bp1
global percentOverShoot
global percentSteadyStateError
global riseTime

% DESIGN REQUIREMENT
% Because the overall system ( simualtion model)
% or the real system is nonlinear the its performance
% can no be analytically computed as will have been for
% purely linear time invariant design system. For this
% reason analytical design of performance can only done on
% the linear time invariant sub-component or the
% so-called design model and then only verifying by
% simulation how well the real  or the simulation model
% meets the specified performance. In general the nonlinear
% real system or simulation model because external load,
% disturbances such as noise and friction all of which
% are not part of the linear time invariant is source of
% additional damping. Because of these external damping
% sources in the real system or simualtion model the linear
% time invariant design model must be designed to have less
% damping than is expected in real system or simulation
% model. Thus the performance designed for a linear time
% invariant design model to make its under-damped when
% verified in simulation on the real system or simulation
% model will show a response that is far more damped due to
% these external damping sources. In conclusion disturbance
% loading and nonlinearity in the real system, effectively,
% is a source of additional damping that robs the system of
% response speed or bandwidth. For this reason the damping
% performance specification to design damping in the response
% of the real or simulation system through analysis using the
% linear time invariant design model must include additional
% margin to offset the extra sources of damping in the real
% or simulation system model

% The overshoot performance specification of the step
% response being used as the basis for the control design is
% for a linear system. But the real system is non-linear due
% to main friction and control saturation etc. A very
% importance result of fthe friction it produce damping in the
% response so that more damping will show up in the real
% non-linear system than as specified by the percent
% over-shoot requirement. For this reason the the overshoot
% pperformance requirement should be set to far higher level
% since friction will cancel large portion of this overshoot.
% The overshoot performance requirement for the LTI system
% design model must be high (low damping) to be produce
% corresponding response in the real or simulation model
% system enough bandwidth to meet required rise time
% specification because a low overshoot also means low
% bandwidth high rise time for both the LTI design model
% and the real or simulation system model.
UNIT_STEP_OVERSHOOT = 15/100; %Increase overshoot to
                                %increase the decrease
```

```
% the rise time or to increase bandwidth

% 2. Step response with steady state error of no more than
% 0.8 percent (ie 0.008)
UNIT_STEP_STEADY_STATE_ERROR = 1;

% 3a.The open loop system's mechanical time constant, Tm, is
% the slower of the 2 finite time constants = (Rm*Jm/Kv/Kt),
% which includes the effect of the load inertia reflected to
% the motor shaft giving effective total inertia at the
% motor shaft of (Jm) is about 65millisecond so we should
% be able to have a closed loop system's step response with
% rise time no more than
UNIT_STEP_RISE_TIME = 0.100;


% 3b. or no more than 93 milliseconds for NORM PWM MODE

% LEAD/LAG DESIGN METHOD
% The UNITY GAIN cross-over frequency, Fx,(Wx) of an LTI
% OPEN LOOP system model is related to its corresponding
% its CLOSED LOOP bandwidth, Fb, (Wb) as Wx = 0.635*Wb.
% This allows us to select any frequency, Wb, on the LTI
% open loop transfer function G(s)( frequency response
% G(jw)) to be unity gain frequency simply by adding to
% the LTI open loop transfer fuction a gain compensation
% equal to the reciprocal of the gain magnitiude of the LTI
% open loop transfer function evaluated at this frequency.
% Then phase angle comepensation required at this frequency
% to meet the phase margin requirement can be added by
% phase lead or phase lag compensation. The closed loop
% system will be approximated to a 2nd order system
% characterized by dominant pair of complex conjugate
% poles at frequency, Fn, (Wn) or damped frequency, Fd,
% (Wd) at a damping factor, ZETA. The damped frequency
% of the LTI model closed loop system (Wd)rads/sec as
% Wd = Wn*abs(sqrt( 1 - ZETA^2)); Then the dominant closed
% loop pole (complex conjugate pair) required to achieve
% this bandwidth are computed as
% cPx(1) = - ZETA*Wn + Wd*i;
% cPx(2) = - ZETA*Wn - Wd*i;
% The 3rd pole is placed further so as to have minimum or
% no effect on the closed loop transient cPx(3) = 3.41*We;

% The specified percent overshoot, UNIT_STEP_OVERSHOOT,
% is related to this 2nd closed loop system's damping
% factor ZETA as
% ZETA = abs(log(UNIT_STEP_OVERSHOOT)/
%                   abs(sqrt( (log(UNIT_STEP_OVERSHOOT)^2
% + pi^2) )));
ZETA =
sqrt((log(UNIT_STEP_OVERSHOOT)/pi)^2/...
                   (1 +(log(UNIT_STEP_OVERSHOOT)/pi)^2));

% The required phase margin of the corresponding LTI
% model OPEN LOOP system is related to the correponding
% 2nd CLOSED LOOP LTI system's damping actor ZETA asis
% then computed as
PM=atan(2*ZETA/sqrt(sqrt(4*ZETA^4+1)- 2*ZETA^2))*180/pi;
```

```
% Because there exist experimental step response of the
% closed loop system over time interval of 384 millisecond
% we shall run simulation during each PID designs iteration
% over 384 millisecond for compare
%SimTime = 0.384;
%SimTime = 1;

% This MATLAB file read into the worskspace the identified
% motor and friction parameters
mfleModelIdentify;

% Set simulation time duration to 1 second
SimTime = 1;

% Compute the desired final range of time interval for
% steady state
steadyStateRange = 0.75*SimTime/Ts;

% Set simulation iterations counter to zero
n = 0;

% Start design iteration at desired closed loop bandwidth
% of 29 Hz for the closed loop system to speed up the
% simulated design (It is known,design will end at FBB = 49)
FBB = 29;

settledTimeReached = 0;

% Set starting iteration errors to 100 percent
percentOverShoot = 1;
percentSteadyStateError = 1;
riseTime      = SimTime;
while ( 1)

    %Wx = Wn*sqrt( sqrt(4*ZETA^4 + 1) - 2*ZETA^2);
    % Create the matlab representation of the open loop
    % transfer function
    sys = tf( cnump, cdenp);

    % Compute the desired unity gain crossover frequency
    % (Wx) rads/sec of
    % the open loop system
    WBB = 2*pi*FBB;
    Wx = 0.635*WBB;

    % Evaluate the gain Gx and phase angle Phx, of the open
    % loop system
    % at this frequency Wx
    [Gx, Phx] = bode( sys, Wx);
    %Hx = freqresp( sys, Wx);
    %Gx = abs( Hx);
    %Phx = angle( Hx);

    % We need a gain compensation Do and phase compensation
    % Php to be added to the open loop such that the gain Gx
    % at this frequency Wx is unity making this frequency
    % the unity gain cross-over frequency and also changing
    % the phase angle Phx at this frequency Wx to a value
    % Phn such that the new phase margin value PMn is exactly
    % PM, that required by the design specification which
    % will also meet the required closed loop bandwidth spec.
    % The gain compensation needed to reach unity gain
    % ,the proportional gain compensation portion of the
    % compensation filter, must be the reciprocal of the gain
    % Gx at Wx. Adding it to (multiplying it with ) the open
    % loop plant simply shifts/translates the open loop gain
    % curve along the magnitude axis until the Wx point
    % intersects the unity gain or 0 dB magnitude line
    Do = 1/Gx;

    % The phase margin, PMx, at PMx at Wx is: PMx =
    % 180 + Phx. Hence the additional phase margin, Php,
    % required to meet, PM, the required  phase margin is:
    % Php = PMx - PM = (180 + Phx) - PM, which is the
    % phase  required compensation:
    Php = (180 + Phx) - PM;

    % If the phase contribution required is negative
    if ( Php < 0 )

         % Then we need a phase LEAD compensator with phase
         % angle magnitude
         PHA = abs(Php) + 5;

         % A phase LEAD filter pole (P) which is the larger
         % of the 2 break freqecies is computed as
         P = Wx*sqrt((1+sin(PHA*pi/180))/(1-sin(PHA*pi/180)));

         % And the crossover frequency (Wx) is the geometric
         % mean of the lead zero and lead pole so the lead
         % zero is
         Z = Wx^2/P;

         % The completed lead compensator filter required
         % is including extra proportional gain, Go, added
         % to the open loop plant prior to the lead compensation
         numc = Do*P/Z*[1, Z];
         denc = [1, P];
         filterType = 2;
    elseif ( Php == 0)
         % Only a simple proportional compensator is required
         numc = Do;
         denc = 1;
         filterType = 0;
    elseif( Php > 0)

         % Thee we need a LAG compensator

         % The LAG filter zero (Z) must be placed at 1 decade
         % below the desired crossover frequency (Wx) and so
         % it computed as
         Z = 0.1*Wx;

         % The LAG filter pole (P) is smaller than the zero (Z)
         % hence Do > 1  the zero (Z) is a  factor of (Do)
         % larger than the pole (P)
         P = Z/Do;
```

```
        % The completed lag compensator filter required is
        % including  extra proportional gain, Go, added to
        % the open loop plant prior to the lead compensation
        numc = Do*P/Z*[1,  Z];
        denc = [1,  P];
        filterType = 1;
    end

    % Discretize the designed compensator with a ZOH at
    % sampling frequency (Fs)
    [dnumc, ddenc] = tfdata( c2d( tf(numc, denc), Ts), 'v');
    Ap1 = dnumc(1)/ddenc(1);
    Ap2 = dnumc(2)/ddenc(1);
    Bp1 = ddenc(2)/ddenc(1);
    Nc = length(numc) + length(denc) - 1;

    Ap1 = floor( 1024*Ap1);
    Ap2 = floor( 1024*Ap2);
    Bp1 = floor( 1024*Bp1);
    Ap3 = 0.0;
    Bp2 = 0.0;

    % Close the Run the simulation with the H-bridge power
    % amplifier in differential PWM control mode on the
    % nonlinear closed loop model
%   open('sPosServoEce725')
%   sim('sPosServoEce725')
%   open('sPosServoEce725y')
%   sim('sPosServoEce725y')
    open('sblkDigital_yComp.mdl')
    sim('sblkDigital_yComp.mdl')

    % Close the Run the simulation with the H-bridge power
    % amplifier in normal PWM control mode on the nonlinear
    % closed loop model sim('sblkdigital_compenNormPwm')

    % Load the simulated closed loop step response from file
    load('simOutFle.mat');
    % u(1) = time
    % u(2) = position command
    % u(3) = position feedback
    sim_time  = sim_compenCtrl_Out(1, :);
    sim_cmnd  = sim_compenCtrl_Out(2, :);
    sim_data = sim_compenCtrl_Out(3, :);
    len = length(sim_time) - 1;
    sim_time  = sim_time(1:len);
    sim_cmnd  = sim_cmnd(1:len);
    sim_data  = sim_data(1:len);

    % compute the unit step command used as input to simulate
    % the stepresponse
    unitStepCommand =
      (sim_cmnd -INITIAL_POSITION_COMMAND_DEGREES) ./...
                                    (sim_cmnd(SimTime/Ts)...
                           - INITIAL_POSITION_COMMAND_DEGREES);

    % compute the unit step response to the simualted unit
    % step input
    unitStepResponseSim =...
        (sim_data - sim_data(1)) ./...
                          (sim_cmnd(SimTime/Ts) - sim_data(1));

    % Determine the unit step response overshoot
    ymax = max( unitStepResponseSim);

    % Overshoot must exceed the unit command else it is
    % NOT overshoot
    if ( ymax > 1)

        % Yes this an overshoot
        percentOverShoot = ymax - 1;
    else
        % No there is NO or zero overshoot
        percentOverShoot = 0;
    end

    % Search the response for steady state error
    % specification convergence and for settling time
    k = 1;
    m = 1;
    settledTimeReached = 0;
    while ( m <= SimTime/Ts)
        % compute the next error
        percentSteadyStateError =...
                                abs(1 - unitStepResponseSim(m));

        % If the current sample is within the range used
        % to determine steady state region
        if (  m > (SimTime/Ts - steadyStateRange) )
            % Then if the current error meets the steady
            % state error spec
            if ( percentSteadyStateError <=...
                                UNIT_STEP_STEADY_STATE_ERROR)
                % Increment the counter
                k = k + 1;
            end
        end
        if ( settledTimeReached == 0)
           % Then if current error meets the settling time
           % error value
           if ( unitStepResponseSim(m) >= 0.9)
               % Then capture the settling time
              riseTime = m*Ts;
              settledTimeReached = 1;
           end
        end
        m = m + 1;
    end
disp(sprintf( 'Ap1 = %0.3f, Ap2 = %0.3f, Bp1 = %0.3f',...
                          Ap1, Ap2, Bp1));
disp(sprintf( 'FBB=%0.3fHz final deadband time=%d mSec\n',...
                                 FBB,k));

    if ( 100*percentOverShoot > 100*UNIT_STEP_OVERSHOOT )
           OVdone = 0;
    else
        OVdone = 1;
    end
    if ( 1000*percentSteadyStateError > ...
                          1000*UNIT_STEP_STEADY_STATE_ERROR)

           SSdone = 0;
    else
        SSdone = 1;
    end
    if ( 1000*riseTime < 1000*UNIT_STEP_RISE_TIME)
            RTdone = 1;
    else

         RTdone = 0;
     end

     if ((OVdone == 1) &&  (SSdone == 1) &&( RTdone == 1))
        f1 = fopen('leadLagCoeffs.txt', 'w');
        if ( f1 == -1)
        error(' Unable to save filter coefficients to file');
        else
        fprintf(f1,'%0.4e\t %0.4e\t %0.4e\t %0.4e\t %0.4e\t',
   Ap1/1024.0, Ap2/1024.0,Ap3/1024.0, Bp1/1024.0, Bp2/1024.0);
           fclose(f1);
        end
        break;
     else
          FBB = FBB + 1;
    end
    % UNIT_STEP_RISE_TIME = UNIT_STEP_RISE_TIME - 0.001;

    %requiredMaxRiseTime = UNIT_STEP_RISE_TIME
    %riseTime
    %requiredMaxOverShot = UNIT_STEP_OVERSHOOT
    %percentOverShoot

    %requiredMaxSteadyState = UNIT_STEP_STEADY_STATE_ERROR
    %percentSteadyStateError

    %FBB
    %k = k + 1
    %Ap1
    %Ap2
    %Bp1
    n = n + 1;

end

figure(4);
plot( sim_time, unitStepResponseSim);
grid;title( 'step response...lead/lag compensation control');
xlabel( 'time in seconds');
ylabel(' position in volts');axis([0, SimTime, 0, 1.1]);

% beep( 'on');
% beep;
% pause(1);
% beep;
% pause(1);
% beep;
% pause(1);
% disp(sprintf('\n Simulation comple...hit
%  any key to run real test... and stay...
%                     clear of the motor!!!'));
% pause();
% beep( 'off');
%
% [s, w] = system('testRun')
%

% if instead the real test succeeds the open the
% disk file for the response
    load( 'leadLagResponse.txt');

    % compute the unit step response to the simualted
    % unit step input
    unitStepResponseTest = ...
    (leadLagResponse - leadLagResponse(1)) ./...
                     (sim_cmnd(SimTime/Ts)...
                                   - leadLagResponse(1));

    figure(5);
    plot( sim_time, unitStepCommand, 'b',
       sim_time, unitStepResponseSim, 'g', sim_time,
                                unitStepResponseTest, 'r');
    grid;
title( 'command = blue, simulation = green, real test = red');
xlabel( 'time in seconds'); ylabel(' position in volts');
axis([0, SimTime, 0, 1.1]);
figure(6); plot( sim_time, sim_cmnd, 'b',...
sim_time, sim_data, 'g', sim_time, leadLagResponse,'r');
grid;
title( 'command = blue, simulation = green, real test = red');
xlabel( 'time in seconds'); ylabel(' position in degrees');
axis([0, SimTime, 800, 1950]);
```

APPENDIX B

MATLAB CONTROL SYSTEM DESIGN FILE FOR THE SIMPLE INVERTED PENDULUM

B.1 MATLAB Script for Design and Simulation of Robot Arm

The following MATLAB code designs and simulates the robot arm control system.

```
   clear all
   clc
   format short e
   global Jm Bp Bm m g L Rm Lm Kt Kv
   global xo simTime
   global Ts
   global sensorRadiansRange sensorDegreesRange
   global sensorVoltsRange sensorCountsRange
   global actuatorVoltsRange actuatorCountsRange
   global pwmGain sensorGain cnump cdenp numc denc
   global sysOrder closedLoopPoles Ap1 Ap2 Ap3 Bp1 Bp2
   global countsPerVolts

   % PSC DESIGN TO CONTROL A ROBOT ARM
% SENSOR AND ANGULAR ORIENTATION ALIGNMENT OF THE ARM
   % A 5-turn rotary potentiometer is attached to the DC motor
   % and the simple inverted pendulum to create a robot. To
   % measure the angular position of the arm the pot is
   % attached such that its CCW rotation is positive
   % (increasing angle and voltage) hence CW rotation is
   % negative ( decreasing angle and voltage ). With the 10
   % equlibrium points are distributed along the 1800 degree
   % (10*pi) radian counts as follows:
   % deg :[0, 180,.. 1800]
   % rads:[0,  pi,.. 10*pi]
   % volts  :[0, 0.5,...5.0]
   % counts :[0, 102,.. 1023]
   % The stable and unstable equilibrium distribution is:
   % STABLE:
   % degrees :[0, 360,  720,  1080, 1440, 1800]
   % radians :[0, 2*pi, 4*pi, 6*pi, 8*pi, 10*pi]
   % volts   :[0, 1.0   2.0,  3.0,  4.0,  5.0]
   % counts  :[0, 205,  410,  617,  821,  1023]
   % UNSTABLE
   % degrees :[180, 540,  900,  1260, 1620]
   % radians :[pi,  3*pi, 5*pi, 7*pi, 9*pi]
   % volts   :[0.5, 1.5,  2.5,  3.5,  4.5]
   % counts  :[102, 307,  512,  719,  923]
% UNIPOLAR SENSOR RANGE AND ACTUATOR RANGE CONSTRAINTS
   % Because of the digital control using microcontroller
   % with unipolar power supply, the ADC conversion is
   % unsigned bits and the output PWM to drive the h-bridge
   % at low is 0 volt and at high is  Vdd volts
   % (microcontroller supply voltage). USING THE UNIPOLAR
   % SENSOR FOR OUTPUT FEEDBACK The usual feedback sensor
   % solution is to avoid drive system to reference commands
   % near the two ends of unipolar sensor range ( 0 volt and
   % Vdd) by creating lower and upper limits or margins
   % Vmargin > 0 volt at both ends. This means the
   % reference command cannot be less than lower margin
   % Vmargin or greater than upper (Vdd - Vmargin). This
   % Vmargin must be chosen equal to the maximum overshoot
   % expected in the output response. The reason for this
   % that, at both ends of the sensor range are non-linear
   % hard limits that can cause system damage.
% USING THE UNIPOLAR PWM SIGNAL FEEDBACK CONTROL
   % The usual solution for unipolar control signal, in this
   % case, unipolar PWM signal control signal is to assign
   % half of the unipolar control signal ( the PWM) range
   % to negative range of the bipolar software control count
   % and remaining half to the positve range of the bipolar
   % software control count by linearly maping the unipolar
   % range of the PWM signal [0<--->Vdd]volts or
   % [0<--->Max_ADC]counts to bipolar software control
   % count [0<--->Max_PWM_Counts]counts as follows:
   % pwmCount = 0.5*Max_PWM_Counts - 0.5*bipolarCtrlCount
   % Thus the biploar software feedback error range is:
   % [-Max_ADC<---> +Max_ADC] The bipolar software feedback
   % control signal is  : [-Max_PWM_Counts<--->+Max_PWM_Counts]
   % The unipolar software feedback PWM signal is      :
   % [0<--->Max_PWM_Counts] with transformation equation:
   % pwmCount = 0.5*Max_PWM_Counts - 0.5*bipolarCtrlCount
   % Note: To reverse control or error polarity reference
   % to the unipolar actuation signalswap the sign as:
   % -0.5*Max_PWM_Counts + 0.5*bipolarCtrlCount

% INITIAL POINT CONSTRAINT
   % Without initial external control, the arm can only
   % start from one of its stable equilibrium points. Thus
   % we must choose one of the stable equilibrium points to
   % start the system. We wiil choose the one nearest to the
   % middle range of the sensor to satisfy avoiding hard
   % limit stops at the end of the sensor ranges in case we
   % drive system with large step. Near the middle are two
   % stable equilibrium points one at 4*pi and the other at
   % 6*pi. We arbitrarily select the 4*pi radian or 720
   % degrees equilibrium point so that in our control testing
   % we will drive in (CCW or increasing angle) direction
   % the pendulum from this stable equilibrium point to the
   % unstable one, 5*pi or 900 degrees, in the middle.
% DERIVING THE DYNAMICS (DIFFERENTIAL EQUATION OF ROBOT ARM)
   % This a 3rd order nonlinear DE. The DC motor is the
   % actuator that provides the torque required drive the
   % pendulum positions. Thus the motor developed torque
   % must be used to overcome the following:
   % 1) Static friction in the motor, Fm, and at the
   % pendulum hinge, Fp
   % 2) Viscous friction, Bm, in the motor and, Bp,
   % around the pendulum which very large compared that of
   % the motor
   % 3)The torque due gravity on the pendulum bob, m,
   % which is m*g*L*sin(x1) where x1 is the angle the
   % pendulum makes with the vertical
   % 4) Inertia of both pendulum m*L^2 and the motor, Jm.
   % The differential equation is (Jm + m*L^2)*d^2x2/dt^2 =
   % Kt*x3-(Bm + Bp)*x2- m*g*L*sin(x1)- Fm - Fp where
   % x2 = velocity of the pendulum and x3 = motor current,
   % Kt = motor torque constant and the dynmics of the motor
   % current is dx3/dt = - Kv/Lm*x2 - Rm/Lm*x3 + Vm/Lm;
   % where Kv is the back emf constant of the motor,
   % Rm = motor resistant and Lm = motor inductance and Vm
   % motor supply voltage. the resulting VMDE is
   % x1dot = x(2);
   % x2dot = - m*g*L/(Jm + m*L^2)*sin(x(1)) - (Bm + Bp)/...
   % (Jm + m*L^2)*x(2)+ Kt/(Jm + m*L^2)*x(3) - Fm - Fp;
   % x3dot = - Kv/Lm*x(2) - Rm/Lm*x(3) + Vm/Lm;
% REALISTIC SIMULATION SIMULINK MODEL
   % As can be seen the VMDE is nonlinear with both types
   % nonlinearity
   % 1) Hard nonlinearties
   %               a) supply voltage Vm and current
   %               Clim saturations
   %               b) Static frictions Fp and Fm
   %               c) backlash at the coupling, Bkd
   % 2) Smooth nonlinearity due state dependent term
   % sin(x1)
   % The nonlinear design method being proposed here
   % will take care of the the smooth nonlinearity. The
   % hard nonlinearities from static friction and saturations
   % will be ignored with the hope that their effect is
   % minnimum on the control. So for a very realistic
   % simulation the SIMULINK MODEL must include the hard and
   % Inspect the SIMULINK MODEL and verify that all these
   % elements are present
% SMOOTH NONLINEAR DESIGN MODEL
% Since the hard nonlinearities of static friction and
% saturations are not described with smooth functions
% they are not involved in the design model but if the
% system is designed well enough it can work well in
% presence of the hard nonlinearities.
% x1dot = x(2);
% x2dot = - m*g*L/(Jm + m*L^2)*sin(x(1)) -
%       (Bm + Bp)/(Jm + m*L^2)*x(2)+ Kt/(Jm + m*L^2)*x(3);
% x3dot = - Kv/Lm*x(2) - Rm/Lm*x(3) + Vm/Lm;

% MEASURING THE NATURAL FREQUENCY OF THE SYSTEM
% The motorized pendulum without input control volta
% position, due to friction and viscous drag will NOT
% oscillate forever when displaced enough away from the
% stable equilibrium rest position, instead will do
% a number cycles of oscillation and finally comes to
% rest at that stable equilibrium. The total number of
% cycles it oscillates through is determined by the
% amount friction and viscous drag.
% The frequency Wo rads/sec or Fo Hz of the oscillation is square root of
% the coefficient of the sin(x1) termm
% Wo = sqrt(m*g*L/(Jm + m*L^2))
% To = 1/To
% Fo = Wo/2/pi;
% This period To is measured by displacing the pendulum to 90 degrees
% position and letting go without any control input voltage while capturing
% the position sensor data ever 1 millisecond inside the microcontroller
To = 1.4; % sec
Fo = 1/To;
Wo = 2*pi*Fo;

% Sensor and digital controller parameters
sensorTurns = 5;
sensorRadiansRange = sensorTurns*2*pi;
sensorDegreesRange = sensorTurns*360;
sensorVoltsRange   = 5;
sensorCountsRange  = 1024;
actuatorCountsRange = 768;
actuatorVoltsRange = 24;
% Counts to volts scale
POWER_SCALER = 1000;
% Sampling Frequency and period
Fs = 1000;
Ts = inv(Fs);
Ws = 2*pi*Fs;

% Analog anti-alias filter bandwidth in (Hz)
Fa = 111;

% PWM frequency and period
Fpwm = 15625;             % experimental value
Tpwm = inv(Fpwm);
Wpwm = 2*pi*Fpwm;


% Built-in or hardware component of total proportional
% gain. The motor input has in series power PWM amplifier
% of gain (PMW_GAIN) volts/volt and the shaft. The power
% amplifier converts the low control voltage from the
% controller to high voltage required to drive the motor
% with a voltage gain of (PWM_GAIN)volts/volt given as
pwmGain = 2*actuatorVoltsRange/sensorVoltsRange;

% The motor position output has in series potentiometer
% of gain factor (POS_SENSOR_GAIN) volts/rads, that converts
% radians of position to volts. Here is how the conversion is
% determined: the motor output comes out as radians which is
% then to degrees by multiplier the (180/pi). Next the
% position sensor, (the potentiometer) has a total of
% (SENSOR_SUPPLY_VOLTS_RANGE) volts applied across it for
% a total of (SENSOR_POSITION_DEGREES_RANGE) degrees of
% its entire span, giving it volts/degree gain of
% (SENSOR_SUPPLY_VOLTS_RANGE/SENSOR_POSITION_DEGREES_RANGE)
% which is a second multiplier finally translating motor
% output from radians to its volts analog. THIS SENSOR GAIN
% STEP IS VERY IMPORTANT STEP WHICH MUST ALWAYS BE DONE
% AND DONE CORRECTLY IN ORDER CONVERT THE OPEN LOOP
% TRANSFER FUNCTION INTO AN ELECTRICAL SYSTEM DESIGN MODEL.
% If this step is not done correctly error in scale factor
% may exists in the design model as a results of which
% expected response will differ from simulation results.
sensorGain = ...
  POWER_SCALER*(180/pi)*sensorVoltsRange/sensorDegreesRange;
```

```
% Motor shaft-to-sensor backlash (amount of play) in degrees
Bkd = 3;

% The motor parameters
% Motor voltage (Vm) volts
Vm = actuatorVoltsRange;

% Motor armature resistance (Rm) Ohms
Rm = 2.96;

% Stall current (limit)
Im = Vm/Rm;

% Motor armature inductance Henries
Lm = 2.51e-003;

% Torque constant (Kt)oz.in/A
Kt = 5.17;

% Back Emf constant (Kv)v/(rad/sec)
Kv = 3.82*60/1000/2/pi;

% Motor rotor moment of inertia (Jm)oz.in.sec^2 260% increase
% contribution from shaft couplers and pulley
Jm = 5.9e-004;

% Motor viscous damping constant(Bm )oz.in/rad/sec
Bm  = (6.7*60/1000/2/pi);

% Motor static friction oz-in
Fm = 0.6;

% The pendulum parameters
L = 2.5*12;   % 2.5*12length in inches of pendulum
g = 32.2*12;  % acceleration due to gravity inches/sec^2
m =  2;       % mass of the pendulum arm in oz

% Pendulum viscous friction set to that of the motor
Bp = Bm;

% Pendulum static friction in terms of that of the motor
Fp = 2.1*Fm;

% CALCULATING INITIAL STATE AND CONTROL
% We start the system from stable equlibrium at the origin
% which is any one of the following positions
%x1 = [0,  2*pi,  4*pi,  6*pi,  8*pi,  10*pi];
% We select the one at 2*pi
x1 = 2*pi;
x2 = 0.00000000001;
x3 = 0.00000000001;

% Read the initial state vector
xo = [ x1, x2, x3];

% Select CCFD SDC state space representation of the system
% and evaluate them from the initial state vector value
a11 = 0;
```

```
a12 = 1;
a13 = 0;
b1  = 0;
c1  = 1;

pp = Kt/(Jm+m*L^2);
a21 = 0;
a22 = 0;
a23 = (-m*g*L/(Jm+m*L^2)*sin(x1)-(Bp+Bm)/(Jm+m*L^2)*x2)/x3 + pp;
b2  = 0;
c2  = 0;

a31 = 0;
a32 = -Kv/Lm;
a33 = -Rm/Lm;
b3  = pwmGain*sensorGain/Lm;
c3  = 0;
% Evaluate the state matrix control and output vectors
A = [ a11,     a12,      a13;
      a21,     a22,      a23;
      a31,     a32,      a33 ];
B = [ b1;      b2;       b3  ];
C = [ c1,      c2,       c3  ];
D =   0;

% Compute the system order and its initial eigenvalues
sysOrder = length( eig(A));
P1 = eig(A);

% PERFORMANCE REQUIREMENTS
   % The most important or most critical requirement that
   % will determine. The lower the phase margin the faster
   % the response ( the higher the closed loop bandwidth)
   % and the higher the overshoot in step response and the
   % lower the steady state error. Although the lower steady
   % state error and higher bandwidth are desired consequences
   % of the lower phase margin, the low relative stability
   % and damped oscilation are not hence a compromise is
   % sought, by a phase margin requirement that provides
   % high enough bandwidth and low steady state error at
   % moderate relative stability. The phase margin requirement
   % is directly related o the requirement on percent overshoot
   % in step response as
   Mp = 1;

   % The damping factor corresponding to this step response
   % overshoot is ZETA = abs(log(Mp/100)/...
   %             abs(sqrt( (log(Mp/100)^2 + pi^2) )));
   ZETA = sqrt((log(Mp/100)/pi)^2/(1 + (log(Mp/100)/pi)^2));


   % The closed loop bandwidth in Hz
   Fb = 500

   Wb  = 2*Fb*pi;

   % The crossover frequency (Wx) rads/sec of the open loop
   % system required to achieve this desired closed loop
   % bandwidth is
```

```
    Wx = 0.635*Wb;

    simTime = 5000*Ts;

    % The simulation using the SIMULINK block of input
    % reference and the reference switch is set up such that
    % upon initialization the system stabilizes to some initial
    % state and remains there up to 4 seconds time point at
    % which time a step input excitation of size yref is applied
    % until the 6 second time point, leaving the system to return
    % to its initial stabilized state Drive the system to the
    % unstable middle point at 5pi or 900 degree point
    yref = 3*pi;

    simCounter = 0;

    % First simualtion iteration of the desired closed
    % loop bandwidth
    FBB = Wx/(2*pi*0.635);

    found = 0;
    % Enter the simulation iteration loop
    while ( found == 0)

          % Convert the closed loop bandwidth specification
          % from Hz to  rads/sec Wb = 2*pi*FBB; Compute the
          % desired unity gain cross-over frequency, Wx, from
          % the desired closed loop bandwidth, Wb Hz.
          Wx = 0.635*2*pi*FBB;

          % Compute an equivalent closed loop 2nd system's
          % natural frequency
          Wn = Wx/sqrt((sqrt( 4*ZETA^4 + 1) - 2*ZETA^2));
% Wn = Wx/sqrt(1 + 2*ZETA^2 + 2*sqrt( ZETA^4 + ZETA^2 + 1/2))
% Wn = Wb/sqrt( 1 - 2*ZETA^2 + sqrt( 2 - 4*ZETA^2 + 4*ZETA^4));

          % Compute an equivalent closed loop 2nd system's
          % damped frequency this is the frequency at which a
          % step response of the equivalent closed loop
          % 2nd-order system will oscillate
          Wd = Wn*abs(sqrt( 1 - ZETA^2));

          % We are now going to compute the desired closed
          % loop poles locations required to achieve this closed
          % loop bandwith and  percent of overshoot. First
          % allocate memory for the total number closed loop
          % poles to place
          closedLoopPoles = zeros(1, 2*sysOrder - 1);

          % Three of the closed loop poles will be made
          % the dominant, one complex  conjugate pair and
          % one real. The complex conjugate pair will be
          % placed at frequency equal the desired closed loop
          % bandwidth ( the desired damped natural frequency),
          % Wd, and the real pole will be placed at the real
          % part frequency of the
          % complex conjugate pair.
%           closedLoopPoles(1)  = - ZETA*Wn;
%           closedLoopPoles(2)  = -ZETA*Wn + Wd*i;
%           closedLoopPoles(3) = -ZETA*Wn - Wd*i;
%           % The rest of the closed loop poles will be placed
            % as real poles
%           % far from the 3 dominant ones at frequencies
            % eqaul to large integer multiples of the real
            % part frequency of the complex conjugate pair
            % so that they no or very minimum effect of the
%           % closed loop system. At say 10x, 15x etc
%           closedLoopPoles(4) = -10*(ZETA*Wn);
%           closedLoopPoles(5) = -15*(ZETA*Wn);

          closedLoopPoles(1) = -ZETA*Wn + Wd*i;
          closedLoopPoles(2) = -ZETA*Wn - Wd*i;
          closedLoopPoles(3) = -5*ZETA*Wn + 5*Wd*i;
          closedLoopPoles(4) = -5*ZETA*Wn - 5*Wd*i;
          closedLoopPoles(5) = -5*(ZETA*Wn);
          % The rest of the closed loop poles will be placed
          % as real poles far from the 3 dominant ones at
          % frequencies eqaul to large integer multiples of
          % the real part frequency of the complex conjugate
          % pair so that they no or very minimum effect of the
          % closed loop system. At say 10x, 15x etc


% IMPORTANCE OF SIMULATION FOR DESIGN BASED ON LTI DESIGN
%MODEL
          % If it were not for the hard nonlinearities, like
          % control actuator saturation (current limits etc)
          % and stiction, backlash and sensor deadzone etc
          % and noise that are always present in any real system,
          % control design that meets desired performance based
          % on the ideal LTI design model will produce the same
          % desired. Because of these hard nonlinearities, we
          % need to control designed based on the ideal LTI
          % design model in simulation until the simulation
          % performance, in presence of these hard
          % nonlinearities meet the desired performance.
% IMPORTANCE OF SIMULATION FOR DESIGN BASED ON THE NONLINEAR
% SDTF DESIGN MODEL
          % There is still the same hard nonlinearities
          % present in general smooth nonnlinear system modeled
          % using the SDTF model. The SDTF-based controller (PSC)
          % performance will a bit worse than fixed gain one
          % based on LTI design model at closed loop bandwidths
          % lower than the dynamic bandwidth of the evolving
          % nonlinear system. So as the PSC closed loop
          % bandwidth increases the response (transient) gets
          % better. But also higher bandwidths will lead to high
          % gain and satuaration and noise filtering that
          % amplifies the hard nonlinearities enough for
          % them to destabilize the system.
          % Run the simulation of the design control
          open('simRobotArmPscControl.mdl');
          sim('simRobotArmPscControl.mdl');

          % Load the simulated closed loop step response from
          % file
          load('simOutFle.mat');
          % u(1) = time
```

```
        % u(2) = position command
        % u(3) = position feedback
        sim_time  = simOutputVector(1, :);
        sim_data = simOutputVector(2, :);
        sim_cmnd  = simOutputVector(3, :);

   simCounter  = simCounter + 1

        kkk = (length(sim_data) - 1000);
        mm = 0;
        while ( kkk < length(sim_data))
            if ( abs(sim_data(kkk) - sim_cmnd(kkk)) < 5)
                mm = mm + 1;
            end
            kkk = kkk + 1;
        end
        % Compute the closed loop SDTF
 sysA = feedback(tf(conv(cnump, numc), conv(cdenp, denc)), 1);
        [nump, denp] = tfdata( sysA, 'v');
        theRoots = roots( denp)
        FBB

        floor(Ap1)
        floor(Ap2)
        floor(Ap3)
        floor(Bp1)
        floor(Bp2)

        if ( mm  >= 999)
             %disp(sprintf( 'Note: For the designed closed
             %loop bandwidth of %d Hz, the anti alias filter
             %bandwidth required is %d Hz and the minimum
   %ampling frequency required is = %d Hz\n', FBB, Fa, Fs ));
             found = 1;
             Fs
             Fa
             FBB
 sysA = feedback( tf(conv(cnump, numc), conv(cdenp, denc)), 1);
             [nump, denp] = tfdata( sysA, 'v');
             theRoots = roots( denp)
        else
            FBB = FBB + 10;
        end
  end

figure(1), plot( sim_time, sim_cmnd, sim_time, sim_data);
grid;
title( 'PSC Control of Robot Arm');
xlabel( 'time in seconds');
ylabel(' Pendulum Angular Position in degrees');
axis([0, 3, 350, 550]);
print( 1, '-djpeg', 'response_plot2')
```

B.2 MATLAB Script for PSC Filter Design By Pole Placement

The following MATLAB function is used to design the compensating filter during the PSC design and implementation online.

```
function[compNum,compDen]= ...
mFnxCompFilterXferFnx( plantNum,plantDen,cePoly)

% This function computes transfer function, D(s) =
% compNum(s)/compDen(s), of a compensator required to place
% the closed loop poles to produce characterstic polynomial
% provided by the caller "cePoly(s)" for an open loop system
% of transfer function numerator and denominator polynomials
% "plantNum(s)" and "plantDen(s)" also provided by the caller
% Check to make sure the transfer function of the plant is
% proper rational
if ( length( plantNum) == length( plantDen))
    % If the first coefficient element (the highest "s" power
    % coefficeint)of the numerator polynomial is zero
    if ( plantNum( 1) == 0)
        % Then remove this first zero coefficient element and
        % reduce the length of the numerator polynomial by
        % 1 this makes the transfer function a prpoer rational
        plantNum = plantNum( 2:length( plantNum) );
    else
error('Degree of numerator must 1 less than its denominator');
    end

end

% Get the system order
sysOrder = length( plantDen) - 1;
% Determine the size of the open loop plant numerator and
% denominator vectors
denLen = length( plantDen);

% Allocate memory to hold the M-matrix which is made up of
% matrices A & B The B-matrix size is:(row len=2*sysOrder-1)
%                       (col len = sysOrder)
B = zeros( (2*sysOrder - 1), sysOrder);
%The A-matrix size is :(row len = 2*sysOrder - 1)
%                       (col len = sysOrder - 1 = numLen)
A = zeros( (2*sysOrder - 1), sysOrder - 1);
%The V-vector size is :(row len = 2*sysOrder - 1)
V = zeros( (2*sysOrder - 1), 1);

% Create the V-vector
for( rows = 1:(2*sysOrder - 1))
    if ( rows <= sysOrder)
        V(rows) = cePoly( rows + 1) - plantDen( rows + 1);
    else
        V(rows) = cePoly( rows + 1);
    end
end
```

```
% Create matrix B (row size = 2*sysOrder - 1)
%                 (col size = sysOrder)
k = 0;
for ( cols = 1:sysOrder)
    for ( rows = 1:denLen - 1)
       B( rows + k, cols) = plantNum( rows);
    end
    k = k + 1;
end
% Create matrix A (row size = 2*sysOrder - 1)
%                 (col size = sysOrder - 1)
k = 0;
for ( cols = 1:sysOrder - 1)
    for ( rows = 1:denLen)
       A( rows + k, cols) = plantDen( rows );
    end
     k = k + 1;
end

% Put matrix A and B together to form matrix M
M = [B,A];

% Invert Matrix M and multiply it with vector V, the
% closed loop poles vector to obtain vector U
U = inv(M)*V;

% The first "sysOrder" rows of vector U are the
% coefficients of the compensator numerator
% polynomial and the rest are the denominator
% polynomial coeeefficient excluding the first
% coefficient which is unity.

compNum = [U(1:sysOrder)]';
compDen = [1; U((sysOrder + 1):(2*sysOrder - 1))]';
```

BIBLIOGRAPHY

[1] B. Armstrong-Helouvry, B. Dupont, and C. Canudas de Wit. A survey of models, analysis tools and compensattion methods for the control of machines with friction. In *Automatica*, volume 30, pages 1083–1138, 1994.

[2] X. Bao, F. Zhuo, Y. Tian, and P. Tan. A new model for control of systems with friction. In *IEEE Trans on Automatic Control*, volume 40, pages 419–425, 1995.

[3] X. Bao, F. Zhuo, Y. Tian, and P. Tan. Simplified feedback linearization of three–phase photovoltaic inverter with lcl filter. In *IEEE Trans. on Power Electronics*, volume 28, pages 2741–2752, 2013.

[4] H. Blomberg and Y. Linen. *Algebraic Theory for Multivariable Linear Systems.* London, England: Academic Press, 1983.

[5] L. Chih-Jer, Y. Her-Teng, and T. Yun-Cheng. Identification of nonlinear friction characteristics and precision control for a linear motor stage. In *IEEE/ASME Trans on Mechatronics*, volume 18, pages 1385–1396, 2013.

[6] T. Cimen. State–dependent riccati equation (SDRE) control: A survey. In *Proceedings of The International Federation of Automatic Control*, Seoul, South Korea, 2008.

[7] J. Cloutier. State-dependent riccati equation techniques: An overview. In *Proceedings of American Control Conference*, Albuquerque, New Mexico, 1997.

[8] J. Cloutier, C. D‘Souza, and C. Mracek. Nonlinear regulation and nonlinear h–infinity control via the state–dependent ricatti equation technique. In *Proceedings of International Conference*, Daytona Beach, FL, 1996.

[9] P. Dahl. A solid friction model. In *Aerospace Corp Tech, Rep. TOR*, pages 3107–3118, Daytona Beach, FL, 1968.

[10] M. Di Benedetto. Nonlinear strong model matching. In *IEEE Trans on Automatic Control*, volume 35, pages 1351–1355, 1990.

[11] D. Ehrler and S. Vadali. Examination of the optimal nonlinear regulator problem. In *Proceedings of the AIAA Guidance, Navigation and Control Conference*, Minneapolis, MN, 1988.

[12] B. Friedland. *Advanced Control System Design.* Englewood Cliffs, NJ, USA: Prentice Hall, 1996.

[13] D. Haessig and B. Friedland. On the modeling and simulation of friction. In *Dynamic Systems, Measurement and Control*, volume 113, pages 354–362, 1991.

[14] M. Halas and M. Huba. Symbolic computation for nonlinear systems using quotients over skew polynomial ring. In *14th Mediterranean Conference on Control and Automation*, Ancona, Italy, 2006.

[15] M. Halas and U Kotta. Extension of the concept of transfer function to discrete–time nonlinear control systems. In *European Control Conference*, volume 7, pages 2–5, Kos, Greece, 2007.

[16] K. Hammet, C. Hall, and D. Ridgely. Controllability issues in nonlinear state–dependent riccati equation control. In *Guidance Control and Dynamics*, volume 21, 1998.

[17] C. Hsieh and Y-C. Pan. Dynamic behavior and modeling of the presliding static friction. In *Wear*, volume 242, pages 1–17, 2000.

[18] L. Jacobsen and R. Ayre. *Engineering Vibrations*. New York, NY, USA: McGraw-Hill, 1958.

[19] A. Jeffrey, X. Xia, and Craig I. When to initiate HIV therapy: A control theoretic approach. In *IEEE Trans on Biomedical Engineering*, volume 50, 2003.

[20] H. Khalil. *Nonlinear Systems*. Englewood Cliffs, NJ, USA: Prentice Hall, 2001.

[21] M. Kristic, I. Kanellakopoulos, and P. Kokotovic. *Nonlinear and Adaptive Control Design*. New York, NY, USA: John Wiley and Sons, 1995.

[22] E. Liceaga-Castro, R. Bradley, and L. Castro-Linares. Helicopter control design using feedback linearization control techniques. In *Proceedings of 28th IEEE Decision and Control Conference*, volume 1, pages 533–534, Tampa FL, 1989.

[23] A. Luviano-Juarez, J. Cortes-Romero, and H. Sira-Ramirez. Algebraic identification and control of an uncertain dc motor using the delta operator approach. In *7th International Conference on Electrical Engineering computing Science and Automatic Control*, pages 482–487, Tampa FL, 2010.

[24] A. Makroglou, J. Li, and Y. Kuang. Mathematical models and software tools for the glucose–insulin regulatory system and diabetes: an overview. In *Applied Numerical Mathematics*, volume 56, pages 559–573, 2006.

[25] O. Ore. Theory on non–commutative polynomials. In *Annals of Mathematics*, volume 34, pages 480–508, 1933.

[26] S. Saab and R. Kae-Bey. Parameter identification of a dc motor: Experimental approach. In *The 8th IEEE Conference on Electronics, Circuits and Systems*, 2001.

[27] J-J. Slotine and W. Li. *Applied Nonlinear Control*. Englewood Cliffs, NJ, USA: Prentice Hall, 1991.

[28] A. Soria, R. Garrido, and A. Concha. Low cost closed loop identification of a dc motor. In *7th International Conference on Electrical Engineering*, pages 40–45, Seoul, South Korea, 2010.

[29] J. Swevers, F. Al-Bender, C. Ganseman, and T. Prajogo. An integrated friction model structure with improved presliding behavior for accurate friction compensation. In *IEEE Trans. on Automatic Control*, volume 45, pages 675–686, Seoul, South Korea, 1995.

[30] C. Taylor and A. Chotai. Non-minimum state–dependent ricatti equation and pole assignment control of nonlinear systems. In *19th International Conference on Systems Engineering*, volume 8, 2008.

[31] T. Tutunji. Dc motor identification using impulse response data. In *The International Conference on Computer as a Tool*, volume 2, pages 1734–1736, 2005.

[32] J. Van de Vegte. *Feedback Control Systems.* Englewood Cliffs, NJ, UAS: Prentice Hall, 1994.

[33] J. Wu, K. Su, and M. Cheng. Friction and disturbance compensation for speed control of servo control systems. In *36th Annual Conferenec on IEEE industrial Electronics Society*, pages 1890–1895, 2010.

[34] Y. Zheng and L. Cao. Transfer function description for nonlinear systems. In *Journal of East China Normal University*, volume 2, pages 15–26, 1995.