A Novel Method for Power System Stabilizer Design

A DISSERTATION
SUBMITTED TO THE FACULTY
OF ENGINEERING AND BUILT ENVIRONMENT
OF THE UNIVERSITY OF CAPE TOWN
IN FULFILMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

By
Lian Chen
April 2003

Declaration

I hereby declare that the work contained in this dissertation is my own, in both concept and execution. It is being submitted for the Degree of Doctor of Philosophy in Engineering at the University of Cape Town, South Africa. It has not been submitted for any degree or examination to any other university.

_____

Candidate – Lian Chen

_____

Supervisor- Professor Alexander Petroianu

Abstract

Power system stability is defined as the condition of a power system that enables it to remain in a state of operating equilibrium under normal operating conditions and to regain an acceptable state of equilibrium after being subjected to a finite disturbance. In the evaluation of stability, the focus is on the behavior of the power system when subjected to both large and small disturbances. Large disturbances are caused by severe changes in the power system, e.g. a short-circuit on a transmission line, loss of a large generator or load, loss of a tie-line between two systems. Small disturbances in the form of load changes take place continuously requiring the system to adjust to the changing conditions. The system should be capable of operating satisfactorily under these conditions and successfully supplying the maximum amount of load.

This dissertation deals with the use of Power System Stabilizers (PSS) to damp electromechanical oscillations arising from small disturbances. In particular, it focuses on three issues associated with the damping of these oscillations. These include ensuring robustness of PSS under changing operating conditions, maintaining or selecting the structure of the PSS and coordinating multiple PSS to ensure global power system robustness.

To address the issues outlined above, a new PSS design/tuning method has been developed. The method, called sub-optimal H_∞ PSS design/tuning, is based on H_∞ control theory. For the implementation of the sub-optimal H_∞ PSS design/tuning method, various standard optimization methods, such as Sequential Quadratic Programming (SQP), were investigated. However, power systems typically have multiple "modes" that result in the optimization problem being non-convex in nature. To overcome the issue of non-convexity, the optimization algorithm, embedded in the

sub-optimal H_∞ PSS design/tuning method, is based on Population Based Incremental Learning (PBIL).

This new sub-optimal H_∞ design/tuning method has a number of important features. The method allows for the selection of the PSS structure i.e. the designer can select the order and structure of the PSS. The method can be applied to the full model of the power system i.e. there is no need for using a reduced-order model. The method is based on H_∞ control theory i.e. it uses robustness as a key objective. The method ensures adequate damping of the electromechanical oscillations of the power system. The method is suitable for optimizing existing PSS in a power system. This method improves the overall damping of the system and does not affect the observability of the system poles. To demonstrate the effectiveness of the sub-optimal H_∞ PSS design/tuning method, a number of case studies are presented in the thesis.

The sub-optimal H_∞ design/tuning method is extended to allow for the coordinated tuning of multiple controllers. The ability to tune multiple controllers in a coordinated manner allows the designer to focus on the overall stability and robustness of the power system, rather than focusing just on, the local stability of the system as viewed from the generator where the controllers are connected.

Dedicated to:

My father

Professor Heng Chen

Preface

Power engineering study has a strong link to the existing problems in power utilities. In particular, the study of power system stability has been a popular topic in power engineering field for many decades. As science and technology progresses, we have the opportunity to find new and better solutions to old problems. In addition, with innovation, often a new set of problems occur and these problems need solutions. This is particularly true in modern power systems. It is my belief that power system research should pay more attention to improve existing power system performance by applying modern science and technology. While investigating South Africa's energy provider's (Eskom) low frequency oscillation problem, it was noticed that many of the modern PSS design methods required the replacement of hardware in the existing power systems, rather than looking for solutions to exploit the existing hardware. From my literature study, I found that not much has been done in application of modern control theory to the tuning of existing power system controllers. This was the basis for starting my research in applying modern control theory to design/tune power system stabilizers.

I would like to thank my family for their love and constant support. This thesis is particularly dedicated to my father, Professor Heng Chen who devoted his whole career to power system education and research in China. His passion for power system engineering motivated me to study and work in this field. My only regret is that I did not finish this thesis before he passed away.

I would like to thank Professor A. Petroianu for giving me this opportunity to do my post-graduate study at University of Cape Town. He not only supervised me on the thesis study, but also gave me advice on life, and surviving in country so different from

my own. Through all these years, his kindness, his toughness, his patience and his support gave me the strength and confidence to complete this thesis.

I would also like to thank Johan for his love and constant encouragement. His patience and understanding helped me to build confidence continuously. I have not only learnt technology, but also the right attitude towards my work.

I would like to thank my friends and colleagues for their friendship and encouragement, in particular Dr S. Ahmed, Dr Y. Bao and Dr J. Schoonees for their invaluable suggestions.

I would like to thank the University of Cape Town and Eskom for their financial support.

Table of Contents

Table of Figures

Nomenclature

$\Re$	Real number space
$\Delta\delta$	Generator rotor speed deviation
$\Delta\omega$	Generator frequency deviation
ω_{gm}	Gain margin frequency (rad/s)
ΔP_{elec}	Electrical power deviation
ω_{bm}	Phase margin frequency (rad/s)
ΔP_{mech}	Turbine mechanical power deviation
ΔV_{ref}	Generator reference voltage deviation
ΔV_t	Generator terminal voltage deviation
3M9B	Three-machine-nine-bus
$\|\bullet\|_\infty$	Infinity norm
AVR	Automatic Voltage Regulators
CL	Competitive Learning
CPSS	Conventional PSS; Classical PSS
ESKOM	Electricity supply commission (South Africa)
GA	Genetic Algorithm
Gamma(γ)	Additional weighting parameter
$H_\infty(a)$	H infinity norm of the system(a)
LR	Learning rate
MIMO	Multiple Input Multiple Output
MP	Probability of mutation
PBIL	Population Based Incremental Learning
P_{elec}	Electrical power
PID	Proportional Integral Differential
P_{mech}	Generator mechanical power
PSS	Power System Stabilizer
PSS/E	Power system simulator for engineers
SISO	Single Input Single Output
SMIB	Single Machine Infinite Bus
SQP	Sequential Quadratic Programming

Chapter 1
Introduction

1.1 Problem Statement

This thesis deals with the improvement of power system small-disturbance stability using Power System Stabilisers (PSS). Three aspects are addressed, namely:

- Design/tuning robust PSS
- Application of Population-Based-Incremental-Learning (PBIL) to the PSS design/tuning.
- Coordinated design/tuning of multiple PSS and PSS-Speed governor

This chapter gives a brief introduction of the methods used to solve power system small-disturbance stability problems. The objective is to provide an overview of power system small-disturbance stability phenomena, to describe some of the problems that exist in power utilities and to outline the objectives of our research work.

1.2 Basic Concepts and Definitions

Power system stability is defined as the condition of a power system that enables the system to remain in a state of operating equilibrium under normal operating conditions and to regain an acceptable state of equilibrium after being subjected to a finite disturbance.

In the evaluation of stability, the focus is on the behaviour of the power system when subjected to both *large* and *small* disturbances. Large disturbances are caused by severe changes in the power system, e.g., a short-circuit on a transmission line, loss of a large generator or load, loss of a tie-line between two systems.

Small disturbances in the form of load changes take place continuously, and the system adjusts itself to the changing conditions. The system must be able to operate satisfactorily under these conditions and successfully supply the maximum amount of load.

Small-disturbance stability phenomena can be classified into two categories, namely

- Small-disturbance rotor angle stability
- Small-disturbance voltage stability.

1.2.1 Small-disturbance Rotor Angle Stability

Small-disturbance rotor angle stability is the ability of a power system to maintain synchronism under small disturbances. Such disturbances occur continually in the system because of variations in loads and generation. Instability that may result can be of two forms: (i) a steady increase in rotor angle due to insufficient synchronizing torque, or (ii) rotor oscillations of increasing amplitude due to insufficient damping torque.

The nature of system response to small disturbances depends on a number of factors including the initial operating point, the transmission system strength, and the type of generator excitation controls. For a generator, without automatic voltage regulator

(AVR), insufficient synchronizing torque contributes to instability. This instability results in a non-oscillatory behavior.

For a generator, with AVR, the transient stability margin of the power system is improved. However, the small-disturbance stability margin deteriorates. The damping of the electromechanical *mode* is reduced as the gain of the AVR is increased, thus resulting in sustained low frequency oscillations. If the oscillations continually increase in amplitude, the system will become unstable. *Mode* is the technical term for a specific oscillation pattern. It is often used, more loosely, to refer to an oscillation at a specific frequency.

The analysis of the electromechanical oscillations intrinsic in power systems is necessary for stability problem.

1.2.2 Small-disturbance Voltage Stability

Small-disturbance voltage stability is the ability of a power system to maintain steady acceptable voltages at all buses in the system under small disturbances. This form of stability is determined by the characteristic of load, continuous controls, and discrete controls at a given instant of time. Instability that may result can be of two forms:

(i) The power system's inability of meeting the demand for reactive power.

(ii) The voltage drop that occurs when active power and reactive power flow through inductive reactances associated with the transmission network.

The basic processes contributing to small-disturbance voltage instability are of a steady-state nature. Therefore, steady-state analysis can be effectively used to determine stability margins, identify factors influencing stability, and examine a wide range of system conditions under a large number of post-contingency scenarios.

Often the angle and voltage instabilities are related. One may lead to the other and the distinction between angle and voltage instabilities are not always clear. In today's power systems, small-disturbance stability is largely a problem of insufficient damping of oscillations. However, to understand the underlying causes of the problems and thereby developing an appropriate design and operating procedures, it is necessary to analyze the two phenomena (i.e. angle stability and voltage stability) separately.

The study of small-disturbance stability can be decomposed into the study of three types of oscillations, namely local oscillations, inter-area oscillations and inter-plant oscillations.

Inter-area oscillations occur in the frequency range 0.1 to 0.8 Hz. These oscillations result from the exchange of power from one group of machines to another group of machines via a tie-line. The local oscillations occur in the frequency range 0.8 to 2 Hz. These oscillations result from the exchange of power from a single machine to an electrically strong network. Inter-plant oscillations occur in the frequency range 2 to 3 Hz. These oscillations result from the exchange of power between electrical units that are electrically close to each other.

1.3 Historical Overview of Small-Disturbance Stability Problems

Power system stability is a complex subject that has received a great deal of attention for several decades. Early stability problems were associated with remote hydroelectric generating stations feeding into metropolitan load centers over long-distance transmission. Such systems were operated close to their steady-state stability limits. In a few instances, instability occurred during steady-state operation, but it occurred more frequently following short-circuits and other system disturbances. The stability problem was largely influenced by the strength of the transmission system, with instability being the result of insufficient synchronizing torque.

As power systems evolved and interconnections between independent systems were found to be economically attractive, the complexity of the stability problems increased. The theoretical work carried out in the 1920s and early 1930s laid the foundation for the industry's basic understanding of the power system stability phenomena [1]. The principal developments and knowledge of power system stability in this early period came about because of the study of long-distance transmission, rather than as an extension of synchronous machine theory. Analysis focused on the network. Generators were viewed as simple voltage sources behind fixed reactances, and loads were considered as constant impedances. Modeling systems in this manner was a practical necessity since the computation tools available during this period were suited for solving algebraic equations, but not differential equations.

In the early 1950s, electronic analog computers were used for analysis of special problems requiring detailed modeling of the synchronous machine, excitation system, and speed-governor. In the mid 1950s, the development of digital computer allowed improvement over network analyzer methods, enabling researchers to model power system elements (e.g. generators) dynamic characteristics and simulate larger power systems. Digital computers provided the ideal means for the study of stability problems associated with growth in interconnections between formerly separate power systems.

Beginning in the late 1950s and early 1960s most of the new generating units connected to power systems were equipped with continuously-acting voltage regulators. As these new generating units became a larger percentage of the generating capacity, researchers discovered that the actions of the Automatic Voltage Regulators (AVR) had a

detrimental impact upon the steady-state stability of the power system. During fault conditions, the AVR enhance power system transient stability by holding the generator in synchronism with the power system. An adverse effect of these high-gain excitation systems is a decrease in damping torque of the generators, leading to a power system sensitive to oscillatory instability. These oscillations are characterized by low frequency but increasing magnitude. The oscillations often persist for long periods and in some cases limit the power transfer capability of the power system.

Before 1970, two popular control methods were used to implement AVR: one was proportional control and the other was proportional-integral-differential (PID) control. Along with the development of power systems, the shortcoming of PID controllers became more obvious.

Since the early 1970s, supplementary excitation controllers, commonly referred to as power system stabilizers (PSS), have been added to generators to counteract the negative damping effect from the high-gain AVR. A PSS uses a signal derived from the generator's electrical torque to damp rotor oscillations. Through the exchange of electromagnetic energy and mechanical energy, the oscillatory mode of a generator can be effectively damped [2-6].

1.4 Application of PSS for Improving Power System Small-Disturbance Stability

The PSS feedback loop provides a practical way of enhancing the small-disturbance stability of a power system. The PSS is connected to provide an additional signal to the voltage loop as shown in Figure 1. The feedback signal is derived from the generator rotor speed deviation $\Delta\delta$, the generator frequency deviation $\Delta\omega$, or the generator terminal active power deviation ΔP_{elec}.

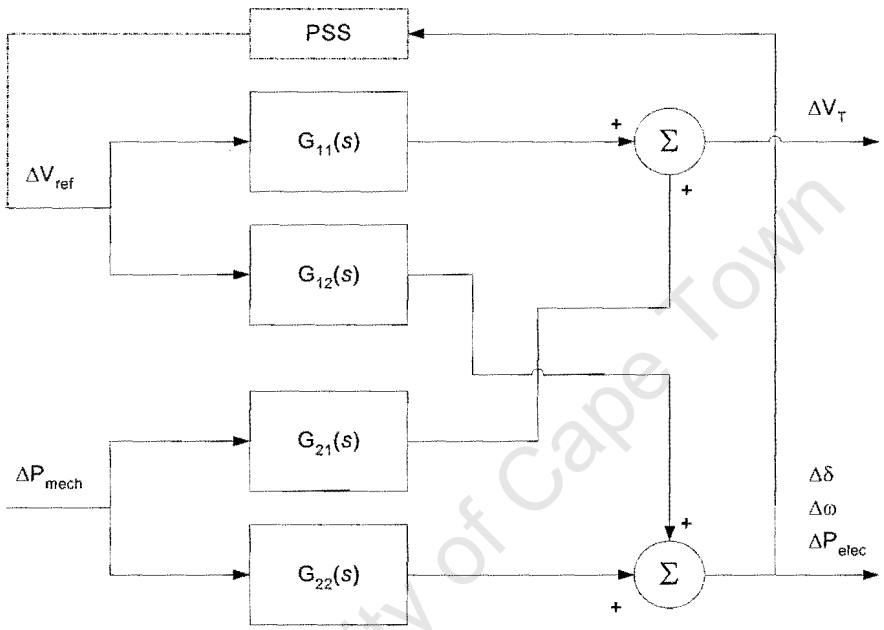

Figure 1 Implementation of a PSS in a generator control system

Figure 1 is used to illustrate the implementation of a PSS in the two principal control loops of a generator control system. The transfer function $G_{11}(s)$ represents the open loop power system response of the terminal voltage ΔV_T due to a step change in the reference voltage ΔV_{ref}. The transfer function $G_{12}(s)$ represents the open loop power system response of the electrical power ΔP_{elec} due to a step in the reference voltage ΔV_{ref}. The transfer function $G_{21}(s)$ represents the open loop power system response of the terminal voltage ΔV_T due to a step in the mechanical power ΔP_{mech}. The transfer function $G_{22}(s)$ represents the open loop power system response of the electrical power ΔP_{elec} due to a step in the mechanical power ΔP_{mech}.

Many methods have been developed to design PSS for Power Systems. In the following section, an overview of PSS design methods is presented.

1.5 PSS Design Methods

PSS design methods can be divided into two groups, namely linear PSS design methods and non-linear PSS design methods. Linear PSS design methods requires that the power system model (which is non-linear) be linearised. To linearise a power system model requires selecting an operating point. The resultant linearised system of equations is only valid for the selected operating point. The PSS design methods described in this section are all linear PSS design methods.

1.5.1 Classical PSS Design Methods

Classical lead-lag phase compensation is a common method used to design PSS. This method, which has been developed based on a single machine infinite bus power system model, typically deals with the local oscillation mode only. A detailed discussion of the application of classical design techniques to conventional PSS is provided in Appendix A.

In larger interconnected power systems, inter-area and inter-plant modes are more pronounced. PSS design/tuning procedures need to take local modes (0.1-0.8 Hz), inter-area modes (0.8-2.0 Hz) and inter-plant modes (2.0 to 3.0 Hz) into account. In order to improve the performance of conventional PSS, many design methods have been developed. Some of the more common design methods include root locus [7-8], optimal setting methods based on eigenvalue sensitivity analysis [9-10], self-tuning PSS [11-12], eigenvalue and frequency response techniques [13], Prony analysis techniques [14-15]. PSS obtained by using these methods improve the overall damping of power systems. However, due to the system uncertainties, the PSS design methods mentioned above cannot maintain adequate system stability. The uncertainties arise due to incomplete knowledge of the system parameters, neglected high frequency dynamics, parameter variations caused by changes in operating conditions or system faults, linearization of the power system model, sensitivity to actuator noise and various other external disturbances.

An important issue in the design of PSS is robustness. The robustness of the PSS can be defined as the ability of the PSS to maintain the performance and the stability of the closed-loop system for the entire range of operating conditions. None of the methods mentioned above use robustness as one of the design criteria. PSS obtained using these design methods are only effective for a specific operating condition and are not guaranteed to function correctly at different operating conditions. Changes in operating conditions can cause a well-tuned PSS to become ineffective. Therefore, there is a need to find alternative PSS design/tuning methods, which consider the robustness requirements.

1.5.2 Modern PSS Design Methods

Recently, researchers have been focusing on the application of modern control theory to the PSS design/tuning problem, in an attempt to overcome the shortcomings that methods discussed above have. Some of the well-known methods include pole placement [16,36,41], variable structure control [17], adaptive control [18-20, 38-39], fuzzy logic [21-23, 34] and H_∞ control [24-26, 35,40].

Pole Placement uses the measurable states (state variables) of a system, and places the poles in the desired location. It allows for the shaping of the dynamic response of the system. Although pole placement attempts to ensure that the PSS is optimal over a wide-range of operating conditions, it has a number of drawbacks. Firstly, the size of the state-space model of a typical power system makes the design process complex and computationally intensive. Secondly, many of the states in a power system are not observable or controllable, and therefore observers are required, which results in a very complex controller structure.

Variable structure PSS (VSPSS) are insensitive to system parameter variations. In VSPSS, the controller is allowed to change structure, i.e. to switch at any instant from one PSS structure to another. The advantage is that the system dynamic performance is insensitive to wide changes in system parameters. The PSS is considered to be quite robust.

Adaptive stabilisers, unlike the conventional fixed parameter stabilisers, determines a new set of control parameters as changes occur in system configurations and load levels, thereby ensuring that PSS parameters are optimal for a wide range of operating conditions. The advantage of adaptive control is that it is robust in the sense that it can operate over a wide range of conditions. The robustness is dependent on the complexity of the reference system and speed of the controller. The disadvantage is the complexity of the design and the cost of implementing the controller. Another disadvantage is that in the presence of inevitable high frequency dynamics, adaptive controllers may become non-robust.

Fuzzy logic is being used more and more in the PSS design. The speed deviation and acceleration of the machine are used as input states to the fuzzy PSS. Using a fuzzy relation matrix (a relation matrix gives the relationship between stabiliser inputs and outputs) a set of fuzzy logic operations are performed to yield a stabiliser output. A fuzzy logic implementation can operate over a wide range of operating conditions and therefore can be considered robust. The disadvantages are similar to that of the adaptive PSS.

H_∞ is a frequency-domain technique, which was proposed by Zames in 1981 [27]. The benefit of H_∞, being a frequency-domain technique, is that most design specifications used in lead-lag compensation techniques can easily be included in the H_∞ design specification. Furthermore, sensitivity reduction and robustness are elegantly formulated.

The H_∞ approach involves designing a PSS with the following objectives:

- To minimize the effects of disturbances in the outputs of the power system, subject to the constraint of power system stability; i.e. improve damping of the system.
- To obtain a PSS that will perform satisfactorily under a wide range of system operating conditions; i.e. improve robustness of the PSS.

Optimal H_∞ PSS design uses robustness as one of its design criteria. Therefore, it is insensitive to the changes of the operating conditions [24]. A detailed discussion on optimal H_∞ PSS design is in Appendix B. Optimal H_∞ design/tuning has several shortcomings. The following shortcomings make the method impractical:

- This method does not improve the damping of dominant modes. Therefore, if the dominant mode is the weakly damped "problem" mode, then the "problem" modes will remain weakly damped.
- This method affects the observability of the open loop poles. The PSS' zeros and poles cancel the poles and zeros of the open-loop systems, therefore, the problem modes become invisible in the closed loop system.
- This method results in a higher order PSS that has the same or even higher order than the power system.

1.5.3 Comments

From the discussion in *1.5.1* and *1.5.2*, most of the methods result in PSS with a complex structure. Since the PSS used in power systems are normally first or second order controllers, the existing PSS would need to be replaced. Replacing existing PSS with more complex PSS is not economically feasible for power systems.

Therefore, there is a need to find a method that maintains the existing PSS structure, while improving the robustness [28] and damping factors of the power system.

1.6 Coordinated Design/Tuning of Multiple PSS Methods

The primary objective of PSS coordination is to estimate the stabilizer settings such that the desired dynamic performance and robustness of the multi-machine power system is achieved. Many different coordinated design/tuning multiple PSS methods have been presented in the literature. Three distinct approaches are presented:

- Decentralized control, where the PSS are partly decoupled, but limited data is transferred between PSS [29].
- Decentralized control, where the PSS are fully decoupled and no real-time data is transferred between PSS [30][37].
- Centralized control can also be considered as a form of coordinated control, where a centralized multi-input, multi-output PSS is used. This method requires system-wide communication, thus resulting in more points of failure and an extra cost factor [29].

Our research focuses on the decentralized coordinated design/tuning of multiple PSS.

1.6.1 Decentralized Coordinated Design/Tuning of Multiple PSS by Using Limited System-wide Communication

Decentralized coordinated design/tuning of multiple PSS with limited system-wide communication presented in [29] is based on a decentralized adaptive control scheme, with generators that tend to dynamically interact strongly. PSS are coordinated designed/tuned by communicating the control inputs between each other. The data is used in such a way that, should a communication failure result, the PSS can still function and stabilize the system. The disadvantage of this method is the PSS's complexity and the need for communication between PSSs. This implies that the failure on the PSS has been increased. The other disadvantage of this method is that without the feedback between PSSs, although the system is still stable, the stability margin is reduced and therefore the overall robustness.

1.6.2 Decentralized Coordinated Design/Tuning of Multiple PSS by Using Augmented Ricatti Equations

The key feature of the method presented in [30] is the use of augmented Ricatti equations to satisfy both the coordination and robustness requirements. It is stressed that in practice PSS are designed in separation and no account of the interactions between PSS is considered.

To ensure the robustness of the PSS, a more systematic approach to power system stability is required and it is necessary to the a-priori interaction between PSS and the time-varying non-linear nature of the power system. This method approaches the design by considering the issue of power system modeling. Therefore, the disadvantage of this method is that the PSS's performance very much depends on the accuracy of the power system models.

1.6.3 Decentralized Coordinated Tuning of Multiple PSS by Using Sequential Application of IST Technique

The basis of the method presented in [31] is the sequential application of the Integral Least-Squares Technique (IST). This method is more superior to methods that attempt to tune PSS simultaneously. This method ignores the fact that the machines interact. The sequence in which the PSS are tuned is important and ultimately determines the overall stability achieved for the system. The procedure consists of introducing one PSS on each machine at a time and observing which of the poorly damped modes is improved the most. The lowest frequency mode is the one that usually is the most important. The poorly damped modes will thus be damped in the order of increasing frequency. The limitation on this method is that the IST technique is only applicable to stable system and therefore it is necessary to make the system stable first.

1.6.4 Decentralized Coordinated Design/Tuning of Multiple PSS Using a Parameter Optimization Method

Decentralized coordinated design/tuning of multiple PSS using a parameter optimization method combines a number of well-known techniques, in conjunction with parameter optimization. The method essentially addresses the issue of maximizing the damping of electromechanical oscillations, by applying the optimization method to the problem of coordinated PSS and AVR tuning in a multi-machine system. The algorithm is based on a gradient projection method, which attempts to force the critical modes as far as possible to the left in the complex Laplace domain [32].

This method addresses the optimization in two steps, namely (1) selection of optimal generator/s for PSS placement and (2) coordinated tuning of multiple PSS parameters.

1.7 Coordinated Design/Tuning of PSS-Speed Governor

The aim of designing a PSS with speed as input is to introduce a damping torque (a torque of electromagnetic origin proportional to machine speed) on the generator shaft. This causes the modes of rotor oscillation of the generator to be shifted to the left in the *s*-space [33]. The procedure for the design of PSS using speed as input is based on the concepts of synchronizing and damping torques and can also be applied to a speed-governor (with speed as input) of a suitable prime-mover. The purpose of a governor is to assist in the damping of low frequency rotor modes, particularly lightly damped inter-area modes. Certain PSS may be ineffective in contributing to the damping of such modes because the speed state has a low participation in the mode, or because the deleterious effect of interactions between PSS is significant. The latter problem does not arise in the case of a speed governor-based damping controller because this controller is not directly coupled to the generator and the electrical network. Therefore, a speed governor is likely to be more robust to changes in operating conditions on the system than a PSS.

A speed governor would be required to compensate for the phase lag of the transfer function between the speed-reference (as input) and the prime-mover shaft-torque (as output). A conventional steam governor-turbine system with reheat can contribute negative damping at a low frequency mode, but the required phase-lead of the compensator may be large. Current practice has not favored the use of speed governor for damping oscillation, due to the excessive wear of mechanical parts.

Because multiple PSS (with different input signals) coordinated design/tuning is necessary for large power systems, the possibility of the coordinated tuning of PSS and speed governor is worth investigating [26]. There is a need to develop a method to coordinate the tuning of speed governors and PSS, thereby ensuring the overall robustness of the power system to changes in operating conditions.

1.8 Outline of Thesis

A method called "Sub-optimal H_∞ PSS design/tuning" will be presented in this thesis. The method is based on a specialized genetic algorithm called Population-Based-Incremental-Learning. The method will be applied to multi-machine power systems, which utilize more than one PSS. The thesis will also cover the area of coordinated design/tuning multiple power system controllers.

Figure 2 is the outline of the thesis.

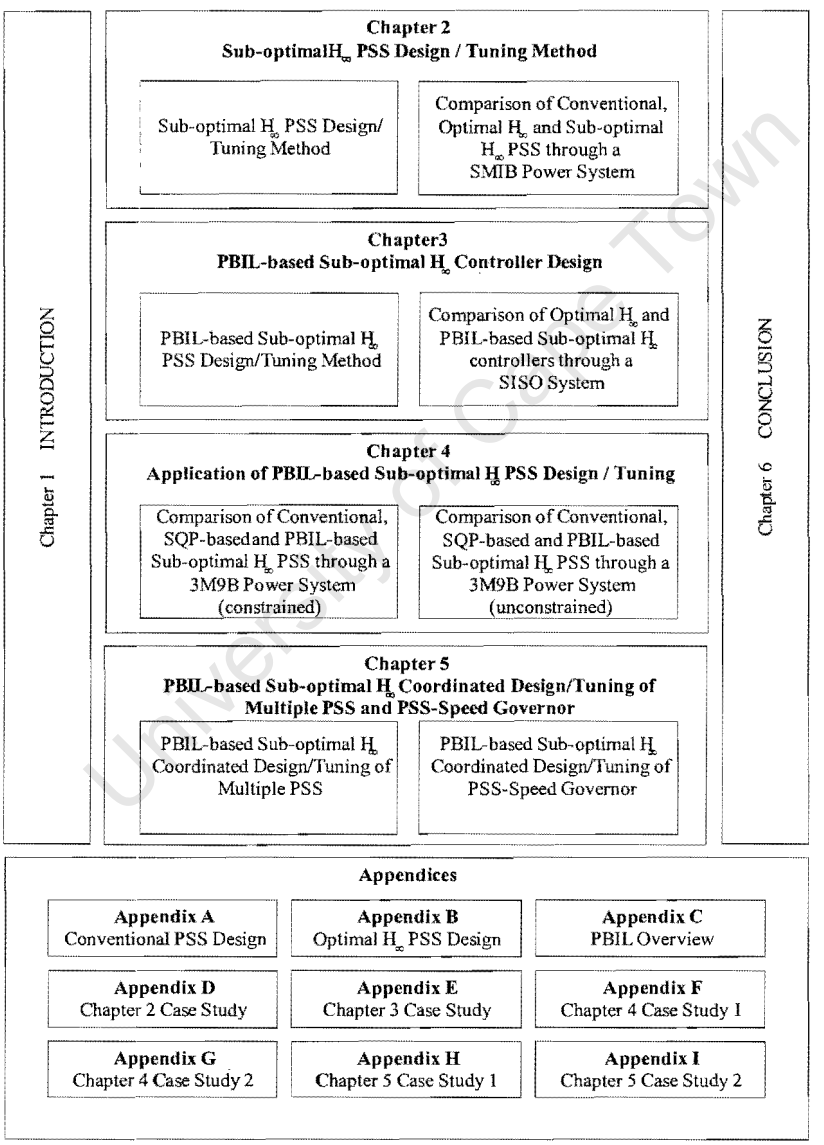

Figure 2 Thesis outline

1.9 Simulation Tools

In this thesis, the software package Power System Simulator for Engineers (PSS/E) has been used to model and simulate the behaviour of the power systems investigated. The load flow, dynamics, linear systems analysis and network equivalents tools from PSS/E were used.

In the linear systems analysis tool, PSS/E generates a set of state space matrices, which represents the linearised mathematical model of the power system. These state space matrices were transferred into the software package MATLAB for the purposes of designing the PSS. The Optimization and Robust Control toolboxes were used in this thesis.

1.10 References

[1] E. Kimbark, "Power System Stability", John Wiley & Sons, Inc., 1948

[2] P. Kundur, "Power System Stability and Control", Prentice-Hall, 1993

[3] Y. Yu, "Electric Power System Dynamics", Academic Press, 1980

[4] F.P. de Mello, C. Concordia, "Concepts of Synchronous Machine Stability as Affected by Excitation Control", IEEE Trans. on PAS, Vol. PAS-88, No. 4, April 1969, pp316-329

[5] E.V. Larsen, D.A. Swann, "Applying Power Stabilizers", IEEE Trans. on PAS, Vol. PAS-100, No. 6, June 1981, pp3017-3046

[6] H.B. Gooi, et al. "Coordinated Multi-machine Stabiliser Settings without Eigen-value Drift", IEEE Trans. on PAS, Vol. PAS-100, No. 8, August 1981, pp3879 - 3887

[7] E.L. Busby, "Dynamic Stability Improvement at Monticello Station – Analytical Study and Field Tests", IEEE Trans. on PAS, Vol PAS-98, No. 3, May 1979, pp889 – 897

[8] K. Bollinger, et al. "Power System Stabiliser Design using Root Locus Methods.", IEEE Trans. on PAS, Vol PAS-94, No.5, September 1975, pp1484-1488

[9] O.H. Abdalla, et al. "Co-ordinated Stabilization of a Multi-machine Power System.", IEEE Trans. on PAS, Vol. PAS-103, No. 3, March 1984, pp483 – 491

[10] Doi, S. Abe, "Coordinated Synthesis of System Stabiliser in Multimachine Power Systems", IEEE Trans. on PAS, Vol. PAS-103, No. 6, June 1984, pp1473 – 1479

[11] Y.Y Hsu, K.L Liou, "Design of Self-Tuning PID Power System Stabilizer for Synchronous Generators", IEEE Trans. on Energy Conversion, Vol.2, 1987, pp 343 – 348

[12] O. Malik, G. Hope, et al "A Multi-micro-computer based Dual-rate Self-tuning Power System Stabilizer", IEEE Trans. on Energy Conversion, Vol. 2, 1987, pp355-360

[13] N. Martins, L. Lima, "Eigenvalue and Frequency Domain Analysis of Small-signal Electromechanical Stability Problems", IEEE Publication, Eigenanalysis and Frequency Domain Methods for System Dynamic Stability (90TH0292-3-PWR) 1989, pp17-33

[14] D.J. Trudnowski, et al. "An Application of Prony Methods in PSS Design For Multimachine Systems", IEEE Trans. on Power Systems, Vol.6, No.1 February 1991, pp118-126

[15] C.E. Grund, et al. "Comparison of Prony and Eigenanalysis for Power System Control Design", IEEE Trans. on Power Systems, Vol.8, No.3, August 1993, pp964-971

[16] H. Chow, J.J. Sanchez-Gasca, "Pole-placement Designs of Power System Stabilisers", IEEE Trans. on Power Systems, Vol. 4, No. 1, February 1989, pp271-277

[17] M.L. Kothari, J. Nanda, K. Bhattacharya, "Design of Variable Structure Power System Stabilisers with Desired Eigenvalues in the Sliding Mode", IEE Proc.-C, Generation, Transmission and Distribution, Vol.140, No.4, July 1993, pp263-268

[18] Y.H. Song, "Novel Adaptive Control Scheme for Improving Power System Stability" IEE Proc.–C, Generation, Transmission and Distribution, Vol.139, No.5, September 1992, pp423-426

[19] A. Ghandakly, P. Idowu, "Design of a Model Reference Adaptive Stabilizer for the Exciter and Governor Loops of Power Generators", IEEE Trans. on Power Systems, Vol.5, No.3 August 1990, pp887-893

[20] S. Cheng, et al. "An Adaptive Synchronous Machine Stabilizer", IEEE Trans. on Power Systems, Vol. 1, No.3 August 1986, pp101-109

[21] M.A.M. Hassan, et al. "A Fuzzy Logic Based Stabilizer for a Synchronous Machine", IEEE Trans. on Energy Conversion, Vol.6, No.3 September 1991, pp407-413

[22] M.A.M. Hassan, O.P. Malik, "Implementation and Laboratory Test Results for a Fuzzy Logic Based Self -Tuned PSS", IEEE Trans. on Energy Conversion, Vol.8, No.2, June 1993, pp221-228

[23] Y.Y. Hsu, C.H. Cheng, "Design of Fuzzy Power System Stabilisers for Multimachine Power Systems", IEE Proceedings - C Generation Transmission and Distribution, Vol.137, No.3, May 1990, pp233-238

[24] R. Asharian, "A Robust H_∞ Power System Stabilizer With no Adverse Effect on Shaft Torsional Modes", IEEE Trans. on Energy Conversion, Vol. 9, No. 3, September 1994, pp475-481

[25] M. Klein, et al., "H_∞ Damping Controller Design in Large Power Systems", IEEE Trans. on Power Systems, Vol. 10, No. 1, February 1995, pp158-166

[26] S.S Ahmed, "Damping of Electromechanical Oscillations Using Power System Stabilisers", Ph.D Thesis, UCT 1995

[27] G. Zames, "Feedback and Optimal Sensitivity: Model Reference Transformations, Multiplicative Seminorms, and Approximate Inverses", IEEE Trans. on Automatic Control, AC-26, No. 2, April 1981

[28] M. Wishart, "Network Stability", Eskom, Technology Research and Investigation, Eskom Reference# TRR/E/95/EL163

[29] D.J. Trudnowski and et. al., "Coordination of Multiple Adapative PSS Units Using a Decentralized Control Scheme", IEEE Transactions on Power Systems, Vol 7, No 1, February 1992, pp 294-300

[30] A.S. Bazanella and et. al., "Coordinate Robust Controllers in Power Systems", IEEE Power Tech Stockholm Conference, Stockholm, Sweden, June 1995, pp 256-261

[31] M. Kothari, J. Nanda and K Bhattacharya, "Co-ordinate Tuning of Power System Stabilizers for a Multi-Machine System By Sequential Application of ISE Technique, IEE 2nd International Conference in Power System Control, Operations and Management, Honk Kong, December 1993, pp 299 – 304

[32] C. Vournas, N. Maratos, et.al. "Power System Stabilizer Co-ordination using a Parameter Optimization Method" , IEE Control 94, March 1994, pp 403-408

[33] "Cigre Technical Brochure on Impact of Interactions Among Power System Controls", Final draft, International Conference on Large High Voltage Electric Systems, Task Force 16 of Advisory Group 02 of Study Committee 38, Nov. 1999

[34] Y. Ruhua, H.J. Eghbali, et. al. , "An online adaptive neuro-fuzzy power system stabilizer for multimachine systems", IEEE Trans. on Power Systems, Vol 18 No. 1, Feb 2003, pp. 128-135

[35] K.A. Folly, N. Yorino, H. Sasaki, "Improving the robustness of H_∞-PSSs using the polynomial approach", IEEE Trans. on Power Systems, Vol 13 No. 4, Nov. 1998, pp. 1359 – 1364

[36] P.S. Rao, I. Sen, "Robust pole placement stabilizer design using linear matrix inequalities", IEEE Trans. on Power Systems, Vol 15 No. 1, Feb. 2000, pp. 313 – 319

[37] P. Zhang, A.H. Coonick, "Coordinated synthesis of PSS parameters in multi-machine power systems using the method of inequalities applied to genetic algorithms",IEEE Trans. on Power Systems, Vol. 15 No. 2, May 2000 pp. 811-816

[38] W. Gu, "System damping improvement using adaptive power system stabilizer", IEEE Canadian Conference on Electrical and Computer Engineering, Vol. 3, May 1999, pp. 1245 – 1247

[39] J. Ritonja, et. al., "Design of an adaptive power system stabilizer", Proceedings of the IEEE International Symposium on Industrial Electronics, Vol. 3, July 1999, pp. 1306 – 1311

[40] I. Ngamroo, "Design of robust H∞ PSS via normalized coprime factorization approach", The 2001 IEEE International Symposium on Circuits and Systems, Vol. 3, May 2001 pp. 129 - 132

[41] J.C.R. Ferraz, et. al., "Simultaneous partial pole placement for power system oscillation damping control", IEEE Power Engineering Society Winter Meeting, Vol. 3 , Jan. 2001, pp. 1154 – 1159

Chapter 2

Sub-optimal H_∞ Design/Tuning of PSS

In chapter 1, a brief introduction of the advantages and disadvantages of PSS design/tuning methods has been given. Optimal H_∞ design method is one of the most effective methods for designing robust PSS. However, as we discussed in chapter 1, this method has several shortcomings, namely:

1. Optimal H_∞ design method does not significantly improve the damping of dominant modes due to pole-zero cancellation[1].
2. Optimal H_∞ design method affects the observability in the model of the original power system.
3. Optimal H_∞ design method results in high-order PSS structures that are of the same order as the power system model or even higher.

One solution to improve the overall damping of the power system and ensure that PSS is of a lower order is to apply the optimal H_∞ design methodology to a reduced-order model of the power system. A number of methods exist to determine which characteristics of the power system should be included in the reduced-order model. In general, the least controllable and least observable modes are eliminated from the model, while the dominant modes are retained. PSS designed for a reduced-order model result in lower order structures improve the overall damping of dominant modes. The disadvantage of model reduction is that only the dominant modes of the power system are taken into account, while other system dynamics are not considered.

An alternative method for design/tuning of PSS has been developed based on optimal H_∞ control theory. This method has the advantages of optimal H_∞ design method (i.e. ensure the robustness), while addressing the shortcomings of optimal H_∞ design method, and eliminating the need for using a reduced-order model of the power system. This method is called the sub-optimal H_∞ design/tuning method. It has the following features:

[1] Methods, like bilinear transformations, may be used to overcome this effect in Optimal H_∞ control.

- Allows for the selection of the PSS structure, i.e. the designer can select the order and parameter limitation ranges of the PSS.
- Can be applied to the full model of the power system, i.e. no need for using a reduced-order model.
- Is based on H_∞ control theory and key objective is the overall robustness, i.e., the obtained PSS works over a wide range of operating conditions.
- Ensures adequate damping of the electromechanical oscillations of the power system.
- Is suitable for optimizing existing PSS in a power system, i.e. this method is practical for the utilities.
- Improves the overall damping of the system and does not affect the observability of the system poles.

In sub-optimal H_∞ design/tuning method, PSS parameters are obtained by using a numerical optimization technique. The PSS, with the unknown parameters, is placed in a feedback control path with an augmented open loop power system. The resulting closed loop power system contains the unknown PSS parameters. The objective function of the optimization problem is to minimize the H_∞ norm of the closed loop, in order to maximize the robustness of the PSS. The constraints of the optimization problem are the stability of the PSS, the limits "on" the values of the PSS parameters and the desired damping of the closed loop power system. The following sections present sub-optimal H_∞ PSS design/tuning method in detail.

2.1 Formulation of PSS Design/Tuning Problem

In order to formulate the PSS design/tuning problem, we need to determine the closed loop response of the augmented system. Figure 3 gives the control configuration of the augmented system, where the power system is represented by the transfer function ***G(s)*** and the PSS is represented by the transfer function ***Q(s)***. Transfer functions ***G(s)*** and ***Q(s)*** have the following general form defined in equation 2.1

$$F(s) = \frac{K\prod_{m=1}^{M}(s+z_m)}{\prod_{l=1}^{L}(s+p_l)} \qquad 2.1$$

Where:

K	is a constant gain.
z_m	has the form $z_m = \alpha_{zm} \pm j\beta_{zm}$ and $\alpha_{zm}, \beta_{zm} \in \Re$
p_l	has the form $p_l = \alpha_{pl} \pm j\beta_{pl}$ and $\alpha_{pl}, \beta_{pl} \in \Re$
$-z_m$	is the m-th finite zero of $F(s)$
$-p_l$	is the l-th finite pole of $F(s)$
M	is the number of zeros
L	is the number of poles

If $M<L$ the transfer function $F(s)$ is strictly proper. In terms of an nth order differential equation, a strictly proper system has higher derivatives of the output variable than of the input variable.

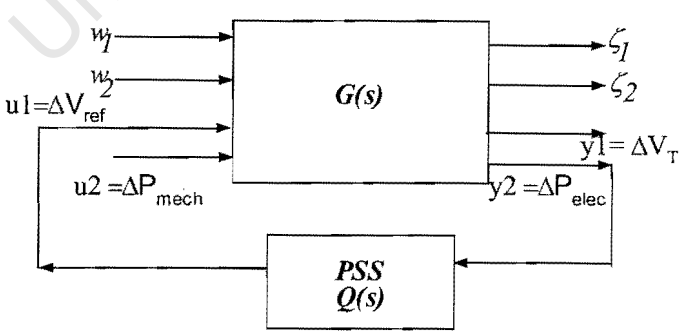

Figure 3 Control configuration

The state-space description of a 2-input 2-output power system model, $G(s)$, is defined by equation 2.2.

$$\begin{aligned} \dot{x} &= Ax + \begin{bmatrix} B_2^1 & B_2^2 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} + B_1 w \\ \varsigma &= C_1 x + \begin{bmatrix} D_{12}^1 & D_{12}^2 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} + D_{11} w \\ y &= \begin{bmatrix} C_2^1 \\ C_2^2 \end{bmatrix} x + D_{22} u + D_{21} w \end{aligned} \qquad 2.2$$

Where:

$x \in R^n$	*is the vector of power system state variables, such as machine speed, angles and fluxes.*
u_i	*is the vector of the input states variables, such as reference voltage and mechanical power. where* $i = 1,2.$
$y \in R^2$	*is the vector of output states variables, such as electrical power, machine speed and bus voltages.*
$w(t) \in R^2$	*is the disturbance vector.*
$\varsigma(t) \in R^2$	*is the performance vector.*
n	*is the number of states of the equivalent open loop system.*
A	*is the coefficient matrix of the equivalent open loop system.*
B_2^1	*is the coefficient vector associated with* u_1 *(reference voltage).*
B_2^2	*is the coefficient vector associated with* u_2 *(mechanical power).*
C_2^1	*is the coefficient vector associated with* y_1 *(terminal voltage).*
C_2^2	*is the coefficient vector associated with* y_2 *(electrical power).*
$B_1, C_1, D_{11}, D_{12}, D_{21}, D_{22}$	*are constant coefficient matrices.*

The detailed derivation of the linearized power system equations is beyond the scope of this thesis. Detailed model development can be found in the references [1-3]. The assumption that the power system is strictly proper has been made, i.e. $D_{22} = 0$. This implies that no proportion of input appears directly in the output. Only with this assumption of having a strictly proper power system, the formulation of PSS design/tuning problem can be mathematically presented as done in this thesis.

The state-space model of the PSS, $\boldsymbol{Q(s)}$, is defined by equation 2.3.

$$\begin{aligned} \dot{z} &= Ez + Fu_c \\ y_c &= Gz + Hu_c \end{aligned} \qquad 2.3$$

Where:

$z \in R^{n_c}$	*is the PSS state vector.*
u_c	*is the PSS input.*
y_c	*is the PSS output.*
n_c	*is the number of PSS states.*
E,F,G,H	*are the coefficient matrices containing the unknown PSS parameters.*

In order to obtain the closed loop state-space of the system, the output of the power system y is connected to the input of the PSS u_c, i.e.:

$$y = C_2 x + D_{21} w + D_{22} u = u_c \qquad 2.4$$

The output of the PSS y_c is connected to the input of the power system u_i, i.e.:

$$u_i = y_c = Gz + Hu_c \qquad 2.5$$

Since we close the loop between ΔP_{elec} and ΔV_{ref} only, the state space description of the closed loop system can be expressed as follows:

$$\begin{aligned}\begin{bmatrix}\dot{x}\\ \dot{z}\end{bmatrix} &= \overline{A}\begin{bmatrix}x\\ z\end{bmatrix} + \overline{B}_2 u_2 + \overline{B}_1 w\\ \varsigma &= \overline{C}_1\begin{bmatrix}x\\ z\end{bmatrix} + D_{12}^2 u_2 + \overline{D}_1 w\\ y &= \begin{bmatrix}C_2^1\\ C_2^2\end{bmatrix} x + D_{21} w\end{aligned} \qquad 2.6$$

Where:

$$\overline{A} = \begin{bmatrix} A + B_2^1 H C_2^2 & B_2^1 G \\ F C_2^2 & E \end{bmatrix}$$

$$\overline{B}_1 = \begin{bmatrix} B_1^1 & B_1^2 \end{bmatrix} = \begin{bmatrix} B_2^1 H D_{21} + B_1 \\ F D_{21} \end{bmatrix}$$

$$\overline{B}_2 = \begin{bmatrix} B_2^2 \\ 0 \end{bmatrix}$$

$$\overline{C}_1 = \begin{bmatrix} C_1 + D_{12}^1 H C_2^2 & D_{12}^1 G \end{bmatrix}$$

$$\overline{D}_1 = D_{12}^1 H D_{21} + D_{11}$$

From the closed loop system given by equation 2.6, the following transfer functions are defined:

$$\begin{aligned} T_{11}(s) = T_{w_1\varsigma_1} &= \overline{C}_1^1 (s\overline{I} - \overline{A})^{-1} \overline{B}_1^1 + \overline{D}_1^1 \\ T_{12}(s) = T_{w_1\varsigma_2} &= \overline{C}_1^2 (s\overline{I} - \overline{A})^{-1} \overline{B}_1^1 + \overline{D}_1^2 \end{aligned} \qquad 2.7$$

The transfer functions $T_{ij}(s)$, where $i = 1 \quad and \quad j = \{1,2\}$, give the responses of the performance variable ς_j due to changes in the disturbance variable w_i. The two variables ς_1 and w_1 are associated with ΔV_T and ΔV_{ref}, while ς_2 and w_2 are associated with ΔP_{elec}. and ΔP_{mech}.

The objective is to minimize the H_∞ norm of the closed loop power system, in order to maximize the robustness of the PSS as a function of the changes in power system parameters. The objective function is defined by equation 2.8.

$$\min_{\xi}(J_1 = \|T_{12}(s)\|_\infty) \qquad 2.8$$

where $\xi = \{K, \alpha_{z1}, \alpha_{z2}, \ldots, \alpha_{zL}; \beta_{z1}, \beta_{z2}, \ldots, \beta_{zL}; \alpha_{p1}, \alpha_{p2}, \ldots, \alpha_{pM}; \beta_{p1}, \beta_{p2}, \ldots, \beta_{pM}\}$ are the unknown PSS parameters.

This objective function is subject to the following constraints:

$$\begin{aligned} &\Psi_q(T_{12}(s)) \geq \Psi^o \\ &\gamma_k^{\min} \leq \xi_k \leq \gamma_k^{\max} \\ &\mathrm{Re}\{\lambda_r(Q(s))\} < 0 \quad (\text{internal stability of the PSS}) \end{aligned} \qquad 2.9$$

where:

ξ *is the set of the unknown parameters of Q(s).*

$\gamma_k^{\min/\max}$ *is the lower/upper limit of* ξ_k *and* $\gamma_k^{\min/\max} \in \Re$

$\Psi_q(T(s))$ *is the damping factor of the q-th mode of the transfer function* $T_{12}(s)$.

Ψ^o *is the minimum damping factor.*

$\lambda_r(Q(s))$ *is the r-th eigenvalue of Q(s).*

In the next section, the procedure for finding the optimal solution to equation 2.8, subject to the constraints defined in equation 2.9, will be given.

2.2 Finding the Unknown PSS Parameters

2.2.1 General Problem

Equation 2.8, with the constraints defined by equation 2.9, is by definition in the form of a General Problem (GP) [4]. The set of design (unknown) parameters can be found using a parametric optimization procedure. By definition parametric optimization is used to find a set of design parameters, $xx = \{xx_1, xx_2, ..., xx_n\}$, that can in some way be optimal. The simple case is the maximization or minimization of some system characteristic that is dependent on xx. In a more advanced formulation, the object function $f(xx)$, to be minimized or maximized, may be subject to constraints in the form of equality constraints, $\Phi_i(xx) = 0 \ (i = 1,...m_e)$, inequality constraints, $\Phi_i(xx) \le 0 \ (i = m_e + 1,...m)$, and/or parameter bounds, xx_l , xx_u. A General Problem description is stated as:

$$\begin{aligned} & \underset{xx \in \Re^n}{\text{minimize}} \quad f(xx) \\ \text{subject to :} \quad & \Phi_i(xx) = 0 \quad i = 1,....m_e \\ & \Phi_i(xx) \le 0 \quad i = m_e + 1,...,m \\ & xx_l \le xx \le xx_u \end{aligned} \qquad 2.10$$

where xx is the vector of design parameters, ($xx \in \Re^n$), $f(xx)$ is the objective function that returns a scalar value ($f(xx) : \Re^n \to \Re$), and the vector function $\Phi(xx)$ returns the values of the equality and inequality constraints evaluated at xx ($\Phi(xx) : \Re^n \to \Re^m$)[5-9].

In the section *2.1*, the mathematical derivation of the objective function in terms of the power system transfer function, the unknown PSS parameters, power system performance uncertainties *($\zeta(t)$)*, and power system disturbance uncertainties *($w(t)$)*, was presented. This objective function is the basis for finding a robust PSS for the power system. Figure 4 shows the flow diagram of the formulation of the sub-optimal H_∞ PSS design/tuning procedure.

Step 1: Select the weighting functions $W_1(s)$, $W_2(s)$ and $W_3(s)$. The weighting functions provide the design specification in terms of the performance and robustness bounds in the frequency domain. (See Appendix B for information on selecting weighting functions.)

Step 2: Augment the plant. Augmentation adds additional outputs to the system. These additional outputs represent the penalized (weighted) error signal, control signal and output signal that are used to design the PSS, as shown in Figure 3.

Step 3: Select the PSS structure.

Step 4: Create the closed loop system (as represented by Equation 2.6). The closed loop system is formulated in terms of the unknown PSS parameters.

Step 5: Bound the unknown PSS parameters. In the case of a tuning problem, the ranges should be defined according to what the existing PSS allows. Bounding PSS parameters allows the numerical optimization to be more efficient.

Step 6: Randomly generate the values for the unknown PSS parameters within the preset bounds and determine the closed-loop transfer function of the power system. Calculate the H_∞ norm and the minimum damping.

Step 7: Validate that the system is stable, that the H_∞ norm is a minimum and damping is a maximum. If the solution is not optimal (i.e. has not converged), generate a new set of values for the unknown PSS parameters and re-iterate.

The process described in steps 5 through 7 is a description of a constrained parametric optimization problem formulation. A wide range of optimization methods exist [4] that may be suited to finding the optimal solution to the design/tuning procedure described in this chapter. Matlab® provides a number of parametric optimization methods, but

only Sequential Quadratic Programming (SQP) is suited for constrained optimization problems.

2.2.3 Sequential Quadratic Programming

Sequential Quadratic Programming (SQP) methods represent state-of-the-art in nonlinear programming methods [4]. SQP mimics Newton's method for unconstrained optimization, while allowing constraints to be defined. At each major iteration, approximations are made of the Hessian of the Lagrangian function using a quasi-Newton updating method. This approximation of the Hessian of the Lagrangian function is used to generate a Quadratic Programming sub-problem, the solution of which is used to form a search direction for a linear search procedure. If the problem is a so-called convex problem, then it is possible to find a global solution point [4].

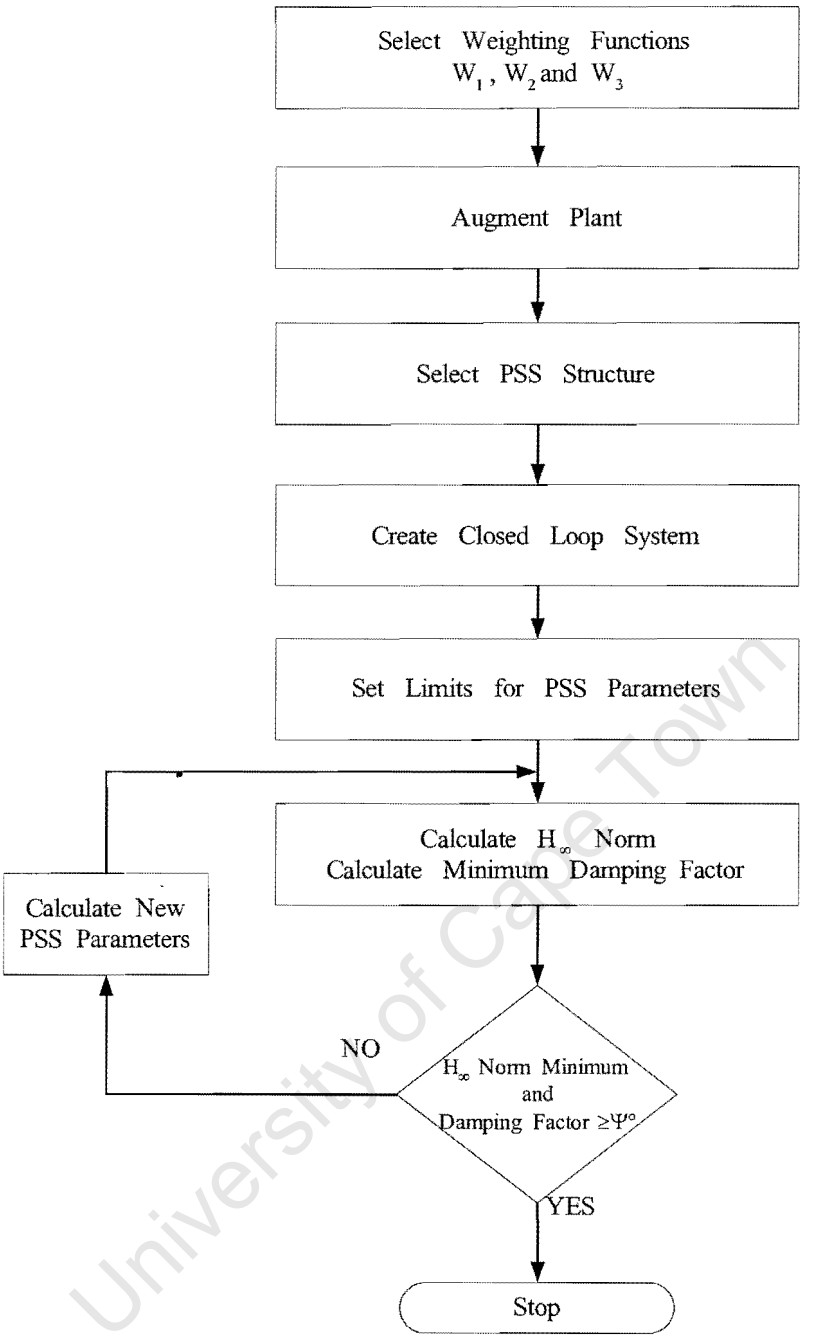

Figure 4 Flow diagram of sub-optimal H_∞ PSS design/tuning procedure

2.3 Case Study – Comparison among Classical, Optimal H_∞ and Sub-optimal H_∞ Design Procedure using a SMIB System

In the following case study, a comparison drawn among PSS design using classical design method (see Appendix A for overview of Classical PSS Design), optimal H_∞ design method (see Appendix B for overview of Optimal H_∞ PSS Design) and sub-optimal H_∞ design/tuning method is presented. The power system model used in the case study is a Single-Machine-Infinite-Bus (SMIB) system. Only the results for the case study are presented here. See Appendix D for the power system model, the MATLAB scripts.

The objectives of the case study are to:

1. Highlight how optimal H_∞design results in high-order PSS.
2. Highlight how an optimal H_∞ PSS does not affect the damping of the dominant mode.
3. Highlight how sub-optimal H_∞ PSS improves damping of the dominant mode while sustains a low-order structure.
4. Compare sub-optimal H_∞ PSS with conventional PSS and with optimal H_∞ PSS.

The power system under investigation is a SMIB system, as shown in Figure 5. In this case study, no model reduction is required.

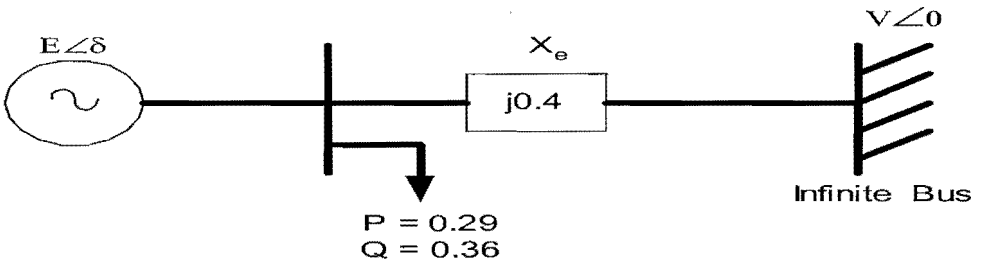

Figure 5 Single machine infinite bus system

Table 1 gives the open loop system's eigenvalues and damping factors. The "problem" mode, i.e. the weakly damped mode, is at a frequency of 5.8 rad/s with a damping factor of 0.0585.

Open Loop System		
Eigenvalue (Pole)	**Damping Factor**	**Frequency (rad/s)**
-3.40e-001 ± 5.79e+000i	**5.85e-002**	**5.80e+000**
-7.59e-001 ± 5.25e-001i	8.22e-001	9.23e-001
-1.35e+000	1.00e+000	1.35e+000
-1.42e+001	1.00e+000	1.42e+001
-1.65e+001	1.00e+000	1.65e+001
-2.80e+001	1.00e+000	2.80e+001

Table 1 Poles and damping factors of open loop SMIB system

2.3.1.1 Coventional PSS

The conventional PSS was designed for comparison using the method described in Appendix A. The PSS is a second order, with a transfer function:

$$K_{cpss}(s) = \frac{2.18(s^2 + 11.76s + 34.570)}{s^2 + 20.001s + 100.010} \qquad 2.11$$

2.3.1.2 Optimal H_∞ PSS

In Appendix B an overview on design an optimal H_∞ PSS is given. The optimal H_∞ PSS was designed for comparison using MATLAB Robust Control Toolbox. The resultant PSS structure is 11 orders, with a transfer function:

$$\begin{aligned} T_1(s) &= \frac{(s+226.76)(s+226.75)}{(s+(1054.6 \pm 1030.2i))} \\ T_2(s) &= \frac{(s+(0.33 \pm 5.7943i))(s+(0.759 \pm 0.525i))}{(s+(0.45 \pm 3.507i))(s+(0.219 \pm 0.000000069i))} \\ T_3(s) &= \frac{(s+28)(s+16.5)(s+14.2)(s+0.1304)(s+1.35)}{(s+43.2)(s+14.3)(s+6.23)(s+1.3)(s+0.138)} \\ K_{hopt}(s) &= T_1(s)T_2(s)T_3(s) \end{aligned} \qquad 2.12$$

2.3.1.3 Sub-optimal H_∞ PSS

For the purpose of comparison, the exact same structure as the conventional PSS was selected to design a PSS using the sub-optimal H_∞ design/tuning method described in this thesis. MATLAB's optimization toolbox was used to implement the optimization procedure. The detailed design procedure is given in Appendix D. The transfer function of the resultant sub-optimal H_∞PSS is:

$$K_{sohpss}(s) = \frac{1.931(s^2 + 9.243s + 2.878)}{s^2 + 7.821s + 19.066} \qquad 2.13$$

Closed Loop System with Conventional PSS		
Eigenvalue (Pole)	**Damping Factor**	**Frequency (rad/s)**
-7.58e-001 ± 5.45e-001i	8.12e-001	9.33e-001
-1.36e+000	1.00e+000	1.36e+000
-1.56e+000 ± 5.52e+000i	**2.71e-001**	**5.74e+000**
-7.81e+000	1.00e+000	7.81e+000
-9.37e+000 ± 8.86e+000i	7.26e-001	1.29e+001
-1.43e+001	1.00e+000	1.43e+001
-3.55e+001	1.00e+000	3.55e+001

Table 2 Poles and damping factors for closed loop SMIB system with conventional PSS

Table 2 gives the eigenvalues and damping factors for the closed loop system with a conventional PSS applied. The damping factor of the "problem" mode at frequency of 5.8 rad/s has increased. The damping factor is now 0.271, as opposed to 0.0585 in the open loop system.

Closed Loop System with Optimal H_∞ PSS		
Eigenvalue (Pole)	**Damping Factor**	**Frequency (rad/s)**
-0.142 ± 8.69e-3i	0.98	0.142
-0.22	1	0.22
-0.22	1	0.22
-0.34 ± 5.79i	**0.0585**	**5.8**
-0.759 ± 0.525i	0.822	0.923
-1.09	1	1.09
-1.11 ± 1.6i	0.569	1.95
-1.27	1	1.27
-1.35	1	1.35
-2.55 ± 0.128i	0.99	2.55
-14.2	1	14.2
-14.3	1	14.3
-16.5	1	16.5
-28	1	28
-43.3	1	43.3
-227 ± 3.05e-6i	1	227
-1005 ± 1003i	0.715	1470

Table 3 Poles and damping factors for closed loop SMIB system with optimal H_∞ PSS

Table 3 gives the eigenvalues and damping factors for the closed loop system with an optimal H_∞ PSS applied. The damping factor of the "problem" mode at frequency of 5.8 rad/s has not improved. The damping factor of the closed loop system is the same as the damping factor for the open loop system, i.e. 0.0585.

Closed Loop System with Sub-optimal H_∞ PSS		
Eigenvalue (Pole)	**Damping Factor**	**Frequency (rad/s)**
-7.79e-001 ± 5.16e-001i	8.34e-001	9.34e-001
-1.35e+000	1.00e+000	1.35e+000
-1.59e+000 ± 2.59e+000i	**5.24e-001**	**3.04e+000**
-2.46e+000 ± 8.70e+000i	**2.72e-001**	**9.04e+000**
-1.15e+001	1.00e+000	1.15e+001
-1.43e+001	1.00e+000	1.43e+001
-3.33e+001	1.00e+000	3.33e+001

Table 4 Poles and damping factors for closed loop SMIB system with sub-optimal H_∞ PSS

Table 4 gives the eigenvalues and damping factors for the closed loop system with a sub-optimal H_∞PSS applied. The original "problem" mode that was at a frequency of 5.8 rad/s has shifted to 3.04 rad/s. The mode is well damped with a damping factor of 0.524. Although the damping factor of the "problem" mode at frequency of 5.8 rad/s has increased, the original power system's characteristics have changed. The dominant mode for the closed-loop system is now located at a frequency of 9.04 rad/s with a damping factor of 0.272.

The choice of PSS structure for the sub-optimal PSS, results in an optimization problem that has more than one local optimum. SQP is only guaranteed to find a local-optimum and it relies heavily on a good initial guess [10]. This behavior is evident in the results in

Table 4, where SQP found a local optimum at 3.04 rad/s, and changed the overall frequency characteristics of the power system.

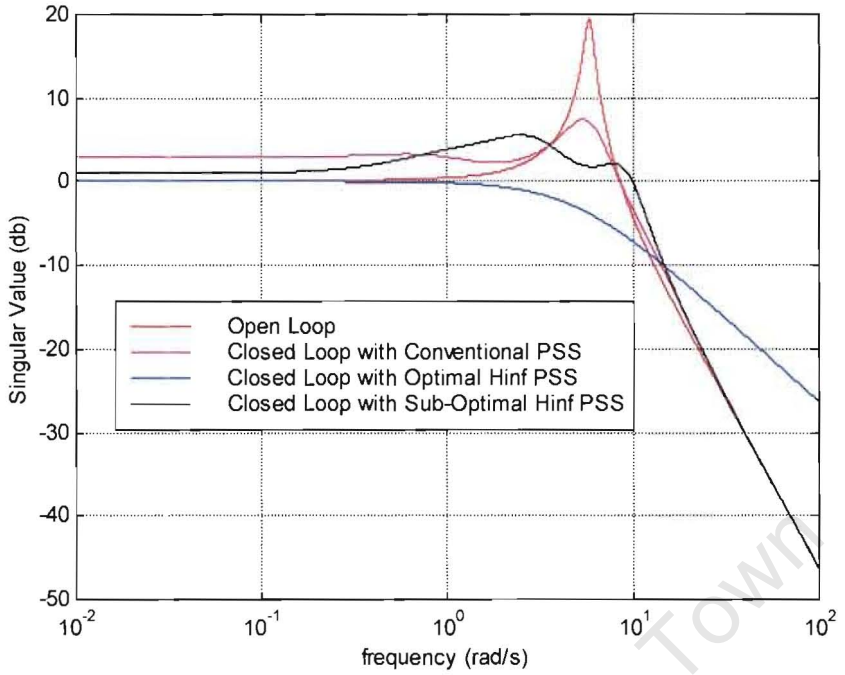

Figure 6 Plots of Singular value of open loop and closed loop SMIB system with various PSS applied

Figure 6 illustrates the plots of singular value of the transfer function between the output ΔP_{elec} and the input ΔV_{ref} for the open loop power system, and closed loop power system with conventional PSS, optimal H_∞ PSS and sub-optimal H_∞ PSS applied. The maximum value of each singular value plot corresponds to the H_∞ norm of the transfer function between ΔP_{elec} and ΔV_{ref}. From Figure 6 it can be seen that the sub-optimal H_∞ PSS has a smaller singular value than the conventional PSS. This means that the closed loop with sub-optimal H_∞ PSS has a lower H_∞ norm than that of the closed loop with conventional PSS. As discussed in Appendix B, the lower the H_∞ norm, the more robust the system is to changes in operating conditions. Thus, the system with the sub-optimal H_∞ PSS has better robustness than the system with the conventional PSS. The blue plot shows how optimal H_∞ PSS affects the system; it guarantees the robustness for the closed loop system.

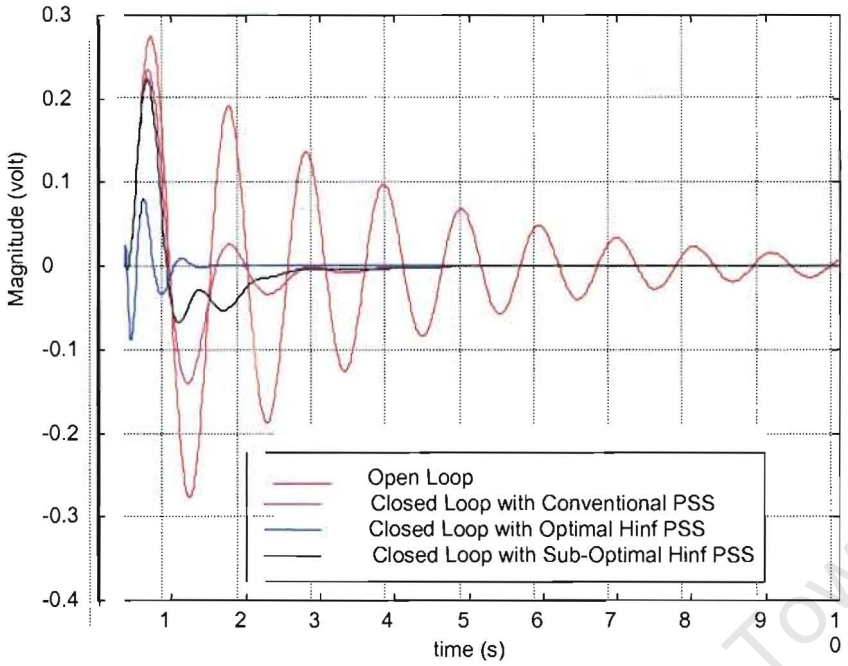

Figure 7 Comparison of step response plots of closed loop SMIB system with various PSS and the open loop SMIB system

Figure 7 illustrates the response of ΔP_{elec} to a unit step in ΔV_{ref} for the SMIB system using the CPSS, optimal H_∞ PSS and sub-optimal H_∞ PSS. From Figure 7, it can be seen that the open loop step response is weakly damped with the oscillation persisting after 10 seconds. The closed loop systems with the CPSS, optimal H_∞ PSS and sub-optimal H_∞ PSS have significantly improved damping of the ΔP_{elec} oscillation. For the closed loop system with conventional PSS, the oscillation in ΔP_{elec} is settled within 3 seconds. For the closed loop system with sub-optimal H_∞ PSS, the oscillation in ΔP_{elec} is settled within 2.5 seconds. For the closed loop system with optimal H_∞ PSS, the oscillation in ΔP_{elec} is settled within 1 second.

2.4 Conclusion

In this chapter, a new method for design/tuning PSS was presented, namely sub-optimal H_∞ PSS design/tuning method. The method is based on formulating the PSS design/tuning problem as a numerical optimization problem. The objective function is taken as the H_∞ norm of the closed loop disturbance-related transfer function. The constraints were selected as the minimum damping factor of the overall system and the limitations of parameters of PSS. A case study was used to prove the following:

- Sub-optimal H_∞ PSS has better robustness than that of the conventional PSS.
- Sub-optimal H_∞ PSS design/tuning method gives the freedom of selecting PSS structure without requiring model reduction and optimizes the use of existing PSS.
- The closed loop system with sub-optimal H_∞ PSS increases the overall damping of the "problem" mode.
- SQP is not always an appropriate optimization procedure for problem formulations that have more than one local optimum.

2.5 References

[1] G. Rogers, "Power System Oscillations", Kluwer Academic Publishers, 2000

[2] P. Kundur, "Power System Stability and Control", McGraw-Hill, New York, 1994

[3] P.W. Sauer and M.A. Pai, "Power System Dynamics and Control", Prentice Hall, New Jersey, 1997

[4] MATLAB Optimization Toolbox Manual, The Mathworks, Natick, 1998

[5] K. Dutton, et al "The Art of Control Engineering", Addison-Wesley, 1997

[6] B. Friedland, "Control System Design", McGraw-Hill , 1987

[7] N. Martins, L. Lima, "Eigenvalue and Frequency Domain Analysis of Small-signal Electromechanical Stability Problems", IEEE Publication, Eigenanalysis and Frequency Domain Methods for System Dynamic Stability (90TH0292-3-PWR) 1989, pp17-33

[8] M. Green, D. Limebeer "Linear Robust Control", Prentice Hall, 1995

[9] MATLAB Robust Control Toolbox Manual, The Mathworks, Natick, 1998

[10] C. Fransson, et. al., "Global Optimization Using Horowitz Bounds", IFAC 15th Triannual World Congress, Barcelona Spain, 2002

[11] B.A. Foss and T.A. Johansen, "Identification and Convexity in Optimizing Control", Article from Internet Search

Chapter 3

PBIL-based Sub-optimal H_∞ Controller Design

Chapter 2 focused on the development of sub-optimal H_∞ PSS design/tuning method. The feasibility of the design/tuning method for PSS was proven through a simple case study. The case study compared different PSS that were designed using SQP-based sub-optimal H_∞ PSS design/tuning method, optimal H_∞ PSS design method and conventional PSS design method.

In Chapter 2, it was shown that the optimization method, Sequential Quadratic Programming (SQP), is not well suited for finding a global solution [1]. SQP is only guaranteed to find a local-optimum; it relies heavily on a good initial guess and was developed for dealing with convex optimization problems. Multi-machine power systems have more than one "problem" mode, and the sub-optimal H_∞ optimization problem is not guaranteed to be convex. Therefore, the optimization procedure used in the sub-optimal H_∞ PSS design/tuning method needs to be able to find global optimal solutions and handle non-convex optimization problems. Obviously, SQP is not a practical optimization method for multi-machine system PSS design and tuning.

In this chapter, an optimization method called Population-Based-Incremental-Learning (PBIL) will be introduced. PBIL was developed to find solutions for non-convex global optimization problems. In this chapter, equation 2.8, with the constraints defined by equation 2.9 will be formulated as a PBIL optimization problem. The intention of this section is purely to compare PBIL and optimal H_∞ in terms of output performance. It will be shown that:

- The PBIL formulation can be used to obtain similar controllers to those obtained using standard optimal H_∞ design techniques.
- The PBIL formulation can be used to design high-order complex controllers.
- The PBIL formulation can be used to give the designer more flexibility in the selection of the characteristics that the required controller should have.

3.1 Brief Introduction of PBIL

Baluja [2] originally proposed Population-Based Incremental Learning (PBIL) method. PBIL is a combination of Genetic Algorithms (GA) and Competitive Learning (CL). The objective of the PBIL method is to create a real valued probability vector, which when sampled, reveals high quality solution vectors. The solution vectors have high probability with respect to the available knowledge of the search space.

The probability vector is updated at each generation. Each element in the probability vector is updated as follows:

$$P_I(k) = (1 - LR) * P_I(k-1) + LR * \max_I(k-1) \qquad 3.1$$

Where:

$P_I(k)$	The probability of generating a "1" at the I-th bit position at generation k.
$\max_I(k)$	The I-th position in the solution vector which the probability vector is moved towards at generation k.
LR	The learning Rate

The present probability vector points towards the better individuals of the population and away from the worst individuals. Each bit or component of the probability vector is analyzed independently. While the search progresses, entries in the probability vector move away from their initial settings of 0.5 toward either 0 or 1. As the probabilities become very close to 0.0 or 1.0, the similarity in the solution vectors increases.

The four parameters in the PBIL method that must be defined for each problem are the size of the population, the learning rate (LR), the probability of mutation (MP) and the relationship of change (MS), i.e. the effect of the mutation on the probability vector.

The pseudo-code for the PBIL algorithm is given in Figure 8. For a detailed overview on PBIL, see Appendix C.

```
P ← initialize probability vector to 0.5
 Loop  # Generations
{
    //Generate Samples
    j←  loop # Samples
    {
      sample_j ← generate sample vector according to probabilities in P
      eval_j ← evaluate(sample_j)
    }

    //Find best sample
    max ← find vector corresponding to maximum evaluation.

   //Update Probability vector
   I← loop # Length
   {
        P_I ← P_I*(1.0-LR)+max_I*LR
   }

   //Mutate Probability vector
   I← loop # Length
   {
        if(random (0,1] < MP)
            P_I ← P_I*(1.0-MS)+random(0 or 1)*MS
    }
}
```

Generations	Number of iterations to allow learning
Samples	Population size, number of samples to produce per generation
Length	Length of encoded solution
MP	Probability of mutation occurring in each position.
MS	Amount for mutation to affect the probability vector
LR	Learning rate
P:	Probability vector
$Sample_{j..samples}$	Solution vectors
$Eval_{j..samples}$	Evaluations of the solution vectors
max	Solution vector corresponding to maximum evaluation

Figure 8 PBIL algorithm

3.2 Control Problem and Design Objectives

To validate that the PBIL-based sub-optimal[1] H_∞ design/tuning procedure can be used to design controllers that are similar in structure and performance to controllers designed using optimal H_∞, the method has been applied to a MATLAB reference example. The decision to use a MATLAB H_∞ reference example, rather than a power system model, is to remove any bias in the comparison, thereby providing a high confidence factor in the validity of the PBIL-based sub-optimal H_∞ design/tuning procedure.

This MATLAB H_∞ reference example is a single input single output (SISO) system, $G(s)$, contains both additive and multiplicative uncertainties. This implies that the mixed-sensitivity approach is required to design the desired controller. The mixed-sensitivity approach requires performance, error and robust weighting functions [3].

3.2.1 Plant Characteristics

The system model under investigation is a standard MATLAB SISO system model [4]. The open loop plant is

$$G(s) = \frac{7.56s^3 + 51.08s^2 + 142.8s + 97.22}{s^4 + 5.62s^3 + 46.34s^2 + 145.29s + 97.22}$$

[1] The definition is based on the fact that we specifically chose the order of the controller. Clearly the choice of structure results in a global solution. When comparing the controllers in this section, the "sub-optimal" controller performs as well as the H_∞ controller, and therefore could be said to be optimal. However, given that we chose an existing structure, the controller is sub-optimal.

3.2.1.1 Bode Diagram

Figure 9 shows the bode diagram for the open loop system under investigation. The system is a second order system with a dominant mode at 6.09 rad/s. The system has an infinite gain margin and a phase margin of 87.3 degrees at 10.76 rad/s.

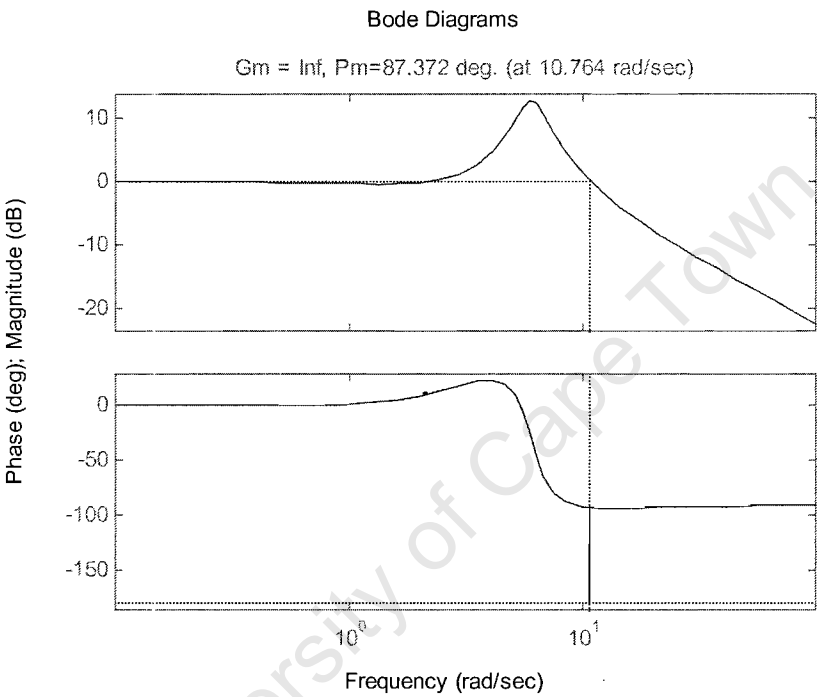

Figure 9 Open loop bode diagram showing gain and phase margins

3.2.1.2 Nyquist Diagram

From Nyquist diagram, as shown in Figure 10, it can be seen that the open system is stable, since it does not enclose -1+0j point.

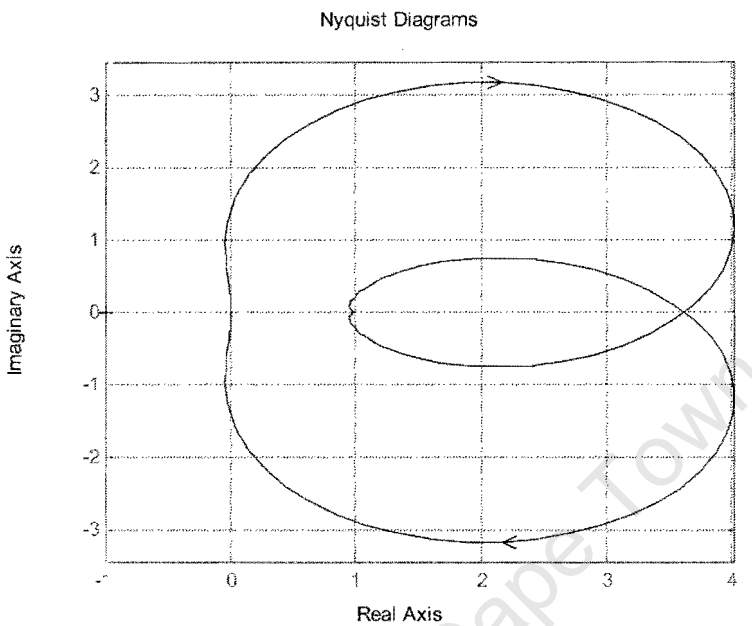

Figure 10 Open loop system - Nyquist diagram

3.2.1.3 Poles and Damping Factors

Table 5 lists the open-loop poles, their associated damping and natural frequency. The system is stable as all the poles are on the left hand side of the s-plane.

Open Loop Poles		
Eigenvalue (Pole)	**Damping Factor**	**Frequency (rad/s)**
-0.904 ± j6.0	0.149	6.070
-0.908	1.000	0.908
-2.910	1.000	2.910

Table 5 Open loop poles and damping

3.2.2 Design Objectives

The design objectives for optimal H_∞ are expressed in terms of weighting functions [3]. The weighting functions are selected based on the required disturbance attenuation and the maximum allowable multiplicative and additive noise.

3.2.2.1 Performance Weighting Function

The performance weighting function determines the desired disturbance attenuation at each frequency. The performance weighting function is given as:

$$w_1^{-1}(s) = \frac{1.5}{100}\left[\frac{s^2 + 2s + 1}{\frac{1}{900}s^2 + \frac{2}{30}s + 1}\right]$$

3.2.2.2 Input Weighting Function

The input weighting function defines the largest expected additive noise that can be expected at each frequency. The input weighting function is given as:

$$w_2(s) = \frac{\frac{1}{100}s + \frac{1}{10}}{s + \frac{1}{10}}$$

3.2.2.3 Robust Weighting Function

The robust weighting function defines the maximum expected multiplicative noise that can be expected at each frequency. The robust weighting function is given as:

$$w_3^{-1}(s) = 3.16\left[\frac{\frac{1}{300}s + 1}{\frac{1}{10}s + 1}\right]$$

3.3 Optimal H_∞ Controller Design

The optimal H_∞ controller, designed by using Matlab's optimal H_∞ functions, used the weighting functions described earlier. See Appendix E for results obtained from Matlab.

3.3.1 Optimal H_∞ Controller Zeros and Poles

The zeros of the desired optimal H_∞ controller are:

-300	-8.5904
-0.90384 + j5.9981	-0.90384 - j5.9981
-2.9113	-0.1
-0.90767	

The poles of the desired optimal H_∞ controller are:

-7190	-57.164
-2.8942 + j2.2143	-2.8942 – j2.2143
-1	-0.99999
-0.96730	-0.00324

3.4 PBIL-based Sub-optimal H_∞ Controller Design

The PBIL-based sub-optimal H_∞ controller was designed using the approach outlined in Chapter 2, except for an additional weighting parameter (γ) on $W_1(s)$. The inclusion of γ was to provide a basis for comparison to the optimal H_∞ design procedure. We adjusted γ for each trial. The weighting functions used, are the same as those used for the optimal H_∞ controller design. The following parameters and constraints were defined for the optimization procedure:

Learning Rate	0.10
Mutation Probability	.02
Number of trials/generation	60
Maximum number generations	150
Damping Range (x)	min(plant damping) < x
Gamma (γ)	$0.5 < \gamma < 1.5$

Table 6 PBIL parameters

The PBIL-based sub-optimal H_∞ controller structure was chosen to have the same order as the optimal H_∞ controller. In addition the zero and pole structures were specified as opposed to representing the controller as a fraction of two polynomials. There are a number of benefits in being more specific on how the poles and zeros should appear in the final design. These include:

- the ability to control the stability of the controller
- the ability to define the desired number of poles and zeros and in addition whether the zeros and poles are real or complex.

The controller, *Q(s)*, had the following structure:

$$Q_1(s) = \frac{(s+\alpha_{z1})(s+\alpha_{z2})(s+\alpha_{z3})(s+\alpha_{z4})(s+\alpha_{z5})}{(s+\alpha_{p1})(s+\alpha_{p2})(s+\alpha_{p3})(s+\alpha_{p4})(s+\alpha_{p5})}$$

$$Q(s) = KQ_1(s)\frac{(s+\alpha_{z6}+j\beta_{z6})(s+\alpha_{z6}-j\beta_{z6})}{(s+\alpha_{p6})(s+\alpha_{p7}+j\beta_{p7})(s+\alpha_{p7}-j\beta_{p7})}$$

The objective function was defined as:

Minimize the maximum singular value, where the minimum damping is no less than the system's minimum damping factor.

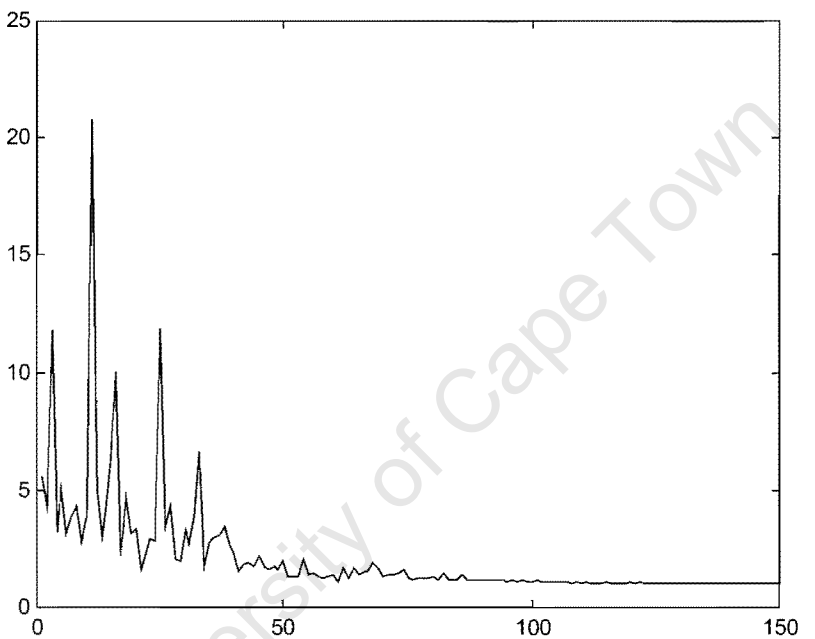

Figure 11 Convergence of singular value

Figure 11 shows the convergence of the singular value. Convergence was reached after 100 generations. The optimal Gamma is: 0.97445.

3.4.1 PBIL-based Sub-optimal H_∞ Controller Zeros and Poles

The zeros of the desired PBIL-based sub-optimal H_∞ controller are:

-1.8853 + j8.1397	-1.8853 - j8.1397
-40.1112	-200.4022
-40.2453	-200.6888
-39.3	

The poles of the desired PBIL-based sub-optimal H_∞ controller are:

-10.9758 + j0.74763	-10.9758 -j0.74763
-5.0384	-0.37059
-150.03	-142.43
-169.47	-236.96

The controller gain (K) is 496.83

3.5 Comparison and Discussion of Results

3.5.1 Comparison between Optimal H_∞ and PBIL-based Sub-optimal H_∞ Controllers

Figure 12 shows the comparison of the frequency responses of the optimal H_∞ and PBIL-based sub-optimal H_∞ controllers. Between 0.1 and 100 rad/s the overall frequency responses are very similar.

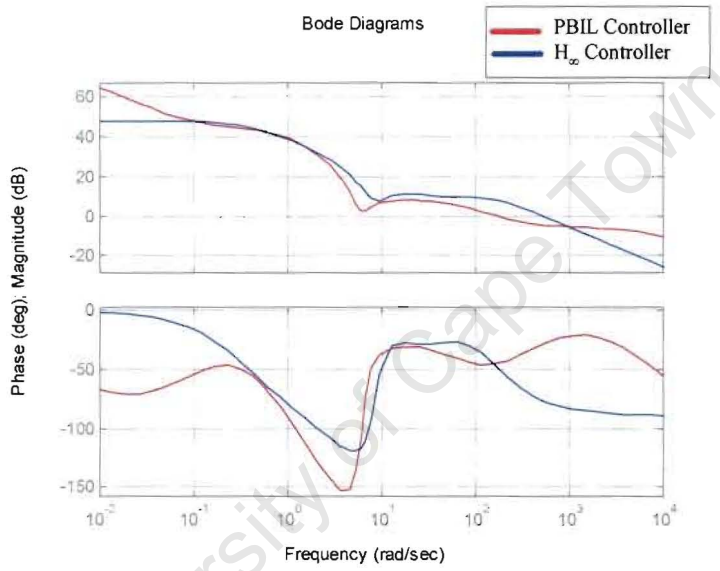

Figure 12 Optimal H_∞ controller and PBIL-based sub-optimal H_∞ controller bode diagrams

Table 7 and Table 8 show the eigenvalues, the damping and the natural frequency for each closed pole with the optimal H_∞ controller and the PBIL-based sub-optimal H_∞ controller applied. The minimum damping of the closed loop system is the same as the open loop system.

Table 7 shows that the "problem" mode at frequency of 6.07 rad/s, with eigenvalue as $-0.904 \pm j6.00$, and the damping factor 0.149, is the same as it shown in Table 5. This again highlights how an optimal H_∞ controller does not affect the damping of the dominant mode.

In Table 8, the closed loop system, with the PBIL-based sub-optimal H_∞ controller applied, has an additional pair of complex poles and zeros. Both of the complex poles and zeros are located far from the imaginary axis and therefore have little impact on the overall system response.

Closed Loop Poles (with Optimal H_∞ Controller)		
Eigenvalue (Pole)	**Damping Factor**	**Frequency (rad/s)**
-0.0995	1.000	9.95e-002
-0.904 ± j6.00	0.149	6.07e+000
-0.908	1.000	9.08e-001
-0.968	1.000	9.68e-001
-2.89 ± j2.22	0.794	3.64e+000
-2.91	1.000	2.91e+000
-10.48 ± j8.77	0.860	1.72e+001
-30.32	1.000	3.32e+001
-7019	1.00	7.19e+003

Table 7 Closed loop poles, damping and natural frequency (Optimal H_∞ controller)

Closed Loop Poles (with PBIL-based Sub-optimal H_∞ Controller)		
Eigenvalue (Pole)	**Damping Factor**	**Frequency (rad/s)**
-0.968	1.000	9.68e-001
-1.28e ± j8.46	0.150	8.56e+000
-2.81e ± j2.21	0.786	3.58e+000
-10.33 ± j10.78	0.599	2.22e+001
-20.61	1.000	2.61e+001
-100.24 ± j40.96	0.928	1.33e+002
-200.04	1.000	2.04e+002
-200.37	1.000	2.37e+002

Table 8 Closed loop poles, damping and natural frequency (PBIL-based sub-optimal H_∞ controller)

3.5.3 Bode Diagram

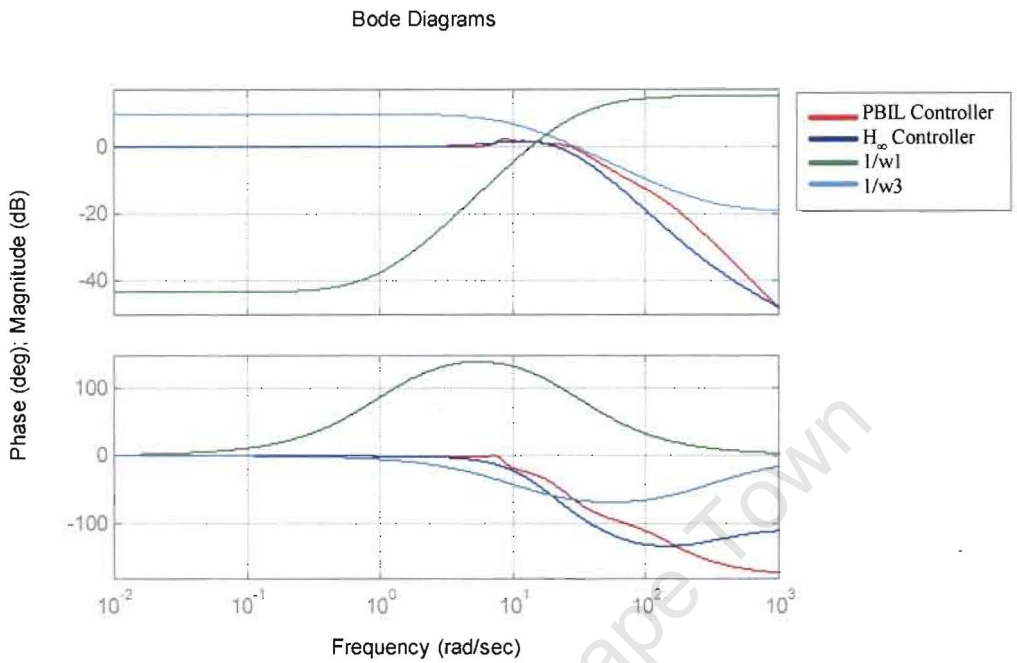

Figure 13 Bode diagram

Figure 13 shows the bode diagrams of the closed loop system with the optimal H_∞ and the PBIL-based sub-optimal H_∞ controllers applied, and the weighting functions $1/w_1(s)$ and $1/w_3(s)$. From Figure 13 it can be seen that both controllers meet the performance and robust design specification.

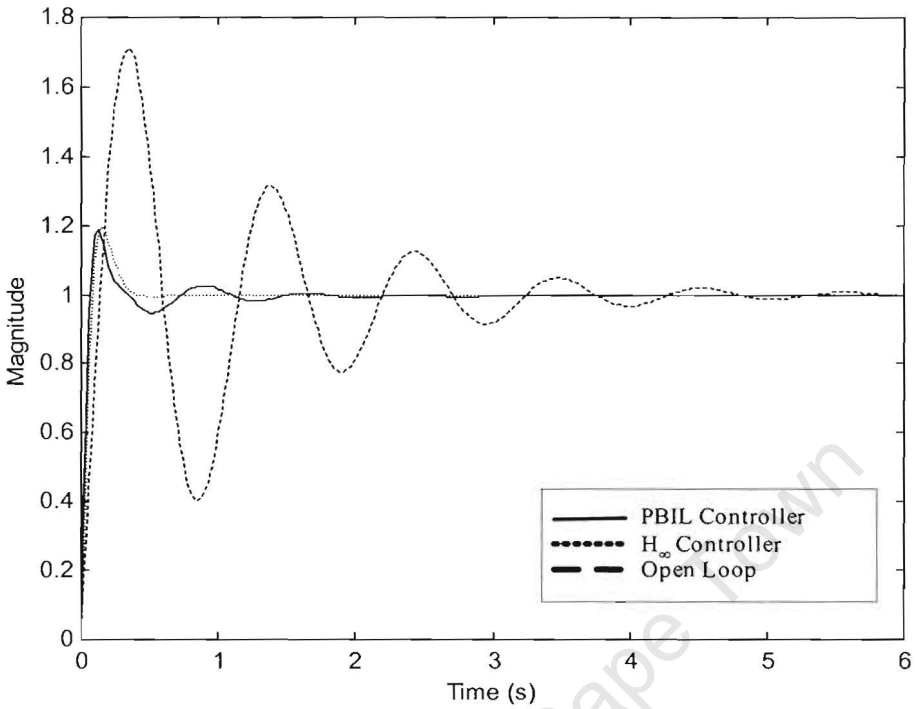

Figure 14 Time response

Figure 14 shows the step responses of the open loop system and the closed loop system with the optimal H_∞ and PBIL-based sub-optimal H_∞ controllers applied. Both controllers damp the overall oscillation.

3.5.5 Singular Values

Figure 15 is a plot of the singular values of the closed loop system with the optimal H_∞ and PBIL-based sub-optimal H_∞ controllers applied. The maximum singular value of the closed loop system with the PBIL-based sub-optimal H_∞ controller applied is 1.3359, which is approximately 8% higher than the maximum singular value of the closed loop system with the optimal H_∞ controller applied. The optimal H_∞ controller is marginally more robust than the PBIL-based sub-optimal H_∞ controller. The PBIL design/tuning procedure guarantees a near-optimal solution.

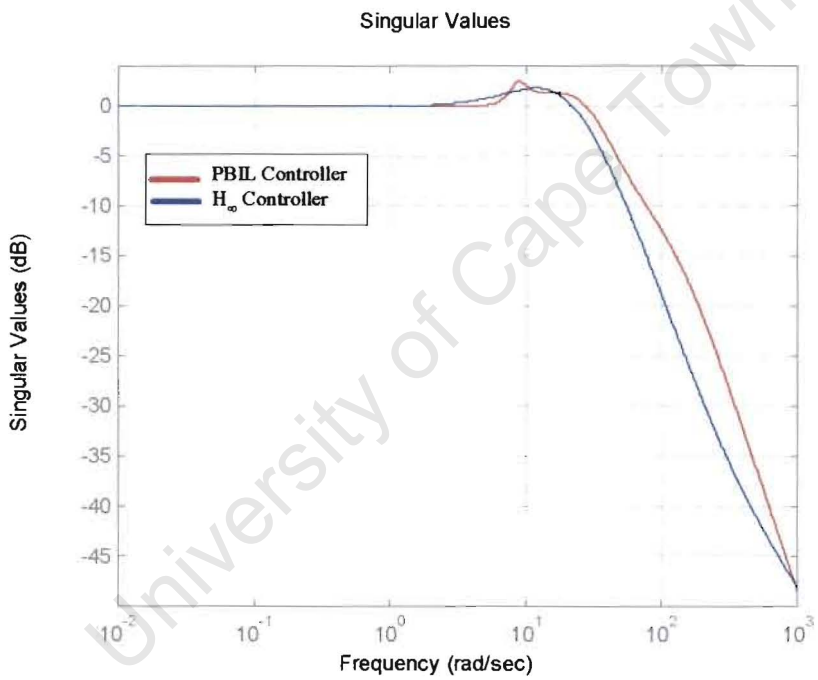

Figure 15 Singular value plot

Gain and Phase margins [5-6] are typical measures of the stability and robustness of SISO systems. Figure 16 and Figure 17 show the bode diagrams of the closed loop system with the PBIL-based sub-optimal H_∞ controller and optimal H_∞ controller applied. Table 9 gives the gain and phase margins of the closed loop system. From the figures and table, it can be seen that PBIL-based sub-optimal H_∞ controller works as effectively as optimal H_∞ controller.

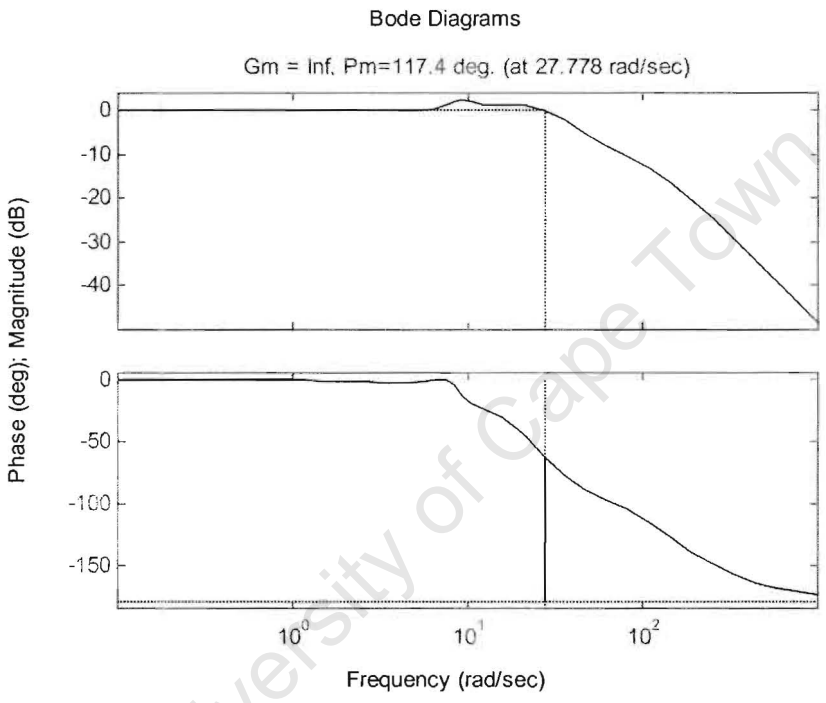

Figure 16 Gain/phase margin of closed loop system with PBIL-based sub-optimal H_∞ controller

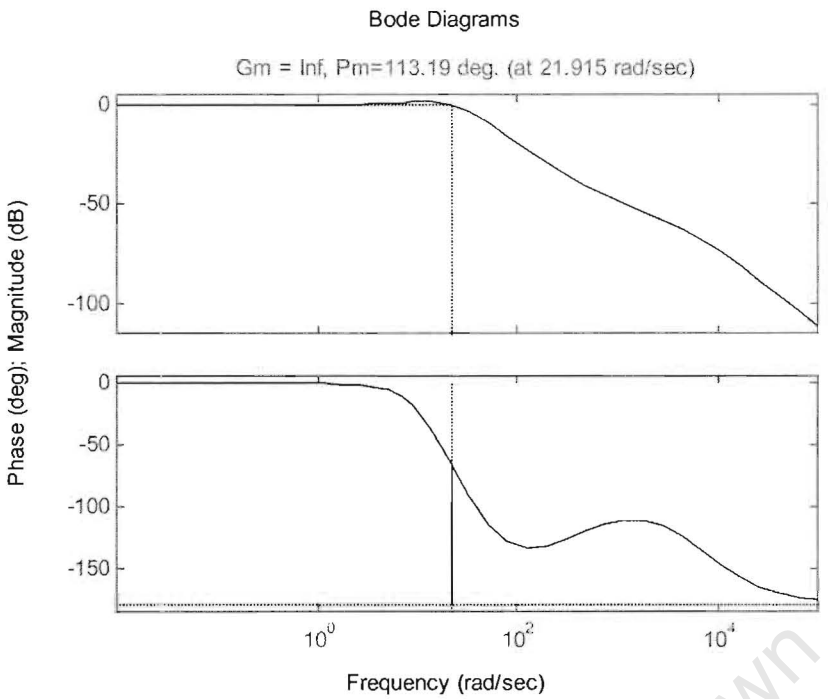

Figure 17 Gain/phase margin of closed loop system with optimal H_∞ controller

	PBIL-based Sub-optimal H_∞ PSS	**Optimal H_∞ PSS**
Gain Margin	Infinite	Infinite
Phase Margin	117.4	113.19
ω_{gm}	n/a	n/a
ω_{pm}	27.778 rad/s	21.915 rad/s

Table 9 Gain and phase margins of closed loop system

3.5.6.1 Nyquist Diagram

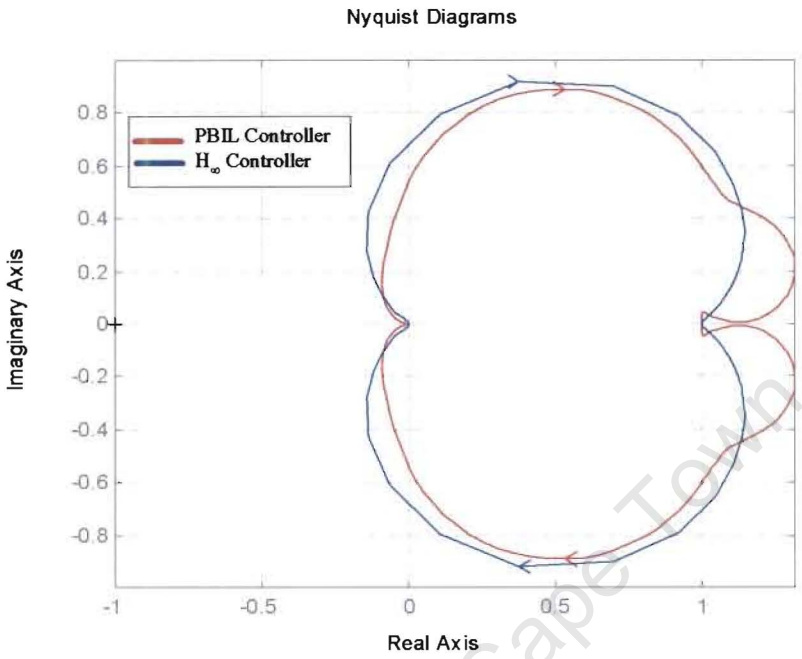

Figure 18 Nyquist diagram

Figure 18 shows the Nyquist diagrams of closed loop system with the PBIL-based sub-optimal H_∞ controller and optimal H_∞ controller. Since the Nyquist diagram does not intersect the negative real axis, the gain margin is infinite [5]. This was shown earlier using the Magnitude and Phase plots of the closed loop systems. So we can comment that the closed loop system with PBIL-based sub-optimal H_∞ controller or optimal H_∞ controller is insensitive to the changes in the system and therefore robust.

3.6 Conclusion

In this chapter, a PBIL-based sub-optimal H_∞ controller design method has been introduced. The comparison to optimal H_∞ controller has been done through a SISO system case study. The following was shown that:

- Using a PBIL formulation of the sub-optimal H_∞ design/tuning procedure, it is possible to obtain controllers, which compare well to standard optimal H_∞ controllers.
- The PBIL formulation presented in this thesis can be used to design high-order complex controllers.
- The PBIL formulation gives the designer more flexibility in the selection of the characteristics that the required controller should have.

3.7 References

[1] C. Fransson, et. al., "Global Optimization Using Horowitz Bounds", IFAC 15th Triannual World Congress, Barcelona Spain, 2002

[2] S. Baluja, "Population Based Incremental Learning" CMU-CS-94-163

[3] M. Green , D. Limebeer, "Linear Robust Control", Prentice-Hall Inc., 1997

[4] MATLAB Robust Control Toolbox, The Mathworks, Natick, 1998

[5] K. Dutton, S. Thompson, B. Barraclough, "The Art of Control Engineering", Addison-Wesley, 1998

[6] B. Friedland, "Control System Design – An Introduction to State-Space Methods", McGraw-Hill, 1987

Chapter 4

Application of PBIL-based Sub-optimal H_∞ PSS Design/Tuning

In chapter 3, the focus was on the introduction of PBIL-based sub-optimal H_∞ design/tuning method. In chapter 3 it was shown that the PBIL-based sub-optimal H_∞ design/tuning method can be used to design controllers that compare well to those designed using optimal H_∞ control design method.

In previous studies [2-8], SQP-based sub-optimal H_∞ methodology was used to design/tune PSS. As described in chapter 2, SQP is not well suited for finding global optimum solution, nor is it suited to solving non-convex optimization problems. In this chapter, the focus will be on the application of PBIL-based sub-optimal H_∞ PSS design/tuning procedure. To demonstrate the effectiveness of the PBIL-based sub-optimal H_∞ PSS design/tuning method, two examples will be given, namely the design of PSS for a three-machine-nine-bus system, using constrained and unconstrained PBIL-based sub-optimal H_∞ PSS design/tuning method. In the case studies, conventional PSS and SQP-based sub-optimal H_∞ PSS will be applied to compare with the PBIL-based sub-optimal H_∞ PSS.

4.1 System under Investigation

The system under investigation is a 3-machine-9-bus power system as shown below [9]. The system was modeled using PSS/E. All the state-space matrices are listed in Appendix F.

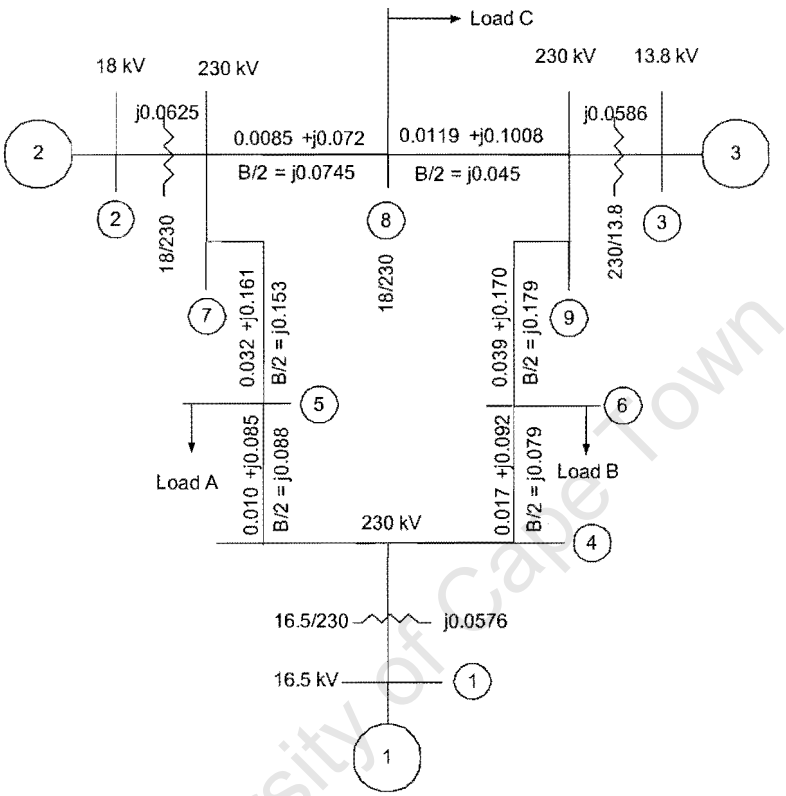

Figure 19 Impedance diagram of 3 machine 9 bus system

A PSS was selected to be second order in structure and using Generator 1's electrical power deviation as input and reference voltage deviation as output. Generator 1 was selected to be the location of PSS, since the dominant mode comes from generator 1. Participation factors analysis [10-12] was used to determine the PSS location. The participation factors are calculated using the right and left eigenvalues corresponding to the electromechanical mode.

4.2 Conventional PSS

The conventional PSS was designed for comparison using the method described in Appendix A. It is a second order PSS and has the following transfer function:

$$K_{cpss}(s) = \frac{0.5(s+7.69)^2}{(s+4.35)^2} \qquad 4.1$$

4.3 SQP-based Sub-optimal H_∞ PSS

The SQP-based sub-optimal H_∞ PSS was designed for comparison using the method described in Chapter 2. The PSS is selected to be a second order as well for the comparison to conventional and PBIL-based sub-optimal H_∞ PSS. This PSS has the following transfer function:

$$K_{sqp}(s) = \frac{0.757(s^2+1.113s+1.882)}{(s^2+1.988s+2.171)} \qquad 4.2$$

4.4 PBIL-based Sub-optimal H_∞ PSS (Constrained Optimization)

The optimization criterion is as follows: *Minimize the H_∞-norm as objective function and maximize the minimum damping as constraint.*

Table 10 gives the basic parameters for the PBIL-based sub-optima H_∞ PSS design/tuning. The MATLAB scripts are located in Appendix F.

Variable	Value	Description
L	0.1	Learning rate
MP	0.02	Mutation probability
NVARS	5	Number of variables (controller parameters)
TRIALS	30	Number of trials per generation
MAXGEN	100	Number of generations
PREC	22	Precision of variables

Table 10 Basic parameters for the PBIL-based sub-optimal H_∞ PSS design/tuning (with constraints)

4.4.1 Convergence Diagrams

Figure 20 and Figure 21 show the convergence of the H_∞ -norm and damping. Analyzing the convergence diagrams, the optimal solution was found after 70 generations. Either by changing the number of iterations (children) per generation, or by changing the learning rate could have accelerated the process of finding the solution.

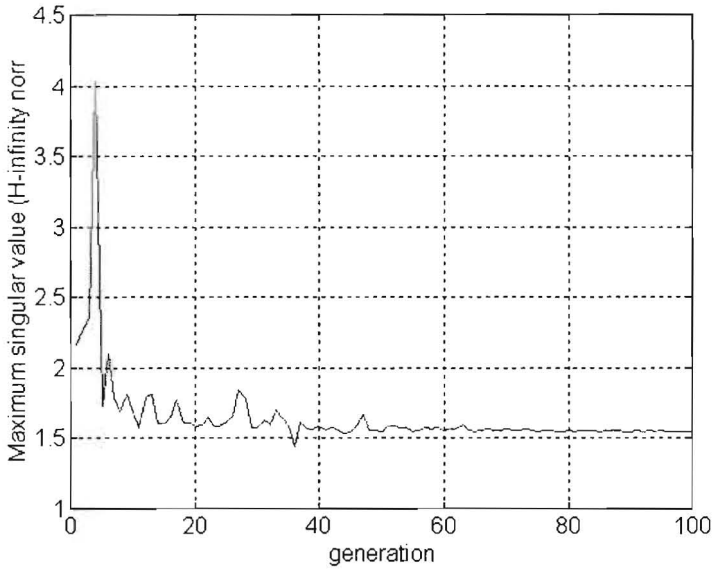

Figure 20 Convergence of H_∞ norm

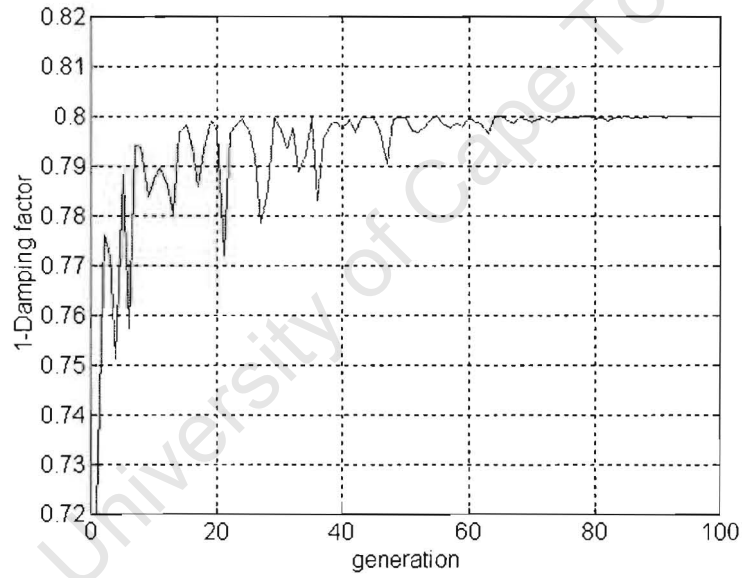

Figure 21 Convergence of damping (Shown as (1-damping) for clarity)

Figure 21 shows the convergence of damping, which was plotted as "1-damping factor". The open loop "problem" modes correspond to damping factors of 0.093 and 0.098 (see Table 11). Figure 21's starting point corresponds to damping factor of 0.38, the result converged at the damping factor of 0.2. This means that the best solution from first iteration would give a minimum damping factor at 0.38, but the simulation only converged after 60 iterations and gave the damping factor at 0.2. This is consistent with the results shown in Table 11.

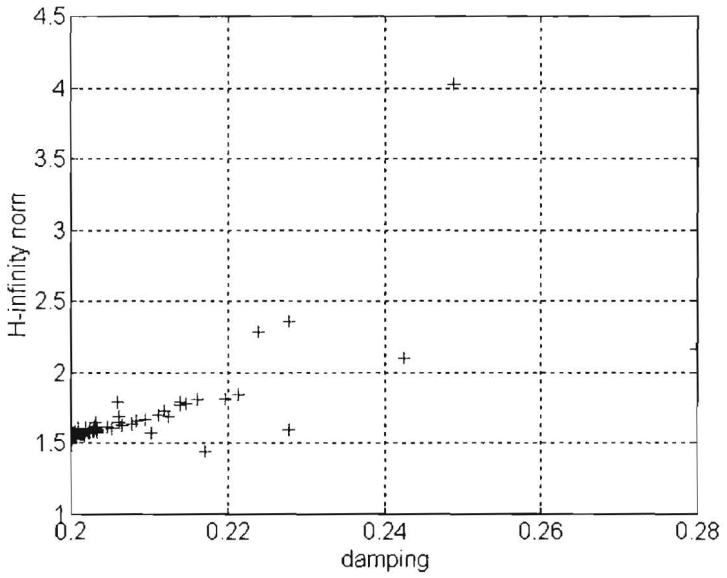

Figure 22 Plot of scatter of H_∞ norm versus damping

Figure 22 shows that the majority of solutions are located in the region with H_∞ norm of 1.5 and with damping factor of 0.2. This figure matches the convergence Figure 20 and Figure 21. One solution appears to be better than all of the others (with H_∞ norm of 4.1 and damping factor of 0.25), but this is an isolated case, and is therefore not an acceptable solution. Only converged results are the acceptable results.

The resultant PBIL-based sub-optimal H_∞ PSS has the following transfer function:

$$K_{pbil}(s) = \frac{0.53(s+0.128)(s+0.103)}{(s+0.108)(s+0.059)} \qquad 4.3$$

4.4.2 Comparison

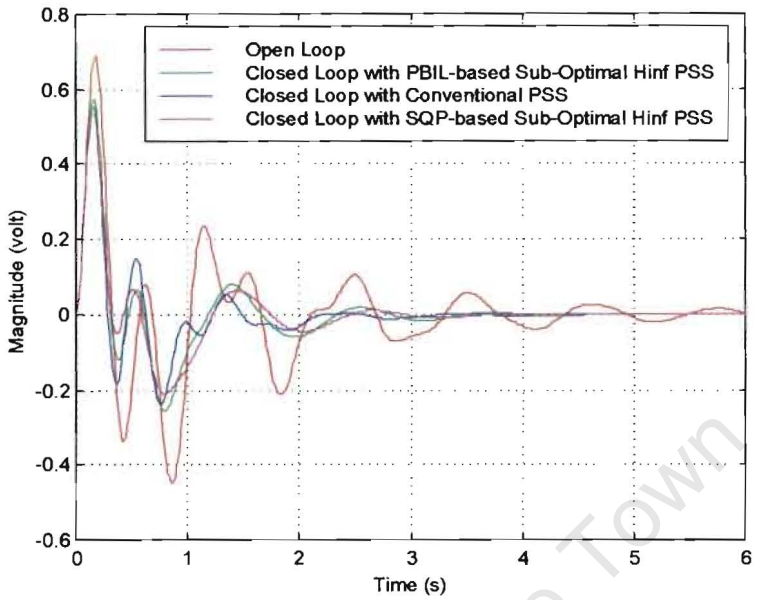

Figure 23: Step response

From the step responses shown in the Figure 23, all three PSS improve the response of the system significantly, as compared to the open loop step response.

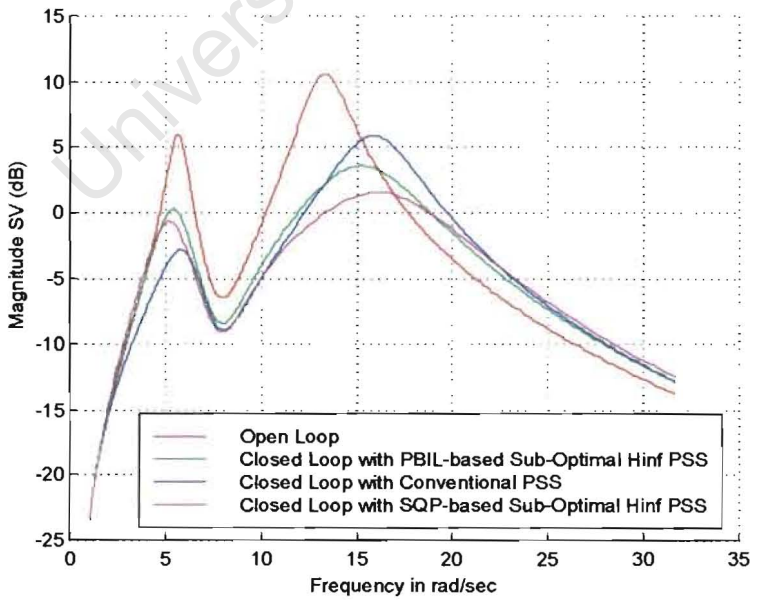

Figure 24 Bode plot

From Figure 24, the SQP-based sub-optimal H_∞ PSS damps the mode mainly at the frequency of 13.2998 rad/s, while the conventional PSS damps the mode mainly at the frequency of 5.6926 rad/s (this is because the conventional PSS was designed at 5.6926 rad/s peak). The PBIL-based sub-optimal H_∞ PSS gives an average response on both modes.

In this case study, SQP-based sub-optimal H_∞ PSS performs slightly better than PBIL-based sub-optimal H_∞ PSS. The "problem" modes for open loop system have been improved. The damping factors for open loop system at 0.0937 and 0.0989 have been increased to 0.2669 and 0.2502 by SQP-based sub-optimal H_∞ PSS, and to 0.2207 and 0.2029 by PBIL-based sub-optimal H_∞ PSS. While running the simulations, SQP optimization depended on the starting points to a great extent. For obtaining the results shown in this thesis, more than 100 starting points had been tested. The initial guessing included a set of parameters for the PSS and the maximum damping factor for the weakly damped modes. For using SQP-based sub-optimal H_∞ PSS design method, experiences are required to take a close guess for start, otherwise, this method can be time consuming and it doesn't ensure the best solution. Although constrained PBIL-based sub-optimal H_∞ PSS, in this case study, gave slightly poorer performance than SQP-based sub-optimal H_∞ PSS, the design procedure is much simpler and easier to implement than SQP-based sub-optimal H_∞ PSS design. Constrained PBIL-based sub-optimal H_∞ PSS design method is practical for the utility to tune their existing PSS, it also allows one to set all the limits for PSS parameters.

Note that the parameters for PBIL-based sub-optimal H_∞ PSS and conventional PSS are real values, while the SQP-based sub-optimal H_∞ PSS are complex values. One can design PSS parameters with complex values for those digital PSS. By specifying the PSS parameter formats, PBIL-based sub-optimal H_∞ design/tuning method can obtain PSS parameters with complex values as well.

Table 11 gives the complex eigenvalues and associated damping factors:

Controller	Eigenvalue (Pole)	Damping Factor	Frequency (rad/s)
PBIL-based Sub-optimal H_∞ PSS	-1.1277 ± 5.4415i	0.2029	5.5571
	-5.7451 ± 5.9849i	0.6926	8.2961
	-3.4268 ±15.1448i	0.2207	15.5276
SQP-based Sub-optimal H_∞ PSS	-1.0571 ± 1.0546i	0.7079	1.4932
	-1.3315 ± 5.1520i	0.2502	5.3212
	-4.7594 ± 6.6581i	0.5815	8.1842
	-4.5145 ±16.2980i	0.2669	16.9117
Conventional PSS	-5.7899 ± 0.4157i	0.9974	5.8048
	-1.3606 ± 6.0446i	0.2196	6.1959
	-5.5465 ± 4.1454i	0.8010	6.9244
	-2.3581 ±15.8054i	0.1476	15.9803
Open Loop	-0.5632 ± 5.6647i	0.0989	5.6926
	-6.5914 ± 4.3424i	0.8351	7.8932
	-1.2461 ±13.2413i	0.0937	13.2998

Table 11 Poles and damping factors of closed loop system with various PSS and open loop system

The gain and phase margins give a good indication of the robustness of the system [12]. The gain and phase margins are given below:

	PBIL-based	Conventional	SQP-based
Gain Margin	17	19	16
Phase Margin	75	63	88
ω_{gm}	54	56	52
ω_{pm}	18	19	18

Table 12 Comparison of gain and phase margin for various PSS

In terms of gain margin, the conventional PSS is better, but in terms of phase margin the PBIL-based sub-optimal H_∞ PSS and SQP-based sub-optimal H_∞ PSS are far superior. The values presented in Table 12 can be calculated from a Bode or Nyquist plot [12].

Figure 25 gives the Nyquist plot. SQP-based sub-optimal H_∞ PSS gives the best performance. PBIL-based sub-optimal H_∞ PSS performs better than conventional PSS. Note that the closer the plot is to the (-1,0) point the more unstable the system (i.e. less phase margin) is.

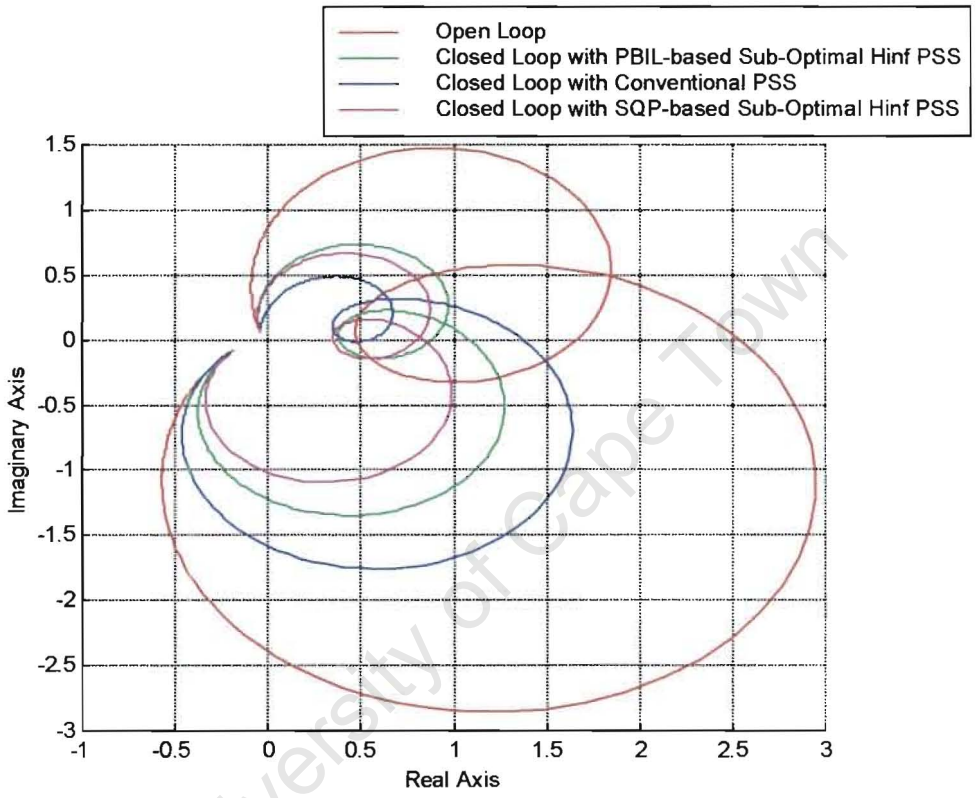

Figure 25 Nyquist plot

4.5 PBIL-based Sub-optimal H_∞ PSS (Unconstrained Optimization)

The optimization criterion was as follows: *Minimize the H_∞-norm such that it is greater than or equal to one.*

No specific constraints were placed on the norm or the damping. The table below gives the basic parameters for the PBIL-based sub-optima H_∞ PSS design/tuning. The MATLAB scripts are located in Appendix G.

Variable	Value	Description
L	0.1	Learning rate
MP	0.02	Mutation probability
MS	0.05	Mutation shift
NVARS	5	Number of variables (controller parameters)
TRIALS	30	Number of trials per generation
MAXGEN	100	Number of generations
PREC	22	Precision of variables

Table 13 Basic parameters for the PBIL-based sub-optimal H_∞ PSS design/tuning (no constraints)

4.5.1 Convergence Diagrams

Figure 26 through Figure 28 show the convergence of the H_∞ norm and damping. Analyzing the convergence diagrams, the optimal solution was found after 80 generations. The process of finding the solution can be accelerated by either changing the number of trials (children) per generation or changing the learning rate.

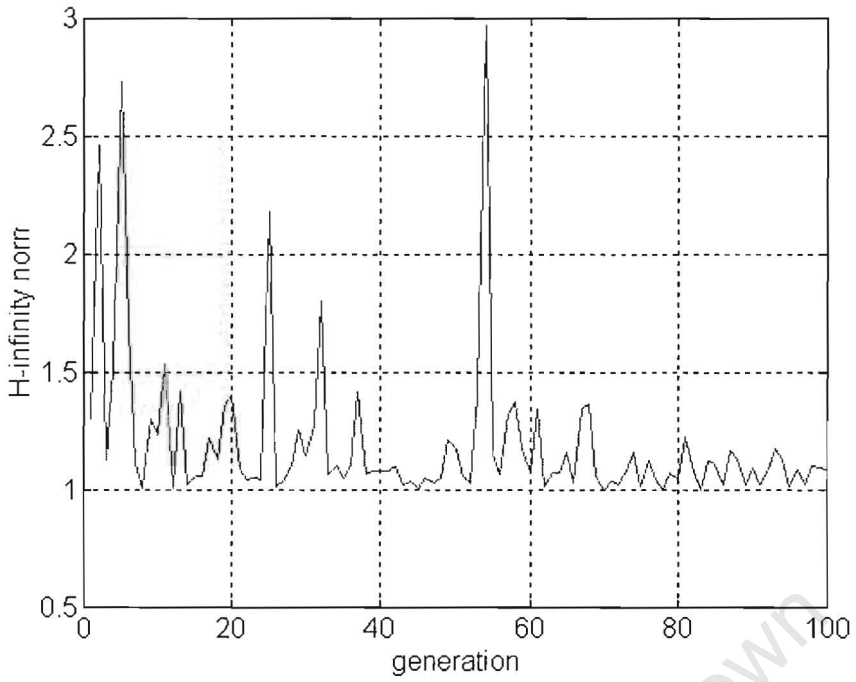

Figure 26 Convergence of H_∞ norm

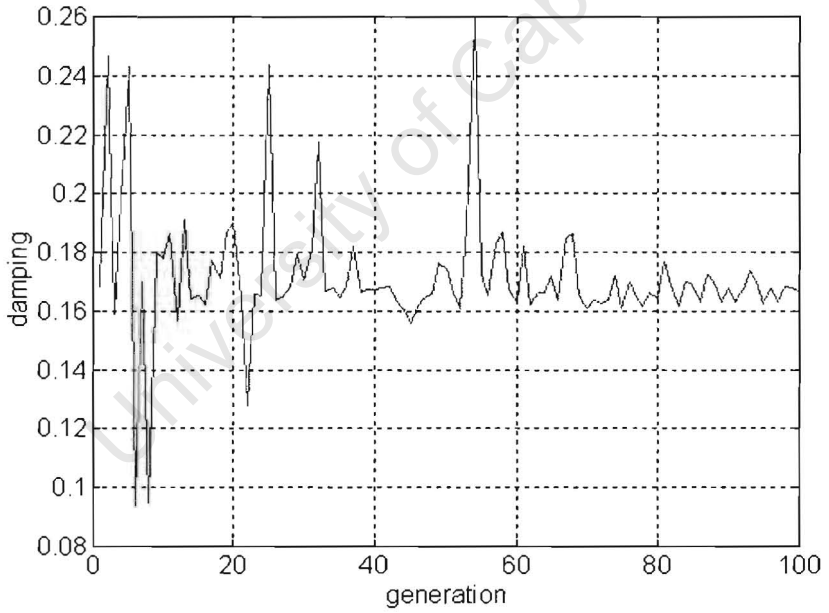

Figure 27 Convergence of damping

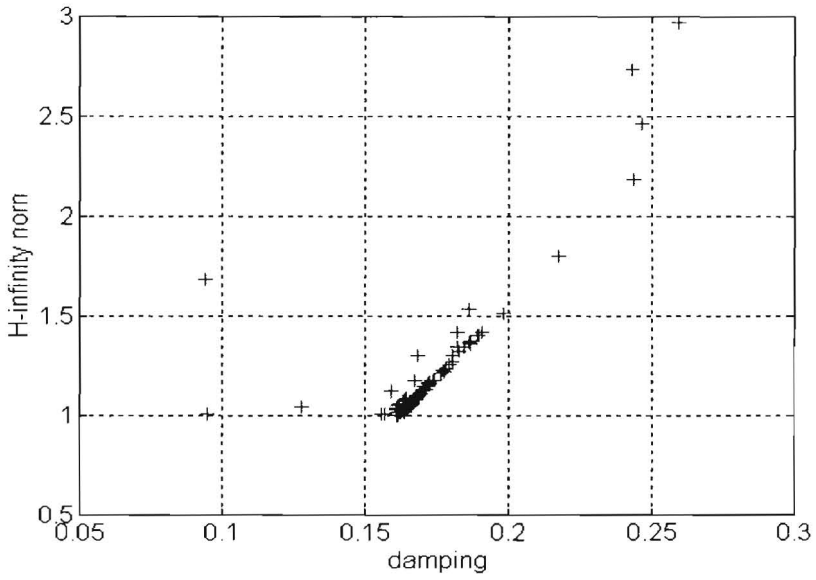

Figure 28 Plot of scatter of H_∞ norm versus damping

Figure 28 shows that the majority of solutions are located in the region with H_∞ norm of 1 and with damping factor of 0.165. This figure matches the convergence Figure 26 and Figure 27. Although one solution appears to be better than all of the others (with H_∞ norm of 2.98 and damping factor of 0.26), but this is an isolated case, and is therefore not an acceptable solution. Only converged results are the acceptable results.

The resultant PBIL-based sub-optimal H_∞ PSS has the following transfer function:

$$K_{pbil}(s) = \frac{1.285(s+0.101)(s+0.023)}{(s+0.012)(s+0.011)} \qquad 4.4$$

4.5.2 Comparison

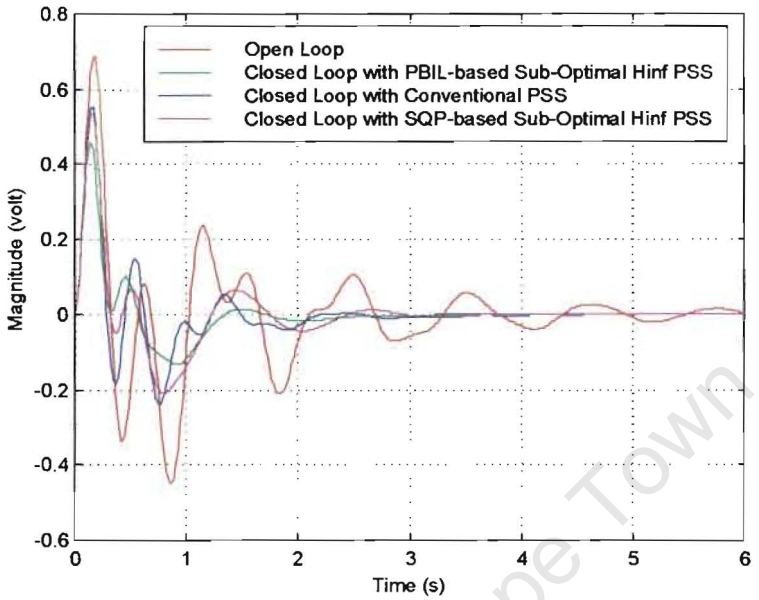

Figure 29 Step response

From the step responses shown in Figure 29, all three PSS improve the response of the system significantly, as compared to the open loop step response.

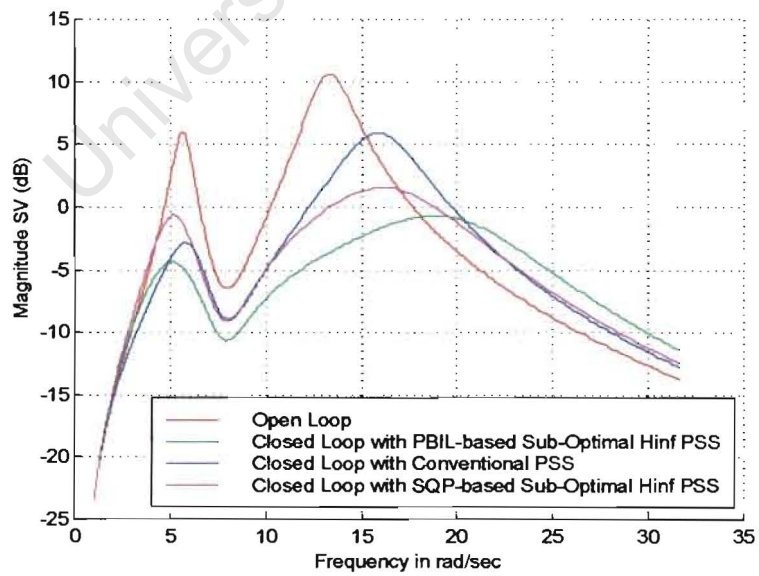

Figure 30 Bode plot

From Figure 30, the SQP-based sub-optimal H_∞ PSS damps the mode at frequency of 13.2998 rad/s well, while the conventional PSS damps the mode at frequency of 5.6926 rad/s well (this is because the conventional PSS was designed at 5.6926 rad/s peak), the PBIL-based sub-optimal H_∞ PSS offers the most damping of both dominant modes. This is expected since PBIL technique searches for a global optimal solution, while SQP technique finds only a local optimal solution. Unconstrained PBIL-based sub-optimal H_∞ PSS design method does allow one to set the limits for PSS parameters. This will avoid the simulation to give a result which is beyond the physical hardware can be tuned. Again, the parameters of PBIL-based sub-optimal H_∞ PSS and conventional PSS are real values, while the SQP-based sub-optimal H_∞ PSS are complex.

Table 14 gives the complex eigenvalues and associated dampings:

Controller	Eigenvalue (Poles)	Damping Factor	Frequency (rad/s)
PBIL-based Sub-optimal H_∞ PSS	-2.32 ± 4.80i	0.435	5.33
	-3.11 ± 6.60i	0.426	7.29
	-5.21 ± 19.4i	0.259	20.1
SQP-based Sub-optimal H_∞ PSS	-1.0571 ± 1.0546i	0.7079	1.4932
	-1.3315 ± 5.1520i	0.2502	5.3212
	-4.7594 ± 6.6581i	0.5815	8.1842
	-4.5145 ± 16.2980i	0.2669	16.9117
Conventional PSS	-5.7899 ± 0.4157i	0.9974	5.8048
	-1.3606 ± 6.0446i	0.2196	6.1959
	-5.5465 ± 4.1454i	0.8010	6.9244
	-2.3581 ±15.8054i	0.1476	15.9803
Open Loop	-0.5632 ± 5.6647i	0.0989	5.6926
	-6.5914 ± 4.3424i	0.8351	7.8932
	-1.2461 ± 13.2413i	0.0937	13.2998

Table 14 Poles and damping factors of closed loop system with various PSS and open loop system

The gain and phase margins give a good indication of the robustness of the system. The gain and phase margins are given below:

	PBIL-based	Conventional	SQP-based
Gain Margin	Infinite	19	16
Phase Margin	Infinite	63	88
ω_{gm}	n/a	56	52
ω_{pm}	n/a	19	18

Table 15 Comparison of gain and phase margin for various PSS

In terms of gain and phase margin, the PBIL-based sub-optimal H_∞ PSS is the best. The gain and phase margins can be calculated from the Nyquist plot below [13].

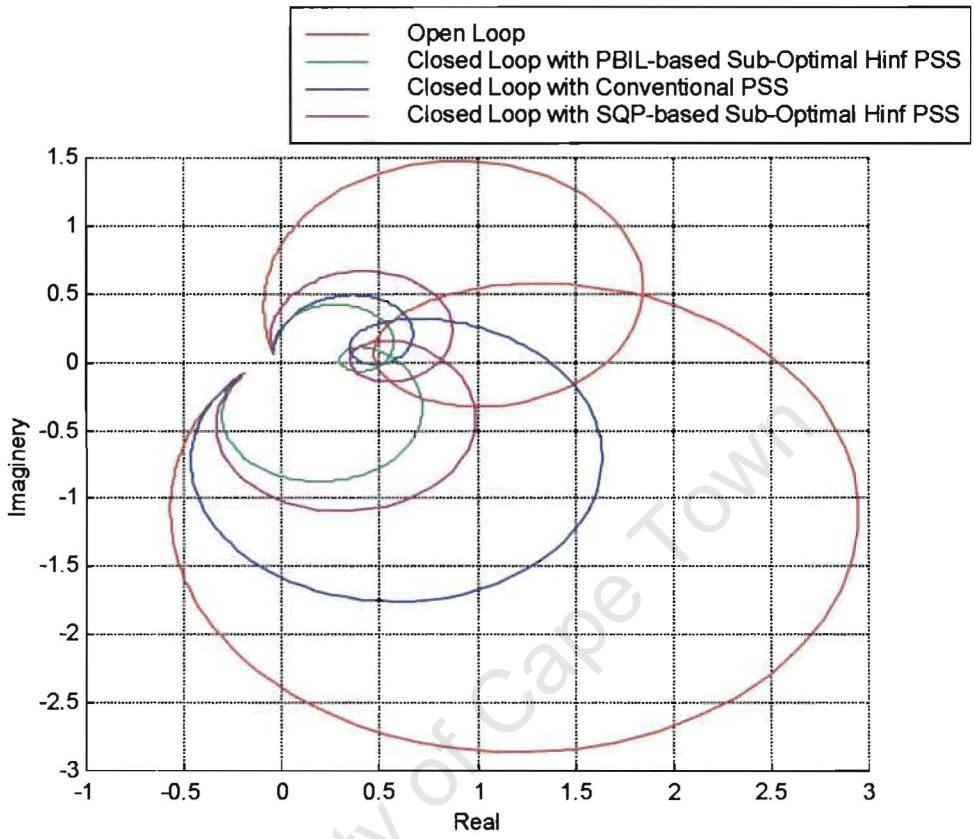

Figure 31 Nyquist plot

Figure 31 gives the Nyquist plot. PBIL-based sub-optimal H_∞ PSS gives the best performance. SQP-based sub-optimal H_∞ PSS performs better than conventional PSS.

4.6 Comparison among Various PSS

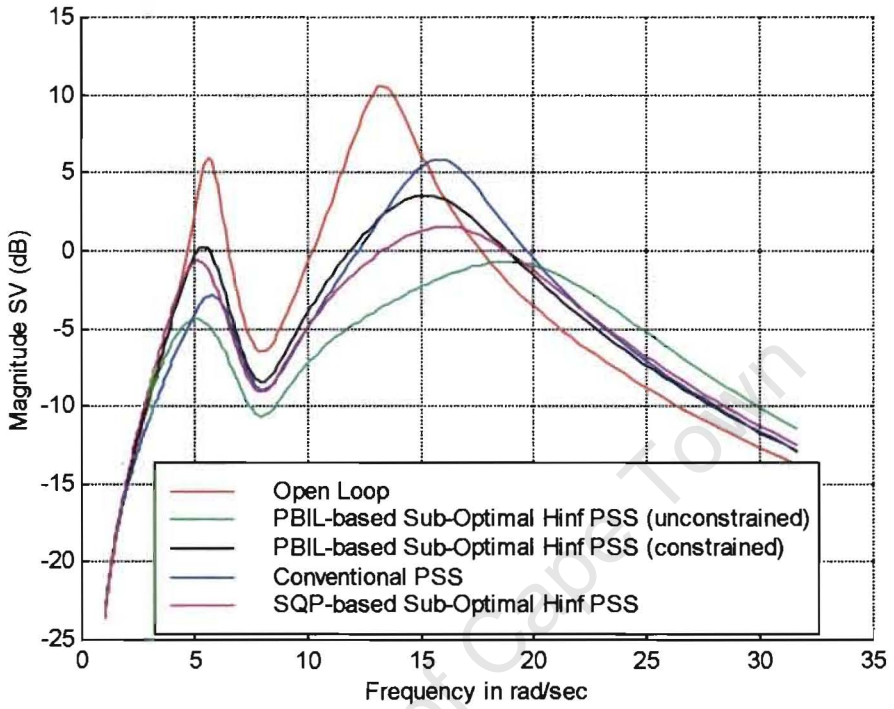

Figure 32 Singular value plot

The plot of singular value gives an indication of the overall robustness of a system. Figure 32 shows the various singular values (in dB) versus frequency. The PBIL-based sub-optimal H_∞ PSS (constrained) designed with damping as a constraint is less robust than the PBIL-based sub-optimal H_∞ PSS (unconstrained). Overall the PSS obtained using the sub-optimal H_∞ PSS design/tuning method have smaller maximum singular values than the conventional PSS and therefore are more robust.

4.7 Conclusion

This chapter detailed the application of PBIL-based sub-optimal H_∞ PSS design/tuning method. The effectiveness of PBIL-based sub-optimal H_∞ design/tuning method was demonstrated by designing PSS with and without constrain[1] for a 3M9B system. It was shown that[2]:

- PBIL-based sub-optimal H_∞ PSS are more robust than conventional PSS and SQP-based sub-optimal H_∞ PSS.
- PBIL-based sub-optimal H_∞ PSS design/tuning method gives the freedom of selecting PSS structure without requiring model reduction and optimizes the use of existing PSS.
- PBIL-based sub-optimal H_∞ PSS design/tuning method gives the freedom of doing constrained and unconstrained PSS design/tuning.
- PBIL-based sub-optimal H_∞ PSS design/tuning method gives the freedom of setting PSS parameters to be real or complex, i.e. this method is suitable for digital PSS design/tuning.
- The closed loop system with PBIL-based sub-optimal H_∞ PSS increases the overall damping of the "problem" modes.

[1] The purpose of comparing PBIL constrained to unconstrained is to demonstrate the impact that constraints can have on the implementation. It is clear that not using constraints results in a better PSS with the proposed method. SQP does result in controllers of acceptable solutions, but does require proper selection of the starting point. Had a different starting point been selected, the results may have been very different.

[2] The statements in this chapter were made as general conclusions of both constrained and unconstrained PBIL-based sub-optimal H_∞ PSS design/tuning method. They are correct and consistent.

4.8 References

[1] S. Baluja, "Population Based Incremental Learning" CMU-CS-94-163

[2] L. Chen, S Ahmed, A. Petroianu, "Robust Excitation Control of Power System using Fixed Structure Controllers" SAUPEC'95, Pretoria, South Africa

[3] S Ahmed, L. Chen, A. Petroianu, "Design of Sub-optimal H-infinity Excitation Controllers" PICA'95, Salt City, USA

[4] S Ahmed, L. Chen, A. Petroianu, "Robust H-infinity Tuning of Power System Stabilisers" IEEE Special Tech. Conference, Stockholm, Sweden 1995

[5] S. Ahmed, L. Chen, A. Petroianu, "Sub-optimal H-infinity Tuning Power System Stabilisers" IEEE Transaction on Power System, Feb. 1996, pp. 312-318

[6] L. Chen, A. Petroianu, "Tuning of PSS based on Sub-optimal H-infinity for Natal System" 13th World Congress of IFAC, San Francisco, USA 1996

[7] L. Chen, A. Petroianu, "A New Method of Tuning Power System Stabilisers", CCECE'96, Calgary, Canada

[8] L. Chen, "Report for Eskom 1996 – Robust Tuning of PSS", 1996

[9] P. M. Anderson and A. Fouad, "Power System Control and Stability", Iowa State University Press, Iowa, 1977

[10] Y. Hsu, C. Chen, "Identification of Optimum Location for Stabiliser Application Using Participation Factors", IEE Proc. Vol.134, Pt. C, No.3, pp.228-234

[11] F. Pagola, et. al., "On Sensitivities, Residues and Participations. Applications to Oscillatory Stability Analysis and Control", IEEE Transaction on Power System, Vol. 4, No.1, 1989

[12] D. Ostojic, "Identification of Optimum Site for Power System Stabiliser Applications", IEE Proc. Vol. 135, Pt. C, No. 5, Sept. 1988, pp416-419

[13] K. Dutton, et al "The Art of Control Engineering", Addison-Wesley, 1997

Chapter 5

PBIL-based Sub-optimal H_∞ Coordinated Design/Tuning of Multiple PSS and PSS-Speed Governor

In chapter 1, a brief introduction of multiple PSS and PSS-speed governor coordination has been given. The existing methods on multiple PSS coordination can effectively improve overall stability of the large system, but there are numerous disadvantages that need to be overcome. Therefore, it is worthwhile to find a solution to coordinate the tuning of multiple PSS, which can improve the overall system stability and ensure system robustness. It was also identified in chapter 1 that there are a number of benefits to coordinate the tuning of PSS and speed governor. The formulation of a method to tune both PSS and speed governor is no different from the method needed to coordinate the tuning of multiple PSS.

In this chapter, the PBIL-based sub-optimal design/tuning method is extended to allow for the coordinated tuning of the multiple PSS and PSS-speed governor. Two case studies are presented namely coordinated tuning multiple PSS and coordinated tuning PSS-speed governor. In both case studies, the Eskom Natal system has been used.

5.1 Formulation of Multiple Controllers Coordinating Problem

In order to formulate the multiple PSS and PSS-speed governor coordinating problem, we need to determine the closed loop response of the augmented power system. In this thesis, a power system with two controllers (two PSSs or one PSS and one speed governor) will be used to demonstrate the PBIL-based sub-optimal H_∞ coordinating procedures. Figure 33 gives the control configuration of the augmented power system. The power system is represented by the transfer function $\boldsymbol{G(s)}$ and the controllers (PSSs or PSS and speed governor) are represented by the transfer function $\boldsymbol{Q_1(s)}$ and $\boldsymbol{Q_2(s)}$. Transfer functions $\boldsymbol{G(s)}$, $\boldsymbol{Q_1(s)}$ and $\boldsymbol{Q_2(s)}$ have the general form defined in equation 5.1

$$F(s) = \frac{K\prod_{m=1}^{M}(s+z_m)}{\prod_{l=1}^{L}(s+p_l)} \qquad 5.1$$

Where:

K	Is a constant gain.
z_m	has the form $z_m = \alpha_{zm} \pm j\beta_{zm}$ and $\alpha_{zm}, \beta_{zm} \in \Re$
p_l	has the form $p_l = \alpha_{pl} \pm j\beta_{pl}$ and $\alpha_{pl}, \beta_{pl} \in \Re$
$-z_m$	Is the m-th finite zero of $F(s)$
$-p_l$	Is the l-th finite pole of $F(s)$
M	Is the number of zeros
L	Is the number of poles

If $M<L$ the transfer function $F(s)$ is strictly proper. In terms of an nth order differential equation, a strictly proper system has higher derivatives of the output variable than of the input variable.

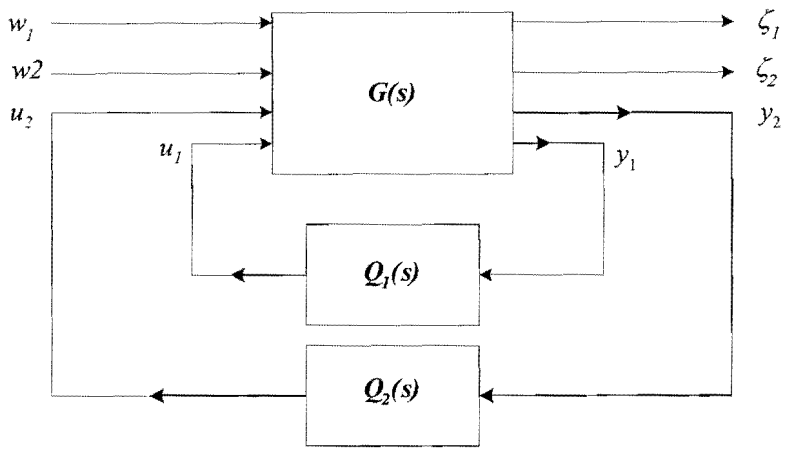

Figure 33 Control configuration

The state-space description of a 2-input 2-output power system model, $G(s)$, is defined by equation 5.2.

$$\begin{aligned} \dot{x} &= Ax + \begin{bmatrix} B_2^1 & B_2^2 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} + B_1 w \\ \varsigma &= C_1 x + \begin{bmatrix} D_{12}^1 & D_{12}^2 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} + D_{11} w \\ y &= \begin{bmatrix} C_2^1 \\ C_2^2 \end{bmatrix} x + D_{22} u + D_{21} w \end{aligned} \qquad 5.2$$

Where:

$x \in R^n$	*is the vector of power system state variables, such as machine speed, angles and fluxes.*
u	*is the vector of the input states variables, such as reference voltage and mechanical power.*
$y \in R^2$	*is the vector of output states variables, such as electrical power, machine speed and bus voltages.*
$w(t) \in R^2$	*is the disturbance vector.*
$\varsigma(t) \in R^2$	*is the performance vector.*
n	*is the number of states of the equivalent open loop system.*
A	*is the coefficient matrix of the equivalent open loop system.*
B_2^1	*is the coefficient vector associated with* u_1 *(input 1 of the open loop power system).*
B_2^2	*is the coefficient vector associated with* u_2 *(input 2 of the open loop power system).*
C_2^1	*is the coefficient vector associated with* y_1 *(output 1 of the open loop power system).*
C_2^2	*is the coefficient vector associated with* y_2 *(input 2 of the open loop power system).*
$B_1, C_1, D_{11}, D_{12}, D_{21}, D_{22}$	*are constant coefficient matrices.*

The assumption that the power system is strictly proper has been made, i.e. $D_{22} = 0$.

The state-space description of the controllers' models, $\boldsymbol{Q_1(s)}$ and $\boldsymbol{Q_2(s)}$, are defined by equation 5.3.

$$\begin{aligned} \dot{z}_1 &= Ez_1 + Fu_{c1} \\ y_{c1} &= Gz_1 + Hu_{c1} \end{aligned} \qquad 5.3$$

$$\begin{aligned} \dot{z}_2 &= Iz_2 + Ju_{c2} \\ y_{c2} &= Kz_2 + Lu_{c2} \end{aligned}$$

Where:

$z_1 \in R^{n_c}$ *and* $z_2 \in R^{n_c}$	*are the controllers' state vectors.*
u_{c1} *and* u_{c2}	*are the controllers' inputs.*
y_{c1} *and* y_{c2}	*are the controllers' outputs.*
n_c	*are the number of controllers' states.*
E,F,G,H,I,J,K,L	*are the coefficient matrices containing the unknown controllers' parameters.*

In order to obtain the closed loop state-space of the system, the outputs of the power system y are connected to the inputs of the controllers' u_c, i.e.:

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} C_2^1 \\ C_2^2 \end{bmatrix} x + D_{21}w + D_{22}u = \begin{bmatrix} u_{c1} \\ u_{c2} \end{bmatrix} \qquad 5.4$$

The outputs of the controllers y_c are connected to the inputs of the power system u,

$$\begin{aligned} u_1 &= y_{c1} = Gz_1 + Hu_{c1} \\ u_2 &= y_{c2} = Kz_2 + Lu_{c2} \end{aligned} \qquad 5.5$$

The state space description of the augmented closed loop system can be expressed as follows:

$$\begin{aligned}\begin{bmatrix}\dot{x}\\ \dot{z}_1\\ \dot{z}_2\end{bmatrix} &= \overline{A}\begin{bmatrix}x\\ z_1\\ z_2\end{bmatrix}+\overline{B}w\\ \varsigma &= \overline{C}_1\begin{bmatrix}x\\ z_1\\ z_2\end{bmatrix}+\overline{D}_1 w \qquad 5.6\\ y &= \begin{bmatrix}C_2^1\\ C_2^2\end{bmatrix}x+\overline{D}_{21}w\end{aligned}$$

Where:

$$\overline{A}=\begin{bmatrix}A+B_2^1HC_2^1+B_2^2LC_2^2 & B_2^1G & B_2^2K\\ FC_2^1 & E & 0\\ JC_2^2 & 0 & I\end{bmatrix}$$

$$\overline{B}=\begin{bmatrix}B_1^1+B_2^2LD_{21}^{21}+B_2^1HD_{21}^{11} & B_1^2+B_2^2LD_{21}^{22}+B_2^1HD_{21}^{12}\\ FD_{21}^{11} & FD_{21}^{12}\\ JD_{21}^{12} & JD_{21}^{22}\end{bmatrix}$$

$$\overline{C}_1=\begin{bmatrix}C_1+D_{12}^1HC_2^1+D_{12}^2LC_2^2 & D_{12}^1G & D_{12}^2K\end{bmatrix}$$

$$\overline{D}_1=[D_{12}^1HD_{21}^{11}+D_{11}^1+D_{12}^2LD_{21}^{21} \quad D_{12}^1HD_{21}^{12}+D_{11}^2+D_{12}^2LD_{21}^{22}]$$

From the closed loop system given by equation 5.6, the following transfer functions are defined:

$$\begin{aligned}T_{11}(s)&=T_{w1\varsigma 1}=\overline{C}_1^1(s\overline{I}-\overline{A})^{-1}\overline{B}_1^1+\overline{D}_1^1\\ T_{12}(s)&=T_{w1\varsigma 2}=\overline{C}_1^2(s\overline{I}-\overline{A})^{-1}\overline{B}_1^1+\overline{D}_1^2 \qquad 5.7\\ T_{21}(s)&=T_{w2\varsigma 1}=\overline{C}_1^1(s\overline{I}-\overline{A})^{-1}\overline{B}_1^2+\overline{D}_1^1\\ T_{22}(s)&=T_{w2\varsigma 2}=\overline{C}_1^2(s\overline{I}-\overline{A})^{-1}\overline{B}_1^2+\overline{D}_1^2\end{aligned}$$

The transfer functions $T_{ij}(s)$, where $i = \{1,2\}$ *and* $j = \{1,2\}$, give the responses of the performance variable ς_j due to changes in the disturbance variable w_i. The two variables ς_1 and w_1 are associated with output1 and input1, while ς_2 and w_2 are associated with output2 and input2.

In order to maximize the robustness of the closed loop power system to the changes in power system parameters, the objective function is defined by equation 5.8 to minimize the H_∞ norms of the closed loop power system.

$$J_{ij} = \left\| T_{ij}(s) \right\|_\infty$$
$$\min_{\xi}(\{J_{ij}\}) \qquad 5.8$$

where

$\xi = \{K, \alpha_{z1}, \alpha_{z2}, \ldots, \alpha_{zL}; \beta_{z1}, \beta_{z2}, \ldots, \beta_{zL}; \alpha_{p1}, \alpha_{p2}, \ldots, \alpha_{pM}; \beta_{p1}, \beta_{p2}, \ldots, \beta_{pM}\}$ are the unknown controllers' parameters.

$\{J_{ij}\}$ is the set of H_∞-norms.

This objective function is subject to the following constraints:

$$\gamma_k^{\min} \le \xi_k \le \gamma_k^{\max} \qquad 5.9$$
$$\mathrm{Re}\{\lambda_r(Q(s))\} < 0 \quad \text{(internal stability of the controller)}$$

where:

ξ *is the set of the unknown parameters of* $Q_1(s)$ *and* $Q_2(s)$.

$\gamma_k^{\min/\max}$ *is the lower/upper limit of* ξ_k *and* $\gamma_k^{\min/\max} \in \Re$

$\lambda_r(Q(s))$ *is the r-th eigenvalue of* $Q_1(s)$ *and* $Q_2(s)$.

Equation 5.8, with the constraints defined by equation 5.9, is by definition in the form of a General Problem (GP) [5]. The solution can be found using the methodology outlined in chapter 2.

5.2 The Power System under Investigation

The system under investigation is an equivalent system of part of Eskom Natal network. This network experienced a low frequency oscillation during the Drakensberg hydro-station conversion from the generation mode to the pump-storage mode. The equivalence model of the network was obtained using PSS/E's equivalent toolbox. The equivalent system model has been tested in the micro-machines lab at the University of Natal. The real PSS was taken from Drakensberg Hydro-station and installed in the lab. A number of parameters were tested to prove the accuracy of the equivalent system [6]. Figure 34 gives the part of the Eskom Natal network.

At the time of the research, the PSSs in Natal were not tuned in a coordinated manner, since they were disconnected from the system. Therefore, no comparison was possible between existing PSSs and PSSs tuned by using the new method developed in this thesis. The decision not use SQP as a comparison was that the validity of the method presented in the thesis was already established. The PBIL formulation provides greater flexibility in selection of constraints and provides a global solution, versus a local optimum solution in SQP. In addition, the need to select a starting point in SQP makes it impractical as compared to the PBIL formulation.

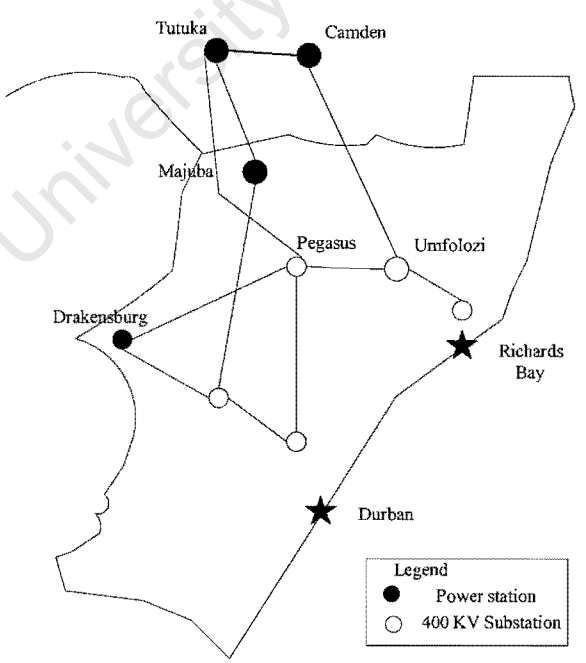

Figure 34 Part of the Eskom (Natal) network

Table 16 gives the open loop power system eigenvalues and damping factors. The "problem" modes - weakly damped modes - are at frequency of 5.36 rad/s with a damping factor of 0.102 and 7.14 rad/s with a damping factor of 0.192.

Open Loop System		
Eigenvalue (Pole)	**Damping Factor**	**Frequency (rad/s)**
-4.11e-001	1.00e+000	4.11e-001
-4.29e-001 ± 3.90e-001i	7.40e-001	5.80e-001
-5.44e-001 ± 5.33e+000i	**1.02e-001**	**5.36e+000**
-1.37e+000 ± 7.01e+000i	**1.92e-001**	**7.14e+000**
-2.98e+000 ± 3.44e+000i	6.54e-001	4.55e+000
-1.20e+001	1.00e+000	1.20e+001
-1.52e+001	1.00e+000	1.52e+001
-2.87e+001 ± 5.28e+000i	9.84e-001	2.92e+001
-4.09e+001	1.00e+000	4.09e+001
-4.39e+001	1.00e+000	4.39e+001
-4.91e+001	1.00e+000	4.91e+001
-1.03e+002	1.00e+000	1.03e+002
-1.48e+002	1.00e+000	1.48e+002

Table 16 Poles and damping factors of open loop power system

5.3 Case Study 1– Coordinated Tuning of Multiple PSS

There are two PSS installed on the generators, one in Drakensberg, the other in Tutuka. In the case study the PSS's parameters were determined using the coordinated design/tuning method outlined in this chapter.

5.3.1 Results

The existing PSS at Drakensberg and Tutuka are standard BBC first order PSS. Equation 5.10 gives the transfer functions of both PSS. The PSS parameters were obtained by using PBIL-based sub-optimal H_∞ coordination method. See Appendix H for the system model and the MATLAB scripts.

$$K_{pss1}(s) = \frac{(0.001787s + 2.078\times10^{-6})}{s+19}$$
$$K_{pss2}(s) = \frac{(6.251s + 62.35)}{s+16.08} \qquad 5.10$$

(PSS1 for Drakensberg and PSS2 for Tutuka)

Figure 35 shows the plots of convergence for minimum damping and maximum singular value. The optimal solution was found after 55 generations.

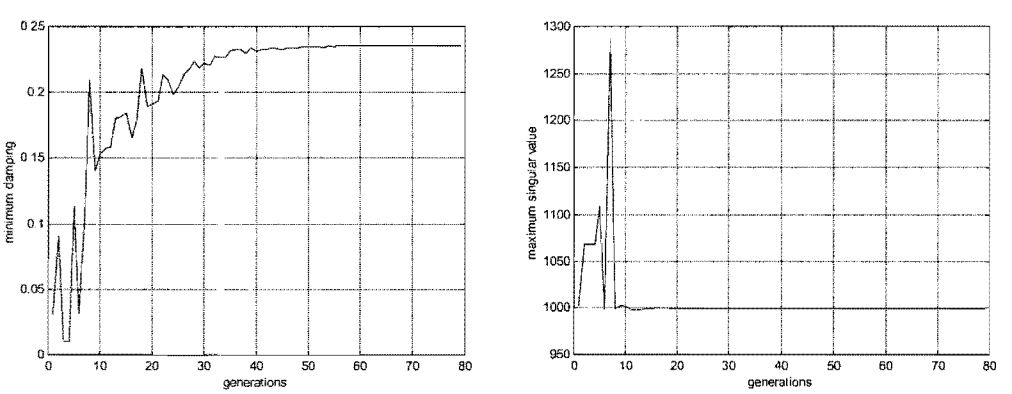

Figure 35 Plots of convergence

Closed Loop System		
Eigenvalue (Pole)	**Damping Factor**	**Frequency (rad/s)**
-4.11e-001	1.00e+000	4.11e-001
-4.02e-001 ± 3.99e-001i	7.09e-001	5.67e-001
-2.96e+000 ± 3.44e+000i	6.52e-001	4.54e+000
-1.28e+000 ± 5.17e+000i	**2.39e-001**	**5.33e+000**
-1.42e+000 ± 5.87e+000i	**2.35e-001**	**6.04e+000**
-1.16e+001	1.00e+000	1.16e+001
-1.44e+001 ± 1.24e+000i	9.96e-001	1.44e+001
-1.90e+001	1.00e+000	1.90e+001
-3.58e+001 ± 1.64e+001i	9.10e-001	3.94e+001
-4.09e+001	1.00e+000	4.09e+001
-4.34e+001	1.00e+000	4.34e+001
-4.91e+001	1.00e+000	4.91e+001
-1.03e+002	1.00e+000	1.03e+002
-1.49e+002	1.00e+000	1.49e+002

Table 17 Poles and damping factors for closed loop power system

Table 17 gives the eigenvalues and damping factors for the closed loop power system with both PSS connected. The damping of the "problem" mode at frequency of 5.36 rad/s has increased from 0.102 to 0.239, while the damping of the other "problem" mode at frequency 7.14 rad/s has increased from 0.192 to 0.235.

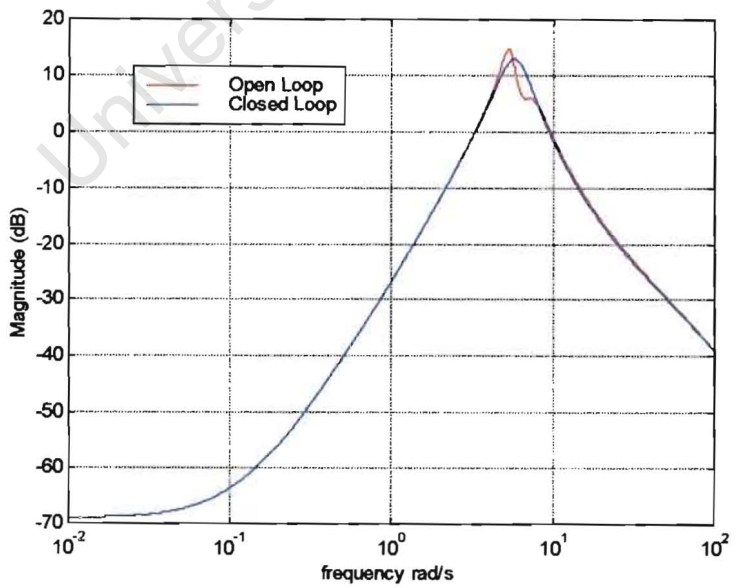

Figure 36 Singular value (for transfer function T_{11}) of the open loop and closed loop power system

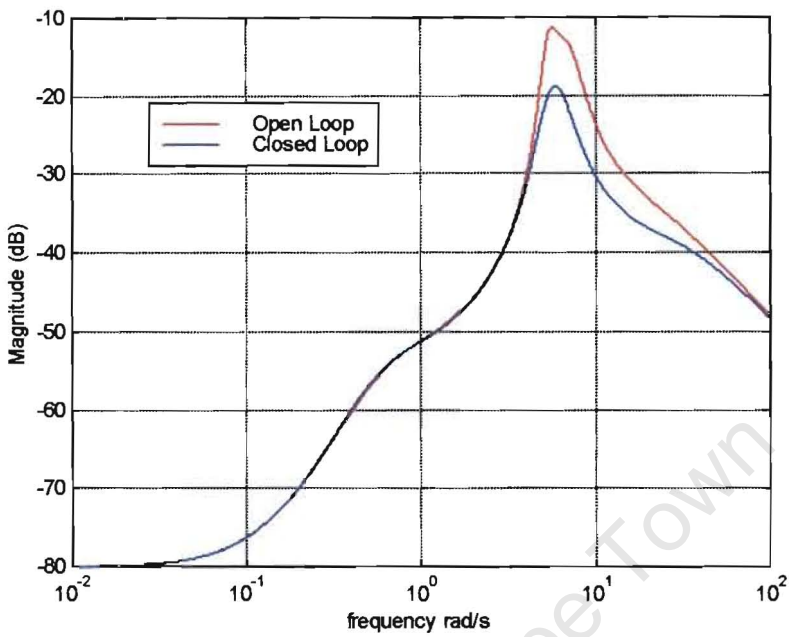

Figure 37 Singular value (for transfer function T_{12}) of the open loop and closed loop power system

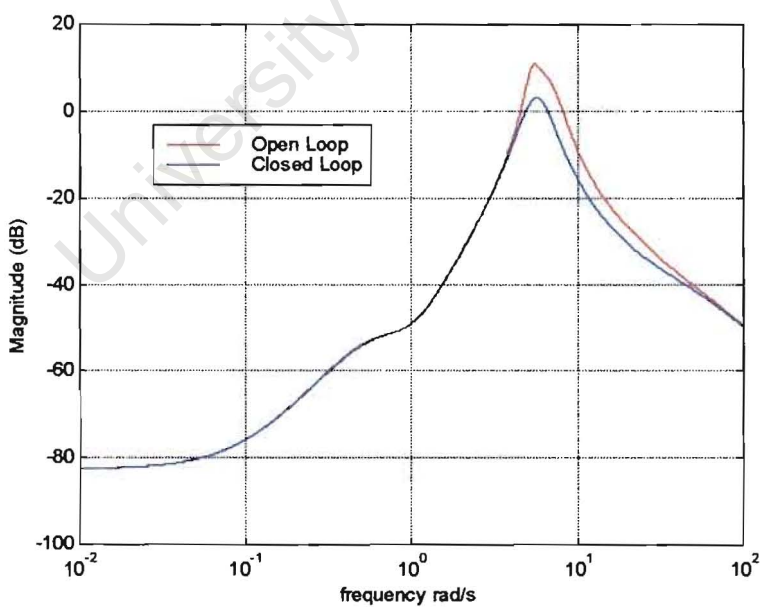

Figure 38 Singular value (for transfer function T_{21}) of the open loop and closed loop power system

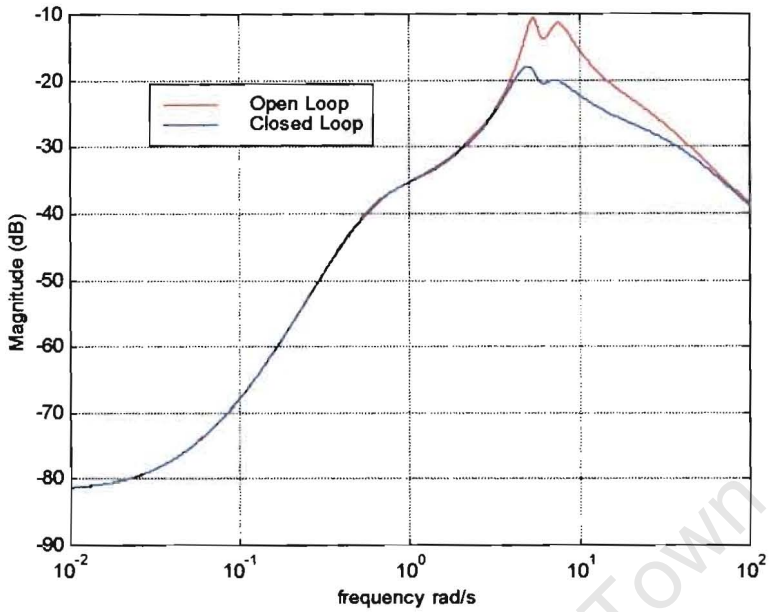

Figure 39 Singular value (for transfer function T_{22}) of the open loop and closed loop power system

Figure 36 illustrates the plots of singular value of the transfer function (T_{11}) between the output ΔP_{elec} and the input ΔV_{ref} at Drakensberg for the open loop and closed loop power system. Figure 37 illustrates the plots of singular value of the transfer function (T_{12}) between the output ΔP_{elec} at Tutuka and the input ΔV_{ref} at Drakensberg for the open loop and closed loop power system. Figure 38 illustrates the plots of singular value of the transfer function (T_{21}) between the output ΔP_{elec} at Drakensberg and the input ΔV_{ref} at Tutuka for the open loop and closed loop power system. Figure 39 illustrates the plots of singular value of the transfer function (T_{22}) between the output ΔP_{elec} and the input ΔV_{ref} at Tutuka for the open loop and closed loop power system. From Figure 36 to Figure 39, it can be seen that the robustness of the closed loop power system with two PSS has been improved.

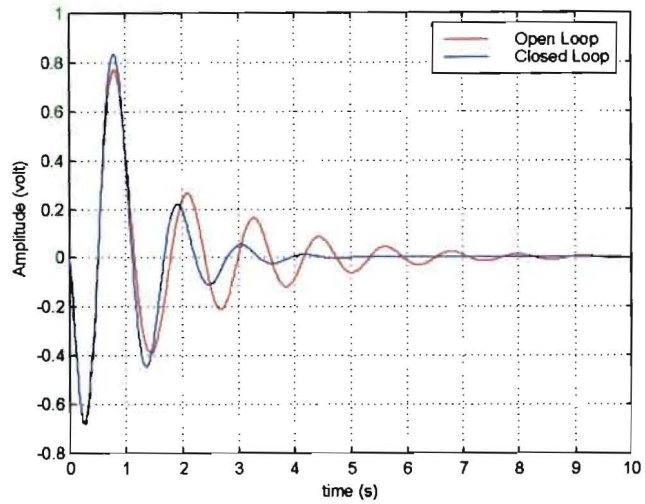

Figure 40 Step response (for transfer function T_{11}) of the open loop and closed loop power system

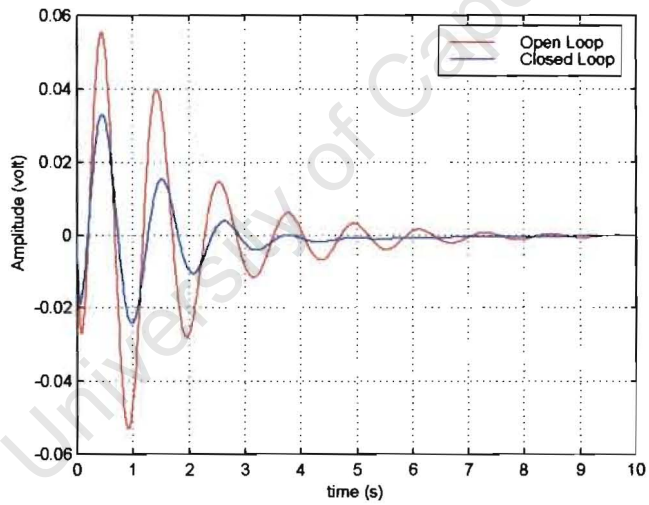

Figure 41 Step response (for transfer function T_{12}) of the open loop and closed loop power system

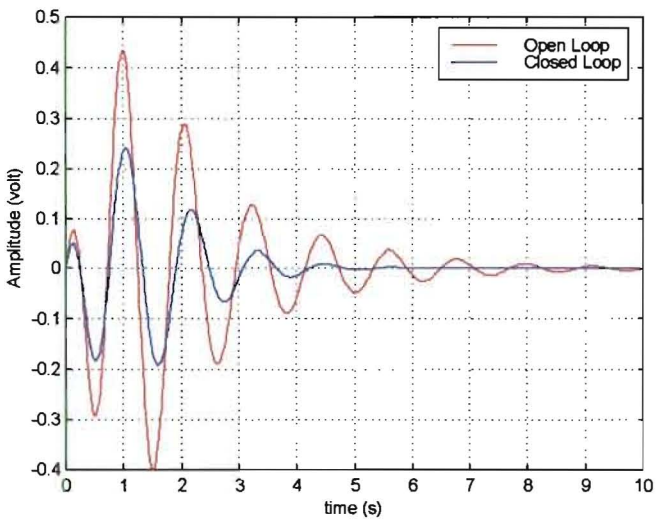

Figure 42 Step response (for transfer function T_{21}) of the open loop and closed loop power system

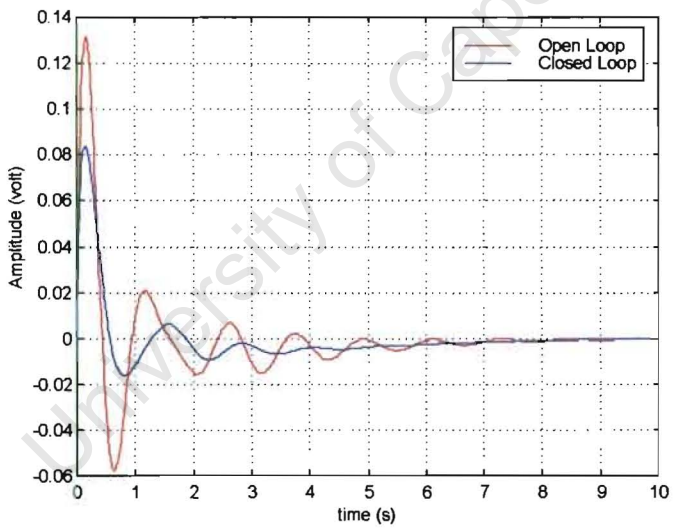

Figure 43 Step response (for transfer function T_{22}) of the open loop and closed loop power system

Figure 40 illustrates the response of ΔP_{elec} at Drakensberg to a step in ΔV_{ref} at Drakensberg (transfer function (T_{11})) for the open loop and closed loop power system. Figure 41 illustrates the response of ΔP_{elec} at Tutuka to a step in ΔV_{ref} at Drakensberg (transfer function (T_{12})) for the open loop and closed loop power system. Figure 42 illustrates the response of ΔP_{elec} at Drakensberg to a step in ΔV_{ref} at Tutuka (transfer function (T_{21})) for the open loop and closed loop power system. Figure 43 illustrates the response of ΔP_{elec} at Tutuka to a step in ΔV_{ref} at Tutuka (transfer function (T_{22})) for the open loop and closed loop power system.

From Figure 40 to Figure 43, it can be seen that the open loop step response is weakly damped with oscillations persisting after 8 to 10 seconds. The closed loop systems have significantly improved damping of the oscillation. For the closed loop system with both PSS, the oscillation settles within 4 seconds.

5.4 Case Study 2– Coordinated Tuning of PSS-Speed Governor

In this case study, the PSS and speed governor will be tuned together, rather than tuning the PSS and speed governor separately. This not only simplifies the design procedure, but also takes into account any coupling or influences, which the PSS and speed governor have one over another. This case study introduces the coordinating procedure, while PSS and speed governor structures are selected for theoretical study only.

5.4.1 Results

The governor system was modeled in PSS/E by using HYGOVM model, which especially models hydro plant governor system. This governor system includes three parts, i.e. three feedback control loops. They are: speed-governor, jet deflector and gate servo, as shown in Appendix I. Speed-governor is one of the feedback loops, which has a first order controller plus wash-out. The PSS at Drakensberg was a second order PSS. All the other "fixed parts" (i.e. all the components with unchangeable parameters) in the power system had been included in the system model. The following parameters were obtained by using PBIL-based sub-optimal H_∞ coordination method. See Appendix I for the power system model and the MATLAB scripts. Equation 5.11 gives the transfer functions of the PSS and speed governor.

$$K_{pss}(s) = \frac{(49.27s^2 + 1.099s + 0.005265)}{(s^2 + 171.4s + 7346)}$$
$$K_{gov}(s) = \frac{(91.67s + 77.3)}{(s^2 + 32.45s + 251.1)} \qquad 5.11$$

Closed Loop System		
Eigenvalue (Pole)	**Damping Factor**	**Frequency (rad/s)**
-4.11e-001	1.00e+000	4.11e-001
-4.29e-001 ± 3.90e-001i	7.40e-001	5.80e-001
-2.00e+000 ± 4.93e+000i	3.76e-001	5.32e+000
-1.26e+000 ± 5.72e+000i	**2.16e-001**	**5.85e+000**
-3.72e+000 ± 5.13e+000i	**5.87e-001**	**6.34e+000**
-1.21e+001	1.00e+000	1.21e+001
-1.43e+001 ± 1.71e+000i	9.93e-001	1.44e+001
-1.88e+001	1.00e+000	1.88e+001
-2.89e+001 ± 5.09e+000i	9.85e-001	2.93e+001
-4.20e+001	1.00e+000	4.20e+001
-4.38e+001	1.00e+000	4.38e+001
-4.87e+001	1.00e+000	4.87e+001
-1.88e+001 ± 5.12e+001i	3.44e-001	5.46e+001
-1.48e+002	1.00e+000	1.48e+002
-1.88e+002	1.00e+000	1.88e+002

Table 18 Poles and damping factors for closed loop power system

Table 18 gives the eigenvalues and damping factors for the closed loop system with both PSS and speed governor. The damping factor of the "problem" mode at frequency of 5.36 rad/s has increased from 0.102 to 0.216, while the other "problem" mode at frequency 7.14 rad/s has increased from 0.192 to 0.587.

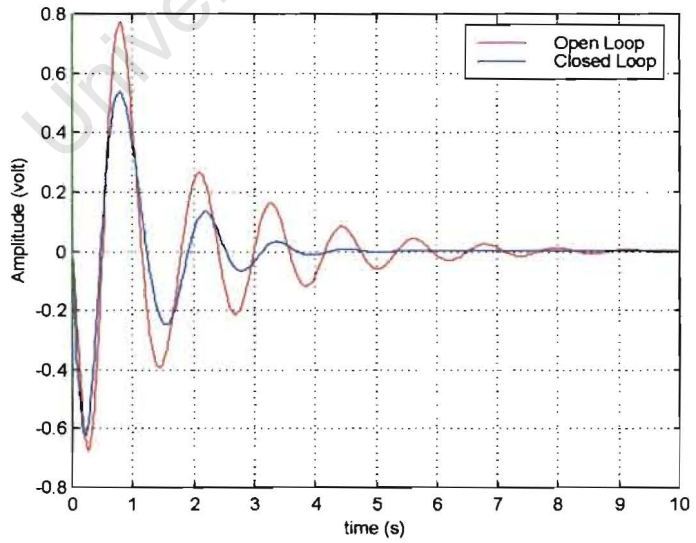

Figure 44 Step response (for transfer function T_{11}) of the open loop and closed loop power system

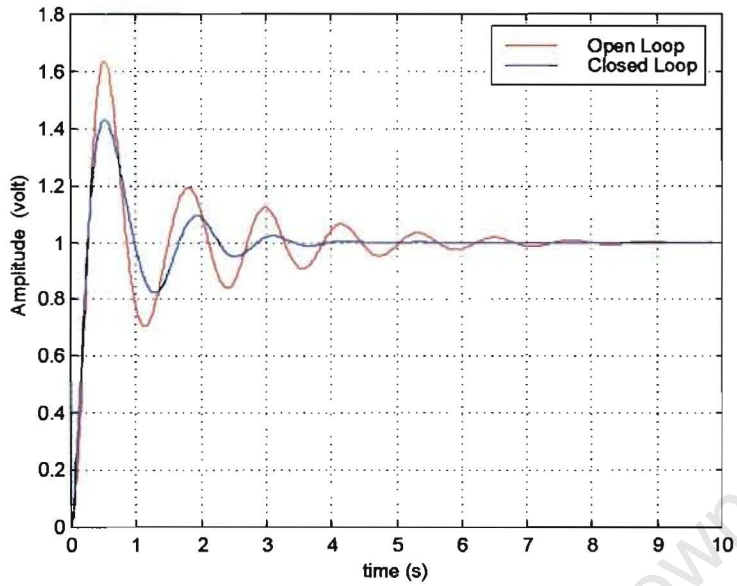

Figure 45 Step response (for transfer function T_{12}) of the open loop and closed loop power system

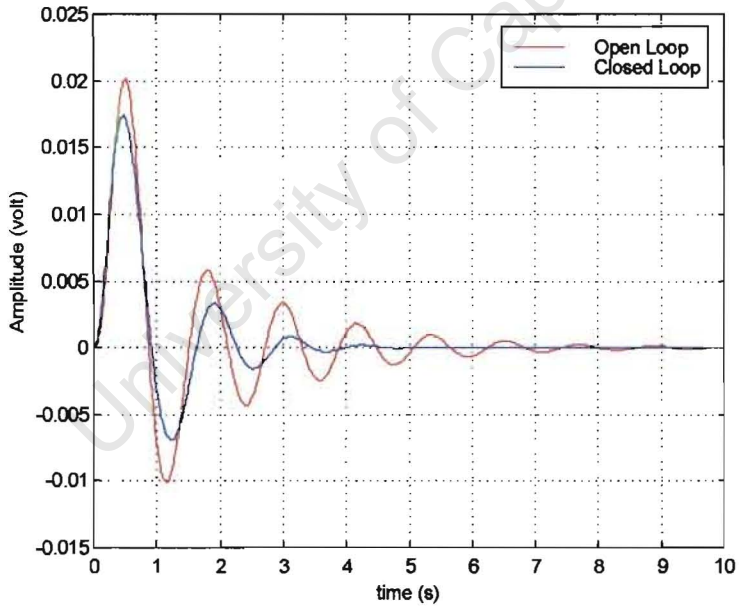

Figure 46 Step response (for transfer function T_{21}) of the open loop and closed loop power system

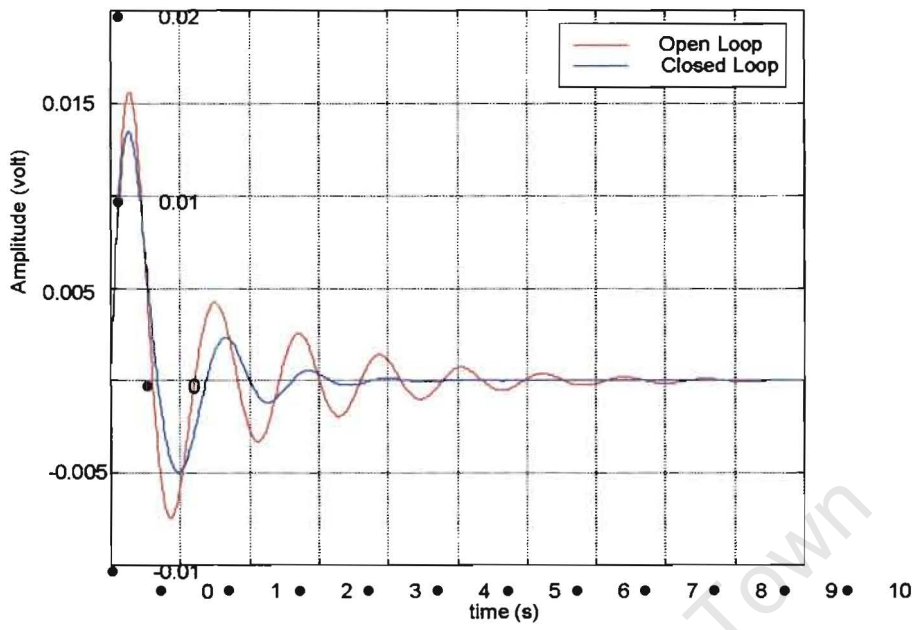

Figure 47 Step response (for transfer function T_{22}) of the open loop and closed loop power system

Figure 44 illustrates the response of ΔP_{elec} to a step in ΔV_{ref} (transfer function (T_{11})) for the open loop and the closed loop power system. Figure 45 illustrates the response of *Speed deviation* to a step in ΔV_{ref} (transfer function (T_{12})) for the open loop and the closed loop power system. Figure 46 illustrates the response of ΔP_{elec} to a step in ΔP_{mrch} (transfer function (T_{21})) for the open loop and the closed loop power system. Figure 47 illustrates the response of *Speed deviation* to a step in ΔP_{mrch} (transfer function (T_{22})) for the open loop and the closed loop power system.

From Figure 44 to Figure 47, it can be seen that the open loop step response is weakly damped with oscillations persisting after 8-10 seconds. The closed loop power systems have significantly improved the damping of the oscillation. For the closed loop power system with both PSS, the oscillation settled within 3.5 seconds.

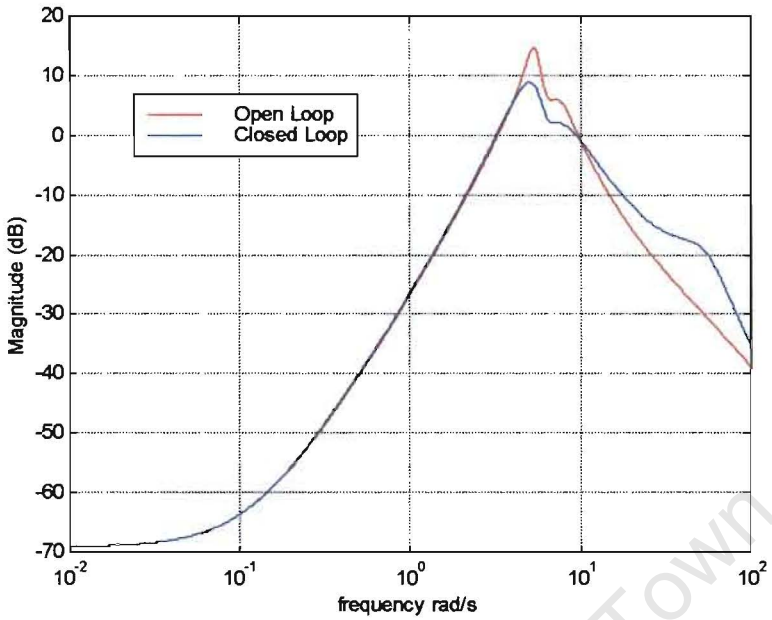

Figure 48 Singular value (for transfer function T_{11}) of the open loop and closed loop power system

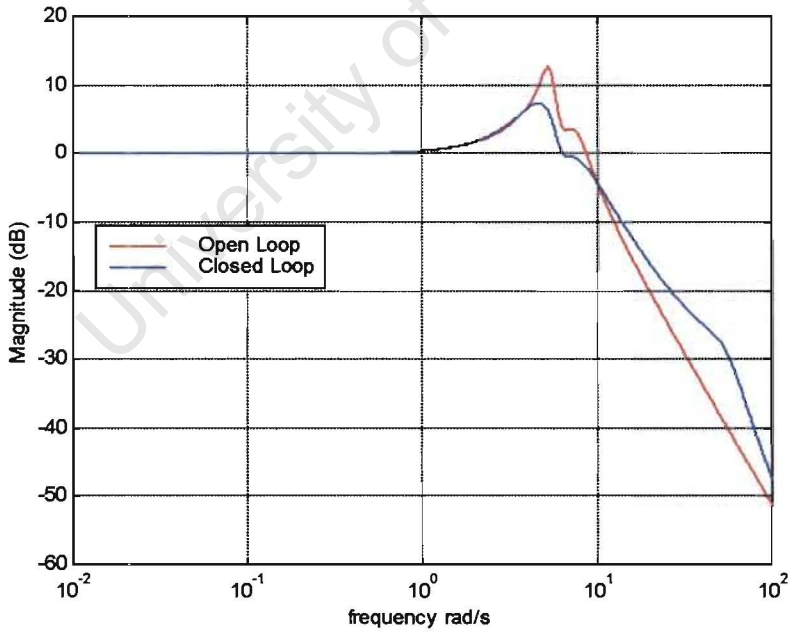

Figure 49 Singular value (for transfer function T_{12}) of the open loop and closed loop power system

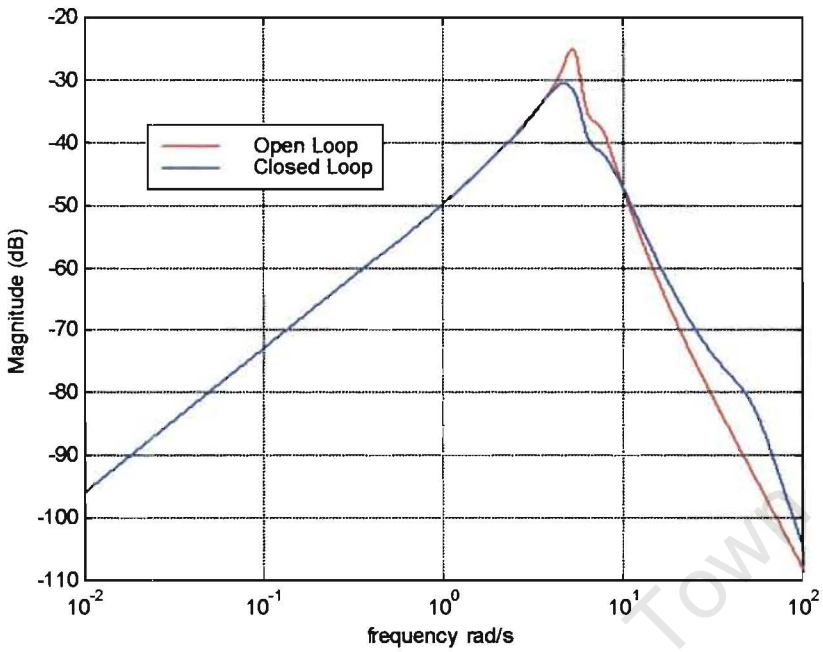

Figure 50 Singular value (for transfer function T_{21}) of the open loop and closed loop power system

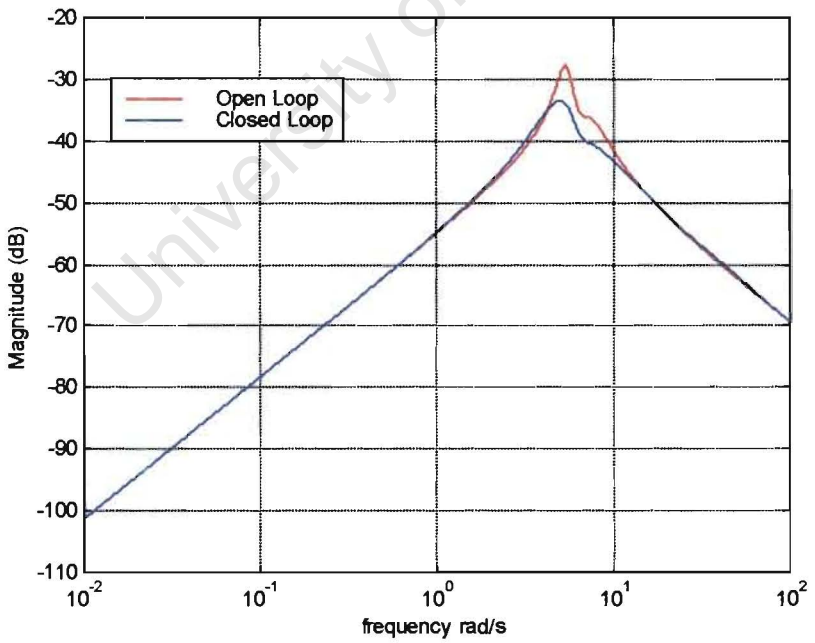

Figure 51 Singular value (for transfer function T_{22}) of the open loop and closed loop power system

Figure 48 illustrates the plots of singular value of the transfer function (T_{11}) between the output ΔP_{elec} and the input ΔV_{ref} for the open loop and closed loop power system. Figure 49 illustrates the plots of singular value of the transfer function (T_{12}) between the output *Speed dviatione* and the input ΔV_{ref} for the open loop and closed loop power system. Figure 50 illustrates the plots of singular value of the transfer function (T_{21}) between the output ΔP_{elec} and the input ΔP_{mech} for the open loop and closed loop power system. Figure 51 illustrates the plots of singular value of the transfer function (T_{22}) between the output *Speed deviation* and the input ΔV_{ref} for the open loop and closed loop power system. From Figure 48to Figure 51, it can be seen that the closed loop power system is stable and has better robustness than the open loop power system.

5.5 Conclusion

In this chapter, a new method for coordinated design/tuning multiple PSS and PSS-speed governor was presented, namely PBIL-based sub-optimal H_∞ coordinating multiple PSS and PSS-speed governor. The method is based on formulating the coordination problem as a numerical optimization problem. The objective function is taken as the H_∞ norm of the closed loop disturbance-related transfer function. This method is flexible at the controller structures, therefore, is practical. An Eskom equivalent network was used to demonstrate the procedure and prove the following:

- PBIL-based sub-optimal H_∞ coordination method can ensure the overall robustness of the closed loop system.
- PBIL-based sub-optimal H_∞ coordination method can increase the overall damping factor of the closed loop system.
- PBIL-based sub-optimal H_∞ coordination method allows operator to not only coordinate multiple PSS, but also multiple controllers for large power system, i.e. PSS-speed governor coordination and PSS-AVR coordination[1].
- PBIL-based sub-optimal H_∞ coordination method allows operator to select controllers' structures and settle the limits on controllers' parameters.

[1] The intention of this chapter is purely to introduce a new method for coordinated design/tuning multiple controllers. While applying this method to AVR-PSS coordination, the program will see AVR as another controller. All the procedure will be the same as we demonstrated in this thesis.

5.6 References

[1] K. Dutton, et al "The Art of Control Engineering", Addison-Wesley, 1997

[2] B. Friedland, "Control System Design", McGraw-Hill , 1987

[3] M. Green, D. Limebeer "Linear Robust Control", Prentice Hall, 1995

[4] MATLAB Robust Control Toolbox Manual, The Mathworks, Natick, 1998

[5] MATLAB Optimization Toolbox Manual, The Mathworks, Natick, 1998

[6] M. Wishart, "Network Stability", Eskom, Technology Research and Investigation, Eskom Reference# TRR/E/95/EL163

Chapter 6

Conclusion

This dissertation addressed the design/tuning of robust Power System Stabilizers for damping of electromechanical oscillations in power systems. The thesis work focused on three key issues, namely: design/tuning of robust PSS, dealing with non-convex optimization problems in robust PSS design/tuning and the coordinated design/tuning of multiple PSS.

In Chapter 2, we focused on design/tuning robust PSS. A new method based on the principles of optimal H_∞ control was developed. This method uses overall system robustness as objective function and minimum overall damping factor as constraint. The optimization method used was Sequential Quadratic Programming. SQP is good at searching for local optimum. The method was used to design a PSS for a SMIB system. The resultant PSS was compared to a conventional PSS and an optimal H_∞ PSS.

In Chapter 3, we focused on dealing with non-convex optimization problems. The SQP-based optimization method used in the sub-optimal H_∞ robust PSS design/tuning method was replaced with a PBIL implementation. PBIL is a combination of Genetic Algorithms and Competitive Learning. PBIL uses a probability vector to present the best solution. PBIL is well suited to deal with non-convex optimization problems. In Chapter 3 it was proven, through an example, that the PBIL-based sub-optimal H_∞ robust controller design/tuning method can be used to design controllers that have the same characteristics as standard optimal H_∞ controllers.

In Chapter 4, we focused on application of PBIL-based sub-optimal H_∞ PSS design/tuning method to a multiple machine power systems. In Chapter 4 we analyzed the effect of including constraints in the PBIL-based sub-optimal H_∞ PSS design/tuning method. Two PSS (one constrained and the second unconstrained) were tuned for a 3-

machine-9-bus power system. The effectiveness of the controllers on the power system was compared to the effectiveness of conventional and SQP-based sub-optimal H_∞ PSS.

In Chapter 5, we focused on PBIL-based coordinated design/tuning of multiple controllers, i.e. multiple PSS coordination and PSS-speed governor coordination. Using an Eskom Natal network case study, PBIL-based sub-optimal H_∞ coordinated design/tuning method was proven to improve overall robustness and damping factors of the system.

The thesis makes several contributions to the robust design/tuning and the coordination of power system control system field. They are:

Chapter 2: A new SQP-based sub-optimal H_∞ PSS design/tuning method.

Based on optimal H_∞ control, SQP-based sub-optimal H_∞ PSS design/tuning method overcomes the disadvantages of optimal H_∞ PSS. PSS, obtained using the sub-optimal H_∞ PSS design/tuning method; improve the overall robustness and the damping factors of the power system. The method does not affect the observability of the system and it does not modify the original power system's characteristics. This method allows the PSS designer to choose the structure of the PSS.

Chapter 3: Dealing with non-convex optimization Problem.

The idea of introducing PBIL to the large power system made design/tuning PSS for large system become possible. PBIL was developed for solving non-convex optimization problems. PBIL overcomes the shortcomings of SQP, since it is well suited to finding global optimum solutions. In Chapter 3 it was shown that the PBIL-based sub-optimal H_∞ design/tuning method can be used to design near-optimal H_∞ controllers.

Chapter 4: Application of PBIL-based sub-optimal H_∞ PSS design/tuning to a multiple machine power system.

PBIL-based sub-optimal H_∞ PSS design/tuning method was applied to a multi-machine power system. Two case studies were presented. The case studies demonstrated the effect of constrained and unconstrained optimization in the design/tuning of PSS. It was shown in the case study that constraints can be used to select the effect of the PSS on the overall power system.

Chapter 5: PBIL-based coordinated design/tuning of multi-controllers.

PBIL-based coordinated PSS design/tuning method was introduced. The method was applied to an Eskom system to demonstrate the benefit of PSS coordination. The method gives the PSS designer many degrees of freedom to select the PSS' structures, the location of the PSS and define the limits on the PSS' parameters. This method ensures the overall robustness and damping of the system. The application of this method to coordinating the design/tuning of PSS and speed governor was demonstrated as well.

Appendix A

Application of Classical Design Techniques to PSS

A.1 Classical Lead-Lag Design

The standard method of conventional PSS design is based on eigenvalue analysis and frequency response analysis [1,2]. The procedure follows standard lead-lag design principles [3]. The PSS design procedure is as follows:

1. Linearise the power system and obtain state-space representation.
2. Plot the Bode diagram of the open loop system.
3. Determine the gain and phase requirements at the frequency of the dominant mode. The number of phase-lead controllers required is determined from the phase requirement. The phase-lead controller is expressed mathematically by equation A.1.

$$H(s) = k\frac{1+(aT)s}{1+Ts} \quad where \quad a>1 \tag{A.1}$$

k is gain

a is a scaling factor

T is a time constant

4. Minimize the high frequency gain (which amplifies the signal noise level), the parameter "a" should be as small as possible. The maximum phase-lead angle (Φ_{max}) obtained with the phase advance unit is given by equation A.2.

$$\Phi_{max} = \sin^{-1}\left(\frac{a-1}{a+1}\right) \tag{A.2}$$

The time constant T determines the frequency ω_{max} at which the maximum phase-lead angle occurs. Equation A.3 defines ω_{max}.

$$\omega_{max} = \frac{1}{T\sqrt{a}} \tag{A.3}$$

The parameter "*a*" is chosen by making the maximum phase advance equal to the required phase shift at the frequency of the dominant mode. The constant T is chosen to make this maximum occur at the frequency of the dominant mode.

The number of phase-advanced controllers required is dependent on the parameter "*a*". For example:

If a phase lead of 70° is required at a frequency of 5.1 rad/s, then a minimum of two phase-lead controllers will be required. If one controller is used "a" is around 130, while if two controllers are used, "a" is only 4.

The gain (k) of the phase-advance controller is used to determine the effect of the input voltage on the controller. Using root-locus techniques, the stability of the system as a function of k, can be determined. The value of k is varied and the trajectories of the critical poles as a function of k are determined. When the poles cross the imaginary axis, the system becomes unstable.

Lead-lag design, as with most classical design techniques, has a number of short-comings including:

- Robustness is not part of the design algorithm. Robustness is determined once the design is complete through simulation and sensitivity analysis. If the controller is not robust, the procedure is repeated.
- The design is based on the dominant mode, while ignoring other system modes that will be impacted by the controller.

A.2 References

[1] Y. Yu, "Electric Power System Dynamics", Academic Press, 1980

[2] N. Martins, L. Lima, "Eigenvalue and Frequency Domain Analysis of Small-signal Electromechanical Stability Problems", IEEE Publication, Eigenanalysis and Frequency Domain Methods for System Dynamic Stability (90^{TH}0292-3-PWR) 1989, pp17-33

[3] D.J. Trudnowski, et al. "An Application of Prony Methods in PSS Design For Multimachine Systems", IEEE Trans. on Power Systems, Vol.6, No.1 February 1991, pp118-126

Appendix B

Optimal H_∞ PSS design

Optimal H_∞ control is a frequency-domain optimization and synthesis theory that was developed in response to the need for a synthesis method that addresses the question of modeling errors. The basic philosophy of H_∞ is to treat the worst-case scenario: "*if you don't know what you are up against, plan for the worst and optimize.*" For such a strategy to be useful, it must have the following properties:

- It must be capable of dealing with system modeling errors and unknown disturbances.
- It should represent a natural extension to existing feedback control theory.
- It must be amenable to meaningful optimization.
- It must be able to deal with multi-variable problems.

B.1 Brief Descriptions of Concepts for Optimal H_∞

In this section, definitions and brief descriptions of concepts for optimal H_∞ control will be given. Some of the material in this section is in the domain of the control theorist, which may not be familiar to power system engineers. Nevertheless, these concepts are essential for understanding of the thesis. For more details, [1-3] should be consulted.

B.1.1 Stability and the H_∞ Norm

This section will introduce the infinity norm and H_∞ optimal control with the aid of a sequence of simple single-loop example. To motivate the introduction of the infinity norm, the question of robust stability optimization for the feedback system shown in Figure 52 will be considered.

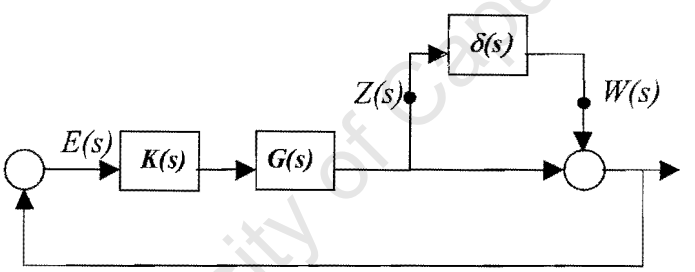

Figure 52 Typical feedback system with disturbance

Where:

G(s) Is a nominal linear, time-invariant model of an open-loop system (In our case it is a power system)

K(s) Is a linear, time-invariant model of controller to be designed (In our case it is a PSS)

δ(s) Is a multiplicative perturbation, which is an unknown linear, time-invariant system.

If the "true" system is represented by $(1+\delta(s))G(s)$, then the modeling error is represented by a multiplicative perturbation $\delta(s)$ at the system output.

The ideal stabilizing controller $K(s)$ cannot be determined easily from Figure 52. Therefore, it is necessary to obtain the equivalent closed-loop system, as shown in Figure 53.

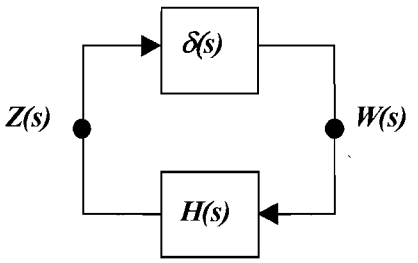

Figure 53 Equivalent closed loop system

$H(s)$ in Figure 53, where $H(s) = Z(s)/W(s)$, is determined as follows:

$$\begin{aligned} & E(s) && = K(s)G(s)E(s) + W(s) \\ \therefore\ & E(s)(1 - K(s)G(s)) && = W(s) \\ & E(s) && = \left[1 - K(s)G(s)\right]^{-1} W(s) \\ & Z(s) && = G(s)K(s)E(s) \\ \therefore\ & Z(s) && = \left[1 - G(s)K(s)\right]^{-1} G(s)K(s)W(s) \end{aligned}$$

The stability properties of the system given in Figure 52 are the same as those given in Figure 53 , in which

$$H(s) = (1 - G(s)K(s))^{-1} G(s)K(s)$$

If the perturbation $\delta(s)$ and the nominal closed-loop system given by $H(s)$ are both stable, the Nyquist criterion says that the closed-loop system is stable if and only if the Nyquist diagram of $H(s)\delta(s)$ does not encircle the +1 point. (We use the +1 point rather than –1 point because of the positive feedback sign convention). Since the condition, defined in equation B-1 ensures that the Nyquist diagram of $H(s)\delta(s)$ does not encircle the +1 point, it can be concluded that the closed-loop system is stable provided equation B.1 holds.

$$\sup_{\omega} \left| H(j\omega)\delta(j\omega) \right| < 1 \qquad \text{B.1}$$

Since $\delta(s)$ is unknown, it makes sense to replace equation B.1 with an alternative sufficient condition for stability in which $H(s)$ and $\delta(s)$ are separated, namely:

$$\sup_{\omega}|H(j\omega)|\sup_{\omega}|\delta(j\omega)|<1$$

If $\delta(s)$ is stable and bounded in magnitude (M), so that

$$\sup_{\omega}|\delta(j\omega)|=M$$

then the feedback loop given in Figure 52 will be stable provided a stabilizing controller can be found such that

$$\sup_{\omega}|H(j\omega)|<\frac{1}{M}$$

The quantity $\sup_{\omega}|H(j\omega)|$ satisfies the axioms of a norm, and is known as the *infinity norm*. Specifically,

$$\|L\|_{\infty}=\sup_{\omega}|L(j\omega)|$$

$\|L\|_{\infty}$ is the highest gain value on a Bode magnitude plot. The quantity $\|\cdot\|_{\infty}$ is a norm, since it satisfies the following axioms:

1. $\|L\|_{\infty}\geq 0$ with $\|L\|_{\infty}=0$ if and only if $L=0$.
2. $\|\alpha L\|_{\infty}=|\alpha|\|L\|_{\infty}$ for all scalars α..
3. $\|L+V\|_{\infty}\leq\|L\|_{\infty}+\|V\|_{\infty}$.
4. $\|LV\|_{\infty}\leq\|L\|_{\infty}\|V\|_{\infty}$

With this background of infinity norm, the optimal robust stability problem is formulated as one of finding a stabilizing controller $K(s)$ that minimizes $\left\|(1-G(s)K(s))^{-1}G(s)K(s)\right\|_{\infty}$. Note that $K(s)=0$ gives $\left\|(1-G(s)K(s))^{-1}G(s)K(s)\right\|_{\infty}=0$, and is therefore optimal, provided the system itself is stable. Thus, when the system is stable and there is no performance requirements other than stability, the optimal controller no need to feedback at all. When the system is unstable, setting $K(s)=0$ is not allowed. The optimal stability margin and the optimal controller are much harder to find.

B.1.2 Singular Values and the H_∞ Norm

Singular Value Decomposition (SVD) is an important tool in modern numerical algebra and numerical analysis. Owing to the linear algebraic nature of many control problems, like PSS, singular value decomposition has become an important part of control theory.

B.1.2.1 Properties of Singular Values

Let σ_j be the singular values of a complex matrix L of dimension $m \times n$. Let σ_j be the non-negative square-roots of eigenvalues of $L^* L$, where L^* denotes the complex conjugate transpose of L. σ_j is given by $\sigma_j(T) = \sqrt{\lambda_j(L^* L)}, \quad j = 1, \cdots, n,$ where $\lambda_j(\cdot)$ denotes eigenvalues, ordered such that

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_p \qquad p = \min\{m, n\}.$$

If $r < p$ then there are $p - r$ zero singular values, i.e.

$$\sigma_{r+1} = \sigma_{r+2} = \cdots = \sigma_p = 0.$$

There exist m x m and n x n matrices U and V, and a diagonal matrix Σ of dimension m x n, such that

$$L = U \begin{bmatrix} \Sigma_r & 0 \\ 0 & 0 \end{bmatrix} V^* \qquad \text{B.2}$$

where $\Sigma_r = diag(\sigma_1, \sigma_2, \cdots, \sigma_r)$.

Expression (B.2) is the singular-value decomposition of L.

The maximum and minimum singular values are denoted $\overline{\sigma}(L)$ and $\underline{\sigma}(L)$, respectively. That is $\underline{\sigma}(L) \leq |\lambda_j(L)| \leq \overline{\sigma}(L)$.

Some useful properties of singular values are given below. Note that x is a complex vector of dimension n and $\|x\| = (|x_1|^2 + \cdots + |x_n|^2)^{1/2} = \sqrt{(x^* x)}$.

$$\bar{\sigma}(L) = \max_{x \neq 0} \frac{\|Lx\|}{\|x\|} \qquad \text{B.3}$$

$$\underline{\sigma}(L) = \min_{x \neq 0} \frac{\|Lx\|}{\|x\|} \qquad \text{B.4}$$

$\underline{\sigma}(L) \leq |\lambda_j(L)| \leq \bar{\sigma}(L)$, where λ_j denotes the j-th eigenvalue of L.

If L^{-1} exists,

$$\underline{\sigma}(L) = \frac{1}{\bar{\sigma}(L^{-1})} \qquad \text{B.5}$$

$$\bar{\sigma}(L) = \frac{1}{\underline{\sigma}(L^{-1})} \qquad \text{B.6}$$

$$\bar{\sigma}(\alpha L) = |\alpha| \bar{\sigma}(L) \qquad \text{B.7}$$

where α is a real scalar.

B.1.2.2 Relationship between H_∞ Norm and Singular Values

The H_∞ norm can be determined directly from the singular value decomposition. The mathematical relationship is defined as:

$$\|L\|_\infty = \sup_\omega |L(j\omega)| = \max_\omega \bar{\sigma}(L(j\omega)) \qquad \text{B.8}$$

B.1.2.3 Physical Interpretation of the H_∞ Norm

The H_∞ design is formulated to minimize the energy of the output of the system for the worst disturbances of some pre-specified characteristics in the system. In the case of a single input single output system, the solution involves minimizing the H_∞ norm of the system transfer function, from the disturbance to the output over the set of all stabilising controllers.

The motivation of minimizing the H_∞ norm is best understood by giving a physical meaning to the H_∞ norm. Suppose $G(s)$ represents the transfer function of a stable system with input $u(t)$ and output $y(t)$. Let $u(t)$ be bounded in energy, that is:

$$Total\ Energy = \int_0^\infty u^T(t)u(t)dt\ is\ finite$$

Then the square-root of the maximum energy gain from input to output over all non-zero $u(t)$ is equal to the H_∞ norm of $G(s)$.

$$Max\,\bar{\sigma}(G(j\omega)) = \|G(j\omega)\|_\infty = Max_{u(t)\neq 0}\sqrt{\frac{\int_0^\infty y^T(t)y(t)dt}{\int_0^\infty u^T(t)u(t)dt}}$$

B.1.3 Disturbance Attenuation

The above interpretation of the H_∞ norm as maximum energy gain is particularly useful when studying the effect of uncertain disturbance signals. To illustrate the point, consider the SISO system shown below.

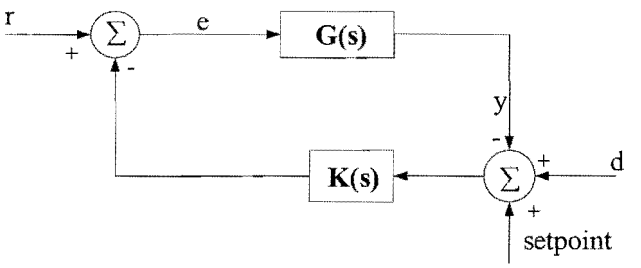

Figure 54 Feedback control system

The system output is required to stay as close to the set point despite the presence of an unknown disturbance *d*, confined to the frequency band $0 \le \omega \le \omega_d$. Let *S(s)* be the transfer function from the disturbance to the regulation error (set point minus output *y*). *S*(*s*) is therefore:

$$S(s) = \frac{1}{1 - G(s)K(s)} \qquad \text{B.9}$$

Minimizing the H_∞ norm of *S(s)* is equivalent to minimizing the square roots of the maximum possible energy in regulation error over all sets of possible disturbances. The resulting H_∞ controller, which minimizes the worst case of excursion of the output *y* resulting from any disturbance, stabilizes the overall system.

Equation B-9 represents an important function in control system design, namely the sensitivity function.

The design can be improved by limiting the minimization of the frequency range in which disturbances occur. The reason for limiting the minimization of the frequency range is that, by making the sensitivity small outside this frequency range will only lead to unnecessary noise amplification and poor stability margins. By including a stable minimum-phase weighting function $W_1(s)$ which is large over the frequency band of disturbances and small outside this band, a more stable controller will be obtained. Incorporating the weighting function into the sensitivity function, we obtain:

$$\min \left\| W_1(j\omega) S(j\omega) \right\|_\infty \qquad \text{B.10}$$

The H_∞ design can also significantly reduce the effects of model uncertainty. The feedback system of Figure 55 is used for the study of the system robustness. If the four elements of the matrix transfer function from *r* (input signal) and *d* (disturbance signal) to e (error signal) and *y* (output signal) are all stable, the system is internally stable. For example, the transfer function from *d* to *y* equals $[1 - G(s)K(s)]^{-1}$. The Nyquist criterion says that the feedback system is internally stable if and only if the Nyquist plot of *G(s)K(s)* does not encircle or pass through the point $s = -1$. Thus, a sufficient condition for internal stability is the small gain condition.

$$\|G(s)K(s)\|_\infty < 1 \qquad \text{B.11}$$

This implies that the open loop gain is always less than unity.

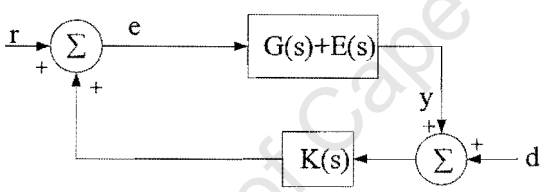

Figure 55 A feedback loop containing system uncertainty

To extend the idea of reducing the effects of model uncertainty to the problem of robust stabilization, we redraw the block diagram of Figure 54 to include the effects of model uncertainty as shown in Figure 55. Here we assume the actual system transfer function is $G(s) + E(s)$, where *E(s)* is an unknown perturbation representing parameter variations and unmodelled dynamics.

Now suppose the feedback system is stable for *E=0,* that is *K(s)* stabilizes *G(s).* How large can $|E(s)|$ be before the feedback system becomes unstable?

One method which is used to obtain a transfer function model is a frequency response experiment. This method yields gain and phase estimates at several frequencies, which in turn provide an upper bound for $|E(j\omega)|$ at several values of ω. Suppose $W_2(s)$ is a stable transfer function bounding the perturbation of *E(s)* in the sense that

$$|E(j\omega)| < |W_2(j\omega)| \quad for\ all \quad 0 \le \omega \le \infty \qquad \text{B.12}$$

Or equivalently

$$\left\| W_2^{-1} E \right\|_\infty < 1 \qquad \text{B.13}$$

How large can $|W_2|$ be, so that the internal stability is maintained?

A simple loop transformation of Figure 55 results in the block diagram in Figure 56 and Figure 57.

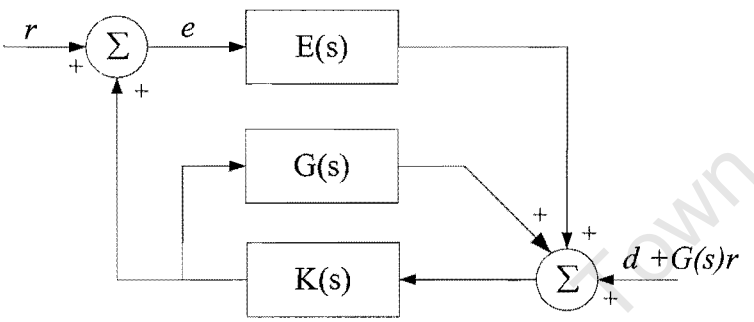

Figure 56 Loop transformation of Figure 55

Since *K(s)* stabilizes *G(s)*, therefore $K(I-GK)^{-1}$ is stable. The system of Figure 57 will be internally stable if the small gain condition of equation B.11 is satisfied, namely

$$\left\| EK(I-GK)^{-1} \right\|_\infty < 1 \qquad \text{B.14}$$

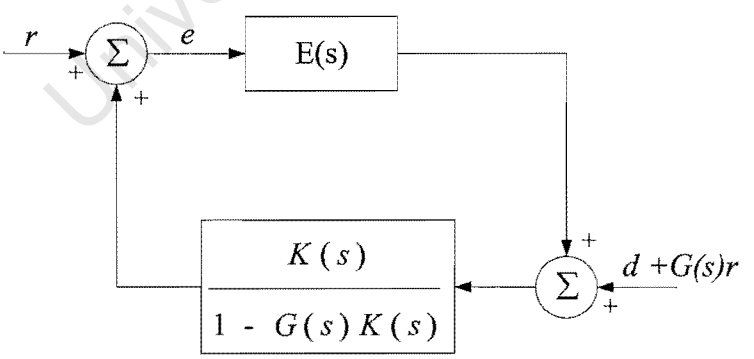

Figure 57 Loop transformation of Figure 56

In view of equation B.13 a sufficient condition for equation B.14 is

$$\left\| W_2 K (I - GK)^{-1} \right\|_\infty \le 1 \qquad \text{B.15}$$

Since if

$$\left\| W_2 K(I - GK)^{-1} \right\|_\infty \leq 1$$

Then from equation B.13 we have

$$\left\| W_2^{-1} E \right\|_\infty \left\| W_2 K(I - GK)^{-1} \right\|_\infty \leq 1$$

But

$$\left\| E W_2^{-1} W_2 K(I - GK)^{-1} \right\|_\infty \leq \left\| W_2^{-1} E \right\|_\infty \left\| W_2 K(I - GK)^{-1} \right\|_\infty \leq 1$$

Or

$$\left\| EK(I - GK)^{-1} \right\|_\infty \leq 1$$

Clearly, an H_∞ norm bound on a weighted closed-loop transfer function, i.e. equation B.15 is sufficient for robust stability.

One measure of system stability is sensitivity. It quantifies the advantages of feedback. To explain this concept, consider the system shown in Figure 58.

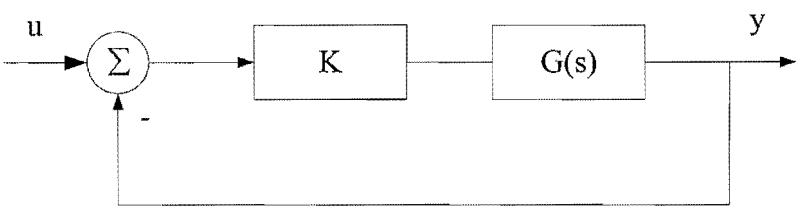

Figure 58 Closed loop for sensitivity comparision

The sensitivity for the open loop system with a gain of K is given as:

$$S_o = \frac{1}{K}$$

and for the closed loop system the sensitivity is given as:

$$S_c = \frac{1}{K(1+G(s)K)}$$

The ration of open loop to closed loop sensitivity is given as:

$$\frac{S_o}{S_c} = \frac{1}{1+G(s)K}$$

The feedback has the effect of reducing the sensitivity to gain variations by the reciprocal of the return difference $1+G(s)K$. The higher the return difference, the lower the sensitivity of the system. High return difference results in speeding up the dynamic response of the system and tends to immunize the system to changes in parameters of the open loop system. Sensitivity can be determined in terms of any parameter of the system, although the gain was selected for illustration purposes.

In principle, one wants the return difference to be as large as possible, although in practice this is not possible. The primary reason is that every practical system is essentially "low pass" in nature, tending to zero (magnitude) at infinite frequency. If the

amplifier in the above system has a constant gain K, the loop transmission will tend to zero at high frequencies, and hence the return difference will tend to be one. One could use a variable gain compensator to counteract the low-pass characteristic of the system, but this is not practical.

Thus, the objective of design is to ensure that the return difference converges to 1 in a graceful manner, rather than keeping the return difference as large as possible over all frequencies.

A reduction in the amplitude of the gain K is accompanied by phase-shift. It is possible for the gain to reach zero, and have a phase-shift of 180 degrees at some frequency. In this case, the return difference becomes zero and the transfer function of the system becomes infinite, i.e. the system becomes unstable. Thus, the return difference can never become zero at any frequency. It also impractical to allow in the design of the controller for the return difference to be near zero, as the application of the controller to the practical system may result in an unstable system.

Thus in the design it is necessary to provide reasonable stability margins. The two margins that are commonly used are gain and phase margins. The gain margin is the amount that the loop gain can be changed, at the frequency at which the phase-shift is 180 degrees, without reducing the return difference to zero. The phase margin is the amount of phase that can be added to the open-loop transfer function, at a frequency at which its gain is unity (0 dB). These margins are easily determined from Bode and Nyquist plots.

The calculation of gain and phase margins in a single-input-single-output (SISO) system is easy, but in a multi-input-multi-output (MIMO) system it is more complex. The return difference is critical in the determination of robustness in the MIMO system as well.

Three weighting functions $W_1(s)$, $W_2(s)$, and $W_3(s)$, described below, define the designing specifications in H_∞ control theory.

The weighting function $W_1(s)$ is related to the performance objective of the error sensitivity function S(s). One may intuitively select a high-gain low-pass filter to reduce the error sensitivity in the low frequency range. However, this selection results in cancellation of the plant's poles by controller's zeros. This makes the poorly damped mode unobservable at the chosen outputs of the plant, but does not improve the damping of the mode as observed from other outputs of the actual system. In power systems, this is not an acceptable solution. Our aim in design the controller is to enhance damping and not just to eliminate the oscillations from one particular output of the system.

H_∞ algorithm produces a controller whose zeros consist of the open loop stable poles of the plant and whose poles contain the poles of $W_1(s)$. We can therefore take advantage of this fact and include the critical poles of the plant within the poles of $W_1(s)$. With this choice of $W_1(s)$, the final controller no longer has the critical poles of the plant as its zeros and hence the pole-zero cancellation is prevented.

The weighting function $W_2(s)$ can be considered as the variation in the plant model due to changes in operating conditions. It is incorporated into the design to guarantee the stability of the controlled system under diverse operating conditions.

Weighting function $W_3(s)$ is applied to the plant output to ensure satisfactory performance of the closed-loop system at high frequencies. This weighting function is necessary since the plant model approximates the actual system, and high frequency dynamics have been neglected because of our emphasis on low frequency oscillations.

B.2 H_∞ Control Design Algorithm

The following are the main steps in the H_∞ control design algorithm:

Step 1: An augmented transfer function matrix *P(s)*, which describes the system *G(s)* with the design constraints expressed in terms of weighting functions $W_1(s)$, $W_2(s)$ and $W_3(s)$, as shown in Figure 59, is defined

$$P(s) = \begin{bmatrix} W_1 & -W_1 G \\ 0 & W_2 \\ 0 & W_3 G \\ I & -G \end{bmatrix} \qquad \text{B.16}$$

Step 2: A stabilizing controller *K(s)* such that the closed-loop transfer function $T_{u_j y_j}$ is internally stable and its infinity norm is less than or equal to one is found. The process involves solving the Riccati equation [1-3] and performing the optimization to satisfy the inequality constraint on *T*:

$$\left\| T_{u_j y_j} \right\|_\infty = \left\| \begin{matrix} W_1 S \\ W_2 R \\ W_3 T \end{matrix} \right\|_\infty \leq 1 \qquad \text{B.17}$$

There transfer functions are defined:

$$S(s) = \frac{1}{1 - K(s)G(s)}$$
$$R(s) = \frac{K(s)}{1 - K(s)G(s)}$$
$$T(s) = \frac{K(s)G(s)}{1 - K(s)G(s)}$$

Where S(s) is the Sensitivity transfer function, R(s) is Related Additive Robustness transfer function and T(s) is Complimentary Sensitivity transfer function.

Step 3: If a solution to the H_∞ control problem does not exist for the specified *P(s)*, the design constraints need to be redefined and the process restarted from Step 1.

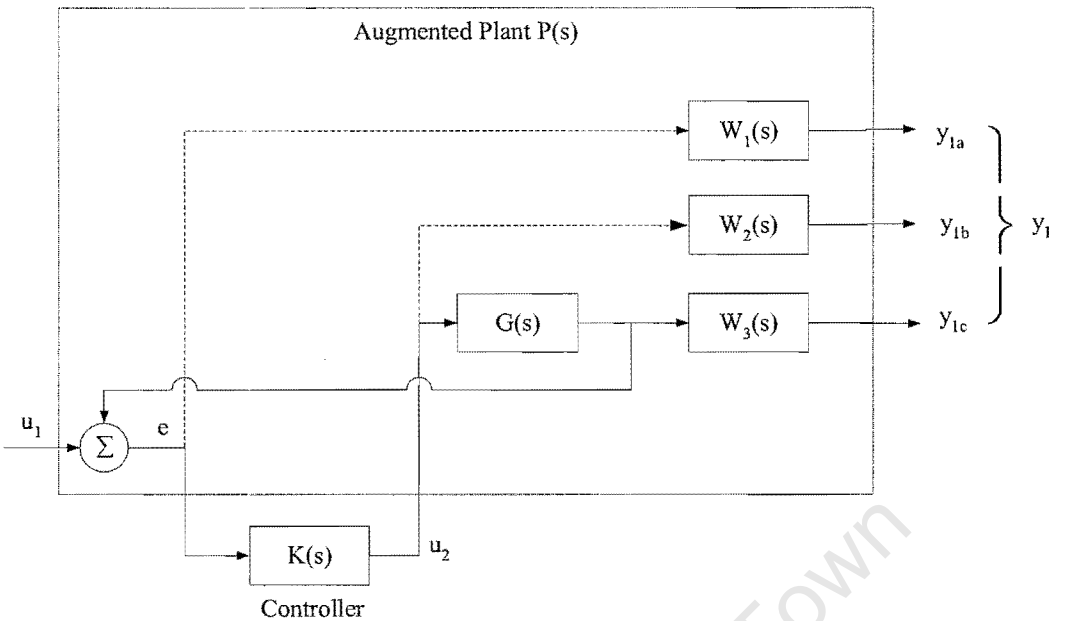

Figure 59 Augmented plant

B.3 References

[1] K. Dutton, et al "The Art of Control Engineering", Addison-Wesley, 1997

[2] B. Friedland, "Control System Design", McGraw-Hill , 1987

[3] M. Green, D. Limebeer "Linear Robust Control", Prentice Hall, 1995

[4] MATLAB Robust Control Toolbox Manual.

[5] P. Kundur, "Power System Stability and Control", Prentice-Hall, 1993

[6] M. Klein, et al., "H_∞ Damping Controller Design in Large Power Systems", IEEE Trans. on Power Systems, Vol. 10, No. 1, February 1995, pp158-166

Appendix C

Population Based Incremental Learning

C.1 Brief Introduction to PBIL

Population Based Incremental Learning is a combination of Genetic Algorithms and Competitive Learning.

C.1.1 Brief Introduction to Genetic Algorithms

Genetic Algorithms (GAs) are biologically motivated adaptive systems that are based upon the principles of natural selection and genetic recombination. Many of the current efforts in GA research have stressed their role as general purpose function optimizers, which have been applied to functions in the fields of biological modeling and standard numerical optimization. Although GAs may have the ability to quickly find regions of high performance in the presence of noise and time-varying payoff functions, they may be unable to find the absolute optimal solution, in time-varying or stationary environments. [1-3]

In GAs, candidate solutions to a problem are analogous to individuals in a population. A population of individuals is maintained within the search space for the GA, each representing a possible solution to a given problem. The initial population can be a random collection of different individuals. The individuals will interact and breed to form future generations. The stronger individuals will reproduce more often than the weaker individuals. Presumably, the population will get collectively stronger as generations pass and weaker individuals die out. Unlike other optimization methods, GAs do not limit by constraints in the form of fitness functions. The fitness function does not need to be differentiable or continuous. This flexibility in which GAs use a fitness function to search for the solution makes GAs a powerful tool for optimization in many difficult problems across many fields.

GAs work with coding of the parameters themselves and then use the genetic operators to evolve the solution with minimum computation. An optimal solution can be found and represented by the final winner in the competitive environment. GAs consist of three simple operators: selection, crossover and mutation.

Selection is the operation, in which the fittest individual of the population in the current generation forms part of the population in a new generation. These potential solutions are called "chromosomes".

Crossover is responsible for providing new future generation by selecting two individuals and exchanging some parts of their structures. The chromosomes with high fitness values will have a higher probability of being selected for recombination than those that do not. The "children" chromosomes produced by the genetic recombination are not necessarily better than their "parent" chromosomes. Nevertheless, because of the selective pressure applied through a number of generations, the overall trend is towards improved chromosomes. Recombining only good chromosomes will quickly converge the population without extensive exploration, thereby increasing the possibility of finding only a local optimum.

Mutation is an operator, which is applied for altering the value of a random position in a parameter in order to maintain diversity and to escape from local optima. Mutations introduce random changes into the population.

GAs are typically allowed to continue for fixed number of generations. At the conclusion of the specified number of generations, the best chromosome in the final population, or the best chromosome ever found, is given.

In using GAs for function optimization, many issues, such as proper scaling of functions, ensuring that good information is not lost due to random chance, and efficient problem representation, need to be solved. Although GAs can often find regions of high performance, it is much harder for the GAs to select the global optimal solution. One potential reason for this inability is that the differential between good and optimal solutions may be very small in comparison with the differential between good and bad solutions. In designing an effective genetic based function optimizer, it is necessary to be able to provide a large enough "incentive" for the GAs to make progress given only small differentials. One method of maintaining a large differential between potential

solutions is to employ a method of dynamic scaling, so that the fitness of each solution is measured relative to the fitness of the other solutions in the current population.

As GAs are randomized, it is possible to lose the best solution due to random chance. There is no guarantee that the best solution in the current population will be selected for recombination, or that if it is selected, that the mutation and crossover operators will not destroy some of its information as it is passed to its successors. If the best solution is lost from the population, there is no guarantee that the solution will be found again. Methods such as elitist strategies have been proposed to address this problem. Elitist strategies ensure that the best solution in the previous population is transferred to the current generation by replacing the worst chromosome in the current generation with the best from the previous generation.

Beyond the mechanisms inherent to the GAs that influence its ability to optimize functions, there is also the issue of problem representation. Although the majority of GAs research has been conducted using a binary solution representation, this is not the only method of encoding problem solutions. Different methods are used for encoding or for the interpretation of the encoding. These methods which alter the cardinality of the alphabet, can have an enormous impact on the performance of GAs.

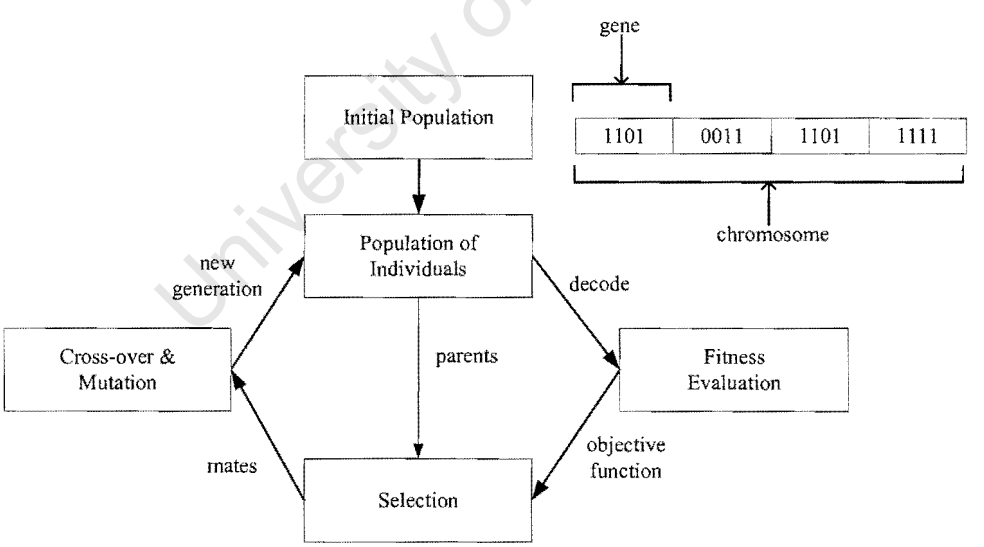

Figure 60 Simple algorithm flow-chart of GAs

In every generation during the run of a GA, a population of potential solutions exists. The search of the function space progresses from these points in parallel with the schemata represented in these points. This is in contrast to other search technique, such as hill-climbing. (The Hill-climbing method introduces a random change in the solution of a problem by comparing the new solution with the old one. If the new solution is better than the old one, then the new solution is accepted as the basis for a new random change. If not, the old solution remains the basis for the next iteration. The optimization process is finished after a prescribed number of iterations.) The ability to search multiple schemata in each solution vector has been termed implicit parallelism. However, useful parallelism, at the level of the population, is not easily maintained. It is possible for the population to converge to very similar solution vectors. Once the population has converged, the ability for crossover operators to aid in exploring new portions of the function space is greatly hindered. Premature convergence of a population can occur when the population becomes too homogenous. As the GA allocates an exponentially increasing number of trials to improve solutions, the entire population may come to be dominated by very similar solution vectors when several consecutive generations do not develop novel high evaluation solution vectors.

In a traditional GA, the problem of premature convergence and the trap of local minima have been partially addressed by the mutation operator. However, other mechanisms, which help to maintain the parallelism explicitly, have been proposed to address the problem of diversity loss and maintaining parallelism in search. To demonstrate the importance of maintaining parallelism in genetic search, the techniques presented here to implement explicit parallelism is sub-population evolution.

One method of implementing explicit parallelism is through models of genetic algorithms often referred to as "island models" or "coarse/fine grain parallel GA's" etc. The underlying premise of these models is that although genetic search often loses the parallelism inherent in a single large population structure, it is possible to maintain parallelism using multiple sub-populations. In this model, the single large population of the traditional genetic algorithm is divided into many smaller sub-populations. Each sub-population evolves its chromosomes primarily independently of the other sub-

populations. Interaction, in the form of chromosome swapping between sub-populations, is restricted, and is based upon temporal and spatial considerations.

C.1.2 Brief Introduction to Competitive Learning

Competitive learning (CL) is often used to cluster a number of unlabeled points into distinct groups. Membership into each group is based upon the similarity of points with respect to the characteristics in study. The procedure is unsupervised, as there is no a priori knowledge of how many groups exist, or what each group's distinguishing features may be. The hope is that the CL procedure will be able to determine the most relevant features for class formation and then be able to cluster points into distinct groups based on these features.

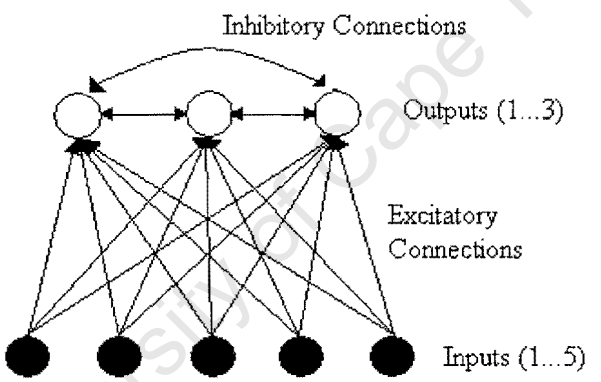

Figure 61 A competitive learning network

Competitive learning is often studied in the context of artificial neural networks as it is easily modeled in this form. A typical competitive learning network is shown in Figure 61. The inputs correspond to the feature vector for each point. The outputs correspond to the class in which the network had placed the point. In this network, there are two types of connections, excitatory and inhibitory. The inhibitory connections, between output units, ensure that only one output is turned on at a time. The output unit that is turned on is the one that has the largest net input. The excitatory connections contribute to the net input of the outputs. The algorithm used to train the network is described below.

The initial weights of the network are chosen randomly, and are subject to normalization constraints. The activation of the output units is calculated by the following formula (in which ω is the weight of the connection between *i* and *j*):

$$output_i = \sum_j \omega_{ij} \times input_j$$

In a competitive learning network, only the output unit with the largest activation is allowed to fire for each point presented. The winning output unit corresponds to the classification of the input point. During training, the weights of the winning output unit are moved closer to the presented point by adjusting the weights according to the following rule (LR is the learning rate parameters):

$$\Delta\omega_{ij} = LR \times (input_j - \omega_{ij})$$

The process of training the CL network involves repeatedly presenting each point to the network until the network has stabilized. Although, in general, the network cannot be guaranteed to stabilize if weight updates are made after each point is presented to the network, other heuristics may be used to determine when to stop training. One possible method of ensuring stability is to reduce the learning rate gradually to 0, as the total number of pattern presentation increases.

After the network training is complete, the weight vectors for each of the output units can be considered as prototype vectors for one of the discovered classes. The attributes with the large weights are the defining characteristics of the class represented by the output. It is the notion of creating a prototype vector that will be central to the discussions of PBIL. The unsupervised nature of the algorithm will not be maintained, as the members of the class of interest will be easily determinable; this will be returned to in much detail in the next section. Although the method of training described in this section has been unsupervised, supervised competitive learning has been explored in the artificial neural network literature, and is central to the discussion in the next section as well. The examination of PBIL through the perspective of supervised competitive learning yields insights that can lead to a much more efficient algorithm than the base-line PBIL described in the next section. Improved versions of PBIL are described later as well and are empirically examined.

C.1.3 Examining the PBIL

C.1.3.1 The Role of a Population in PBIL

One of the fundamental attributes of the genetic algorithm is its ability to search the function space from multiple points in parallel. In this context, parallelism does not refer to the ability to parallelize the implementation of genetic algorithms; rather, it refers to the ability to represent a very large number of potential solutions in the population of a single generation. This section describes the implicit parallelism of PBIL, and explicit methods of ensuring parallelism. Because of the failure of the implicit parallelism to exist in the latter parts of genetic search in the simple genetic algorithm, the utility of maintaining multiple points, through the use of a population, decrease. The limited effectiveness of the population in the latter portions of search allows it to be modeled by a probability vector, specifying the probability of each position containing a particular value. This concept is central to the PBIL algorithm.

C.1.3.2 Replacing the Population in PBIL

The PBIL algorithm attempts to create a probability vector from which samples can be drawn to produce the next generation's population. As in standard GAs, it is assumed that the solution is encoded into a fixed length vector. Ignoring the contribution of mutation, the expected contribution of values in each position of the population during generation *G*, can be computed based upon the population of generation *G-1*. Assuming a fully generational GA (each member of the population is replaced in the subsequent generation), fitness proportional selection, and a general pair-wise recombination operator, the probability of value *j* appearing in position *i* in a solution vector *x*, in a population at generation *G*, can be computed as follows:

$$P(i,j) = P(x_i = j) = \frac{\sum_{v \in Population_{G-1} \wedge v_i = j} EvaluateVector(v)}{\sum_{v \in Population_{G-1}} EvaluateVector(v)}$$

This is simply a counting argument, weighted by the evaluation of each solution string. Given a population, a unique representation can be made by a probability matrix defined by the above equation. Two comments should be made about this representation. First, many populations will have identical probability matrices. Second, if the above representation was used to represent the population, and samples for generation *G* were drawn by sampling points based upon this distribution, the solution vectors produced are unlikely to improve over those in generation *G-1*. This is due to the implicit assumption that each bit position's value is independent of all other bit position's values across individual solution vector. Traditional crossover does not assume this; a pair-wise crossover operator maintains more information between the bit positions, as the composition of the "children" solution strings is chosen from only two "parent" solution strings.

Although a population represented directly as mentioned above may not be useful for sampling in order to generate the next generation's vectors, a variation of the above representation can be useful. From a population of size *N*, consider probabilistically (based upon fitness) selecting *N* solution vectors for recombination. These newly generated vectors can be represented as a probability matrix by simply counting the number of occurrences of each value in each bit position. This probability matrix can be used to create a new population. This representation has been used to simulate crossover.

In a manner similar to the methods described above, the population-based incremental learning (PBIL) algorithm uses a probability vector to describe the population of a genetic algorithm. In a binary encoded solutions string, the probability vector specifies the probability of each bit position containing a '1'. The probability of the bit position containing a '0' is obtained by subtracting the probability specified in the vector from 1.0. For example, see Figure 62.

Although from this point, PBIL will be considered for solution vectors encoded in a binary alphabet, this method can be applied to higher-cardinality representations.

Population #1	Population #2	Population #3
0011	1010	1010
1100	1100	0101
1100	1100	1010
0011	1100	0101
Representation	Representation	Representation
0.5 0.5 0.5 0.5	1.0 0.75 0.25 0.0	0.5 0.5 0.5 0.5

Figure 62 Probability representation of three small populations of four bit solution vectors

The size of the population is four. Notice that the first and third representation for the population are the same, although the solution vectors each represents, are entirely different.

The PBIL algorithm attempts to create a probability vector that can be considered a prototype for high evaluation vectors for the function space being explored. A very basic observance of genetic algorithm behavior provides the fundamental guidelines for the performance of the PBIL. One of the key features in the early phase of genetic optimization is the parallelism in the search; many diverse points are represented in the population of a single generation. In representing the population of a GA in terms of a probability vector, the most diversity will be found in setting the probabilities of each bit position to 0.5. This specifies that generating a 0 or 1 in each bit position is completely random.

The PBIL algorithm uses the probability vector representation for defining a population. Rather than passively transforming each population into a probability vector, from which solution vectors are generated and recombined, etc., the aim of PBIL is to actively create a probability vector, which, with high probability, represents a population of high evaluation solution vectors. In PBIL, unlike the mechanisms inherent to a GA, operation takes place directly on the probability vector. This mechanism is derived from those used in competitive learning.

In a manner similar to the training of a competitive learning network, the values in the probability vector are gradually shifted towards representing those in high evaluation vectors. A simple procedure to accomplish this shifting is described below. The probability update rule, which is based upon the competitive learning update rule, is shown below.

$$P_I(k) = (1 - LR) * P_I(k-1) + LR * \max_I(k-1) \qquad \text{C.1}$$

Where:

$P_I(k)$	The probability of generating a "1" at the I-th bit position at generation k.
$\max_I(k)$	The I-th position in the solution vector to which the probability vector is moved at generation k.
LR	The learning rate

The probability update rule, described above, is similar to the weight update rule in a competitive learning network when an output moves towards a particular sample point. Each bit is examined independently. In equation C.1, representing each bit independently disregarded much of the information which standard crossover preserved. The reason assumption of independence is not detrimental, is that PBIL does not attempt to represent the entire population by the probability vector. Rather, PBIL represents a single point, based upon the vector with the highest evaluation, around which the next population of point should be generated.

The step, which remains to be defined, is determining which solution vectors to move towards. If the good vectors were known a priori, the problem would, of course, already be solved. Several alternative methods have been explored. The basis of some of the earlier attempts involved generating a single vector, and deciding whether to push the probability vector towards the generated vector. However, a more effective method, which has proven empirically to be more resistant to getting caught in local minima, is to generate a number of vectors, all based upon the probabilities specified in the probability vector, and to push the probability vector towards the generated vector with the highest evaluation. After the probability vector is updated, a new set of vectors is produced based upon the updated probability vector, and the cycle is continued.

Generating a population of potential solutions, rather than a single solution vector, is an attempt to maintain the ability to explore large regions of space in a paralleled manner, as the GA does in the early stages of search. In the early stages of search, there is a large amount of diversities in the regions of the function-space, which are simultaneously explored. In PBIL the values in the probability vector move away from

0.5, towards either *0.0* or *1.0*. In the GA, this corresponds to the respective bit positions in the majority of the solution strings having the same value. As the search progresses, the population of the GA tends to converge around a good solution vector in the function space. Analogously to genetic search, in PBIL the population converges around a single point. The PBIL algorithm and a standard GA face the same problem of premature convergence. In PBIL, as the probabilities become very close to either *0.0* or *1.0*, the similarity in the vectors generated increases. One advantage, which the PBIL algorithm offers, is explicit control of the speed at which the population converges. The setting of the learning rate parameter can greatly affect the speed towards convergence.

A concern is that, because only a single probability vector is used, it may have less expressive power than a full population GA. A GA, which uses a population of points, can represent a large number of points simultaneously. For example, in Figure 62, the representations for population *#1* and population *#3* are represented by probability vectors of *0.5*; therefore, it is unlikely that sampling the probability vector would regenerate the population members. This appears to be a fundamental limitation of PBIL, as it is possible for a GA to contain either of these populations. However, for the reasons mentioned previously (genetic drift), in simulating genetic search, a traditional single population GA would not be able to maintain either of these populations. Because of sampling errors, the population will converge to one point; it will not be able to maintain multiple dissimilar points. Therefore, even if the members of the populations shown in Figure 62 had equally high evaluations, the GA would be unable to maintain them in its population, and would converge to only one. Similarly, in PBIL, the values of *0.5* will quickly be changed to favor either *0.0 or 1.0* through the search's progression; the probability vector can only represent one of the dissimilar points. There are several methods have been developed to address this problem for both GA and PBIL.

As the population, which is represented by a probability vector, is not unique, this aids in maintaining diversity in search. For example, the probability vector used to represent these populations can generate members of population #1 and population #3. In traditional GAs, uniform crossover also often has the same characteristics from one generation to the next – extremely dissimilar children can be produced from the same parents. GA does not rely on crossover to perform extensive search. Mutation plays a prominent role.

C.1.3.3 The Probability Vector in PBIL

The probability vector maintained by the PBIL can be viewed as a prototype vector for generating solution vectors which have high evaluations with respect to the available knowledge of the function space. In each generation, the probability vector is adjusted to represent the highest current evaluation vector. As values in the bit positions become more consistent between the highest evaluation vectors produced in subsequent generations, the probabilities of generating the value in the bit position increases. In PBIL, the class of high evaluation vectors is defined during the algorithm's search. In each population of points generated, the highest evaluation vector produced is defined to be in the class of interest. It should also be noted that the probability vector not only specifies the prototype based upon the high evaluations of the sample solutions, but also guides the search, which produces the next sample point from which to "learn".

C.1.3.4 The role of Mutation in PBIL

As mentioned previously, mutation plays an important role to avoid local optimum. For PBIL, there are two ways of defining a mutation operator. The first is to perform the mutation directly on the vectors generated. The second method is to perform a mutation on the probability vector; this mutation can be defined as a small probability perturbation on each of the positions in the probability vector. Both of these forms of mutation have the same effect as mutation in the standard GA: to preserve diversity. In principle, either could be selected, but the second is the preferred choice.

It is important to understand the role of operators used in GAs to determine what their relevance is to the PBIL algorithm, and what their role should be. Crossover is valuable in the early portion of search, as it takes larger steps towards better solutions. In many optimization problems, much of the fine-tuning in genetic search occurs because of the mutation operator, as recombining similar chromosomes exclusively through the use of crossover does not introduce enough diversity or induce exploration of the function space.

Through experimentation, it has been shown that using mutation improves the performance of the algorithm. In the case of PBIL, the mutation operator is not as important as in GAs. Essentially, in PBIL, the mutation operator prevents the prototype vector from converging too quickly to an extreme value in each of the bit positions.

C.1.3.5 The Learning Rate in PBIL

There are four parameters, which can be adjusted in the PBIL algorithm, namely the population size, the learning rate, the mutation probability and the mutation shift. The learning rate parameter does not have an equivalent in GAs. The learning rate has a dramatic influence in PBIL, since it determines how fast the prototype vector is shifted to resemble a correctly classified point. The larger the learning rate, the less the function space will be searched. This implies that for small learning rates convergence may be slower. The learning rate should not be set too small, however.

C.2 PBIL Example

The following example illustrates how the PBIL algorithm works for a single unknown parameter.

Problem:

For the function $y(x) = 1 - x^2$, find the value of x that maximizes y.

Given PBIL Parameters:

Learning Rate = 0.1

Mutation Probability = 0.02

Mutation Rate = 0.05

Population Size = 5

Generations = 11

Bits per parameter (precision) = 6

Initial probability vector:

P_i=[0.5 0.5 0.5 0.5 0.5 0.5];

Generation 1:

Using the probability vector, generate the population:

x (base2)	x (scaled)	y	Best x	Best y
011101	4.5313e-001	7.9468e-001	011101	7.9468e-001
010101	3.2813e-001	8.9233e-001	010101	8.9233e-001
101001	6.4063e-001	5.8960e-001	010101	8.9233e-001
101111	7.3438e-001	4.6069e-001	010101	8.9233e-001
100000	5.0000e-001	7.5000e-001	010101	8.9233e-001

The best solution in this generation is: 010101.

The non-mutated probability vector for generation 2 is:

[0.55 0.45 0.45 0.55 0.55 0.55]

Generation 2:

Using the probability vector from generation 1, a new value for x is generated.

x (base2)	x (scaled)	y	Best x	Best y
011000	3.7500e-001	8.5938e-001	010101	8.9233e-001
010110	3.4375e-001	8.8184e-001	010101	8.9233e-001
011011	4.2188e-001	8.2202e-001	010101	8.9233e-001
011011	4.2188e-001	8.2202e-001	010101	8.9233e-001
001111	2.3438e-001	9.4507e-001	001111	9.4507e-001

The best solution in this generation is: 001111.

The non-mutated probability vector for generation 3 is:

[0.21523 0.26306 0.33399 0.38182 0.33399 0.48434]

Generation n:

The procedure is repeated for each generation. If in a generation no solution is generated that is better than the best solution from all previous generations, then the best solution from the previous generations is carried forward.

Generation 9:

x (base2)	x (scaled)	y	Best x	Best y
001011	1.7188e-001	9.7046e-001	000001	9.9976e-001
010101	3.2813e-001	8.9233e-001	000001	9.9976e-001
001101	2.0313e-001	9.5874e-001	000001	9.9976e-001
110000	7.5000e-001	4.3750e-001	000001	9.9976e-001
000101	7.8125e-002	9.9390e-001	000001	9.9976e-001

No best solution was generated for this population group, so the best solution from a previous generation was kept. The non-mutated probability vector for generation 10 is:

[0.19371 0.23676 0.30059 0.34364 0.30059 0.43591]

Generation 10:

x (base2)	x (scaled)	y	Best x	Best y
010110	3.4375e-001	8.8184e-001	000001	9.9976e-001
000011	4.6875e-002	9.9780e-001	000001	9.9976e-001
100001	5.1563e-001	7.3413e-001	000001	9.9976e-001
000010	3.1250e-002	9.9902e-001	000001	9.9976e-001
000000	0	1	000000	1

In this generation, the best solution is 000000. Although this is the solution to the optimization problem, we need to go through additional generations to make sure that the results converge to the best solution. The non-mutated probability vector for generation 11 is:

[0.17434 0.21308 0.27053 0.30927 0.27053 0.39232]

Generation 11:

x (base2)	x (scaled)	y	Best x	Best y
000000	0	1	000000	1
000000	0	1	000000	1
000000	0	1	000000	1
000000	0	1	000000	1
000001	1.5625e-002	9.9976e-001	000000	1

In this generation, the best solution from the previous generation is generated for each population (except the last one). This localized convergence is indicative of having found the solution to the optimization problem. However, it is critical that global convergence (i.e. over a number of generations is achieved) is achieved to declare that the optimal solution has been found.

The final solution to the problem is: x=0 and y=1. If one were to solve the equation mathematically the same solution would be found.

C.3 References

[1] D.E. Goldberg, "Genetic Algorithms in Search, Optimization and Machine Learning" Addison Wesley, New York, 1989

[2] S. Baluja, "Population Based Incremental Learning" CMU-CS-94-163

[3] K. Hongesombut et. al. "An Incorporated Use of Genetic Algorithn and Modelica Library for Simutaneous Tuning of Power System Stabilizers", 2nd International Modelica Conference, Proceedings, Japan, March 2002, pp.89-98

Appendix D

Chapter 2 Case Study

D.1 Plant

The following table defines the generator data for the SMIB plant under consideration.

Synchronous Generator Data MVA base = 300			
T'_{do}=1.125	X'_d=0.36	T''_{do}=0.0216	T''_{qo}=0.09
X_d=2.72	X''_d=0.26	X_q=2.6	X''_q=0.26
R_a=0.0		H=3.84kWs/KVA	

The AVR has the following transfer function: $AVR(s) = 50/(1 + s0.05)$

The following are the state-space matrices for the SMIB plant under consideration:

A=[
-3.64450 0.00651 3.11980 -0.11680 0 0.08373 0 0.18553
0.19651 -1.2531 -0.6533 0.14284 0 -0.01993 0 0
19.92700 0.01872 -22.343 -0.35182 0 -1.44010 0 0
-0.42633 7.82170 1.399 -14.9000 0 4.34740 0 0
0.04578 -0.00329 -0.1507 0.05956 -0.0774 -0.11276 0 0
0 0 0 0 376.99 0.00000 0 0
0.02292 0.00140 -0.0753 -0.02540 0 0.00724 -0.1000 0
50.94300 3.11400 -167.42 -56.4340 0 16.09400 2000.00 -20.000
]

B=[
0 0
0 0
0 0
0 0
0 0.0782
0 0
0.0900 0
200.00 0
]

C=[
-0.25471 -0.01557 0.83708 0.28216 0 -0.08047 0 0
-0.58558 0.04233 1.9287 -0.76187 0 1.4425 0 0
]

D=[
0 0
0 0
]

D.2 Weighting Functions

The weighting functions are defined as:

$$w_1(s) = \frac{0.01(s+70)^2}{(s+0.2199)^2}$$
$$w_2(s) = 0.001$$
$$w_3(s) = \frac{100(s+2.268)^2}{(s+226.757)}$$

D.3 SQP-based Sub-optimal H_∞ design/tuning Method Matlab Script

D.3.1 Augmentation Script

```
% plant.m contains the state-space matrices of the plant
load plant.m;

% Specify the weighting functions

numw1=1000*conv([0.0144 1],[0.0144 1]);
denw1=conv([4.547 1],[4.547 1]);

numw3=0.01*conv([0.441 1],[0.441 1]);
denw3=conv([0.00441 1],[0.00441 1]);


w1=[numw1;denw1;numw1;denw1];
w2=[0.001;1;0.001;1];
w3=[numw3;denw3;numw3;denw3];

% Augment the plant

[AA,B1,B2,C1,C2,D11,D12,D21,D22] = augtf(A,B,C,D,w1,w2,w3);

save augplant.mat;
```

D.3.2 Objective Function

This function is called by Matlab's "constr" function. It selects values for the unknown controller parameters and calculates the maximum singular value and the minimum damping for the closed loop response.

```
% objfunc.m
function [f,g]=objfunc(x)

% load the augmented plant
load augplant.mat;

% define the frequency range for evaluation
w=logspace(0.01,1.5,200);

% define PSS structure
connum=x(1)*[1 x(2) x(3)];
conden=[1 x(4) x(5)];
[E,F,G,H]=tf2ss(connum,conden);

%create closed loop matrices
AT=[(AA+B2(:,1)*H*C2(2,:)) (B2(:,1)*G)
    (F*C2(2,:))             E        ];
BT=[(B2(:,1)*H*D21(1,:))+B1
     F*D21(1,:)              ];
CT=[(C1+D12(:,1)*H*C2(2,:)) (D12(:,1)*G)];
DT=[(D12(:,1)*H*D21(1,:))+D11];

%create objective function and calculate singular value
[T12]=max(sigma(AT,BT(:,1),[CT(2,:);CT(6,:)],[DT(2,1);DT(6,1)],w))
;
SV=[T12];

% return the maximum singular value
f=max(SV)

% calculate the damping
[x,y]=damp(AT);
g=min(y)

% set the damping constraint (requirement of matlabs constr
function)
g(1)=-g+0.274;
```

D.3.3 Optimization Script

The following script calculates the PSS parameters by using SQP constrained optimization.

```
% Set the number of iterations
options(14)=1000;

% Define an initial starting point
x0=[1 ;12; 23; 12 ;2.3];

% Construct the SQP optimization function
% x contains the unknown PSS parameters

[x,options]=constr('objfunc', x0, options,[0],[20])

% Validate that the solution converges, otherwise increase number
% of iterations. If not, adjust the starting point or reduce the
% damping constraint defined in the objective function. The
% [0] and [20] above is the constraint on the gain x(1).
```

D.4 Optimal H_∞ design/tuning Method Matlab Script

The optimal H_∞ design was done using Matlab's Robust Controller Toolbox. The toolbox provides a standard set of procedures to design Optimal H_∞ controllers.

D.4.1 Matlab Script

The following Matlab script is used to calculate the optimal H_∞ controller.

```
% plant.m contains the state-space matrices of the plant
load plant.m

% Extract the T12 transfer function from the plant for purposes of
% comparison to SQP example
B = B[:,1];C = C[2,:];D = D[1,2];

% Specify the weighting functions
numw1=1000*conv([0.0144 1],[0.0144 1]);
denw1=conv([4.547 1],[4.547 1]);
numw3=0.01*conv([0.441 1],[0.441 1]);
denw3=conv([0.00441 1],[0.00441 1]);
w1=[numw1;denw1];
w2=[0.001;1];
w3=[numw3;denw3];

%Augment the plant
[Aa,Ba1,Ba2,Ca1,Ca2,Da11,Da12,Da21,Da22] =
augtf(A,B,C,D,w1,w2,w3);
D11q=Da11;D12q=Da12;D22q=Da22;D21q=Da21;

% Build a balanced system using OBALREAL
[aa,bb,cc,mm,tt] = obalreal(Aa,[Ba1 Ba2],[Ca1;Ca2]);
Aq = aa; B1q = bb(:,1); B2q = bb(:,2); C1q = cc(1:3,:); C2q =
cc(4,:);
sys_ = mksys(Aa,Ba1,Ba2,Ca1,Ca2,Da11,Da12,Da21,Da22,'tss');

%Determine the optimal Hinf optimal PSS
[gamm,ss_cp,ss_cl]=hinfopt(sys_,1,[0.0001 1.5 0]);

% Controller
[Ac_hinf,Bc_hinf,Cc_hinf,Dc_hinf]=branch(ss_cp);

% Closed Loop Plant
[Acl_hinf,Bcl_hinf,Ccl_hinf,Dcl_hinf]=branch(ss_cl);
```

D.4.2 Output from Optimal H_∞ Design

The following table is the output generated by the MATLAB function "hopt". Based on the specified tolerance, iteration number 25 is deemed by Matlab to be the best answer.

No	Gamma	D11<=1	P-Exist	P>=0	S-Exist	S>=0	lam(PS)<1	C.L.
1	1.5000e+000	OK	FAIL	FAIL	OK	OK	OK	UNST
2	7.5000e-001	OK	FAIL	FAIL	OK	OK	OK	UNST
3	3.7500e-001	OK	FAIL	FAIL	OK	OK	OK	UNST
4	1.8750e-001	OK	FAIL	FAIL	OK	OK	OK	UNST
5	9.3750e-002	OK	FAIL	FAIL	OK	OK	OK	UNST
6	4.6875e-002	OK	FAIL	FAIL	OK	OK	OK	UNST
7	2.3438e-002	OK	FAIL	FAIL	OK	OK	OK	UNST
8	1.1719e-002	OK	FAIL	FAIL	OK	OK	OK	UNST
9	5.8594e-003	OK	FAIL	OK	OK	OK	OK	STAB
10	2.9297e-003	OK	FAIL	FAIL	OK	OK	OK	UNST
11	1.4648e-003	OK	FAIL	OK	OK	OK	OK	STAB
12	7.3242e-004	OK	OK	OK	OK	OK	OK	STAB
13	1.0986e-003	OK	FAIL	OK	OK	OK	OK	STAB
14	9.1553e-004	OK	OK	OK	OK	OK	OK	STAB
15	1.0071e-003	OK	OK	OK	OK	OK	OK	STAB
16	1.0529e-003	OK	FAIL	OK	OK	OK	OK	STAB
17	1.0300e-003	OK	FAIL	OK	OK	OK	OK	STAB
18	1.0185e-003	OK	FAIL	OK	OK	OK	OK	STAB
19	1.0128e-003	OK	OK	OK	OK	OK	OK	STAB
20	1.0157e-003	OK	OK	OK	OK	OK	OK	STAB
21	1.0171e-003	OK	FAIL	OK	OK	OK	OK	STAB
22	1.0164e-003	OK	OK	OK	OK	OK	OK	STAB
23	1.0167e-003	OK	OK	OK	OK	OK	OK	STAB
24	1.0169e-003	OK	FAIL	OK	OK	OK	OK	STAB
25	1.0168e-003	OK	OK	OK	OK	OK	OK	STAB

Table 19 Output from Matlab's "hopt" function

Appendix E

Chapter 3 Case Study

E.1 Plant

The following are the state-space matrices for the system under investigating. The model is a MATLAB standard example provided as part of the Robust Control Toolbox. The example only provides the state-space model. The intention of this section is purely to compare PBIL and optimal H_∞ in terms of output performance. The decision to use this MATLAB H_∞ reference example, rather than a power system model, is to remove any bias in the comparison, thereby providing a high confidence factor in the validity of the PBIL-based sub-optimal H_∞design/tuning procedure.

$$A = \begin{bmatrix} -1.0285 & 0.9853 & -0.9413 & 0.0927 \\ -1.2903 & -1.0957 & 2.8689 & 4.7950 \\ 0.1871 & -3.8184 & -2.0788 & -0.9781 \\ 0.4069 & -4.1636 & 2.5407 & -1.4236 \end{bmatrix} \quad \text{E.1}$$

$$B = \begin{bmatrix} 0 \\ 6.6389 \\ 0 \\ 0 \end{bmatrix}$$

$$C = \begin{bmatrix} -1.7786 & 1.1390 & 0 & -1.0294 \end{bmatrix}$$

$$D = \begin{bmatrix} 0 \end{bmatrix}$$

E.2 PBIL-based Sub-optimal H_∞ Design Method Matlab Scripts

E.2.1 Augmentation Script

The augmentation script (`pbilex1aug.m`) is used to augment the plant.

```
function [opo]=pbilex1aug(gamma)

% Load the plant state-space description contained in
% in the matlab script pbilex1plant.m

pbilex1plant;

% Specify the weighting functions

numw1=[1 2 1]/100/1.5;
denw1=[1/30/30 2/30 1];

numw3=3.16*[1/300 1];
denw3=[1/10 1];

numw2=[1/100 1/10];
denw2=[1 1/10];

w1=gamma*[denw1;numw1];

w2=[numw2;denw2];

w3=[denw3;numw3];

%Augment the plant

[AA,B1,B2,C1,C2,D11,D12,D21,D22] = augtf(A,B,C,D,w1,w2,w3);
opo=1;

% Save the augmented plant. This is used by the optimization
% function
save pbilex1mat.m
```

E.2.2 Objective function

This function is called for each iteration by the primary optimization script "simulation.m". It takes values for the unknown controller parameters and calculates the minimum damping and maximum singular value for the closed loop response.

```
function [fuop,fuobj,fudamp]=pbilex1obj(xZ)

%Load the augmented plant
load pbilex1mat.m -mat

w=logspace(-2.5,2.2,20000);

% controller definition in terms of the unknown parameters

cnumz1 = conv([1 (xZ(1)+xZ(2)*1i)],[1 (xZ(1)-xZ(2)*1i)]);
cnumz2 = conv([1 xZ(3)],[1 xZ(4)]);
cnumz3 = conv([1 xZ(5)],[1 xZ(6)]);
cnumz4 = ([1 xZ(16)]);

cdenz1 = conv([1 (xZ(7)+xZ(8)*1i)],[1 (xZ(7)-xZ(8)*1i)]);
cdenz2 = conv([1 xZ(9)],[1 xZ(10)]);
cdenz3 = conv([1 xZ(11)],[1 xZ(12)]);
cdenz4 = conv([1 xZ(13)],[1 xZ(14)]);
connum = xZ(15)*conv(conv(cnumz1,cnumz2),conv(cnumz3,cnumz4));
conden = conv(conv(cdenz1,cdenz2),conv(cdenz3,cdenz4));

% create state-space matrices for the controller
[E,F,G,H] = tf2ss(connum,conden);

%create closed loop matrices
AT = [(AA+B2*H*C2)   (B2*G)
      F*C2                E];
BT = [(B2*H*D21)+B1
      (F*D21)];
CT = [(C1+D12*H*C2)  (D12*G)];
DT = [(D12*H*D21)+D11];

% create objective function and calculate max singular value
[T11] = max(sigma(AT,BT,[CT(1,:);CT(3,:)],[DT(1,1);DT(3,1)],w));
SV    = [T11];
Fuop  = max(SV);
Fuobj = max(SV);

% calculate minimum damping
[x,y]=damp(AT);
fudamp=min(y);
```

E.2.3 Main Script

This is the main simulation script (simulation.m). It uses the data and functions described in the above.

```
% simulation.m
%
% Define the constants for PBIL algorithm
    pos     =   0;
    L       =  0.10 ;            % learning rate
    MP      =  .02 ;             % mutation probability
    NVARS   =   17 ;             % number of variables
    TRIALS  =  60 ;              % number of trials per generation
    MAXGEN  =  150 ;             % number of generations
    PREC    =   22 ;             % precision (number of bits)
    len = NVARS*PREC;            % number of bits in vector
    rand('seed',sum(100*clock))  % initialize random generator
    PV = 0.5*ones(1,len);        % probability vector init to 0.5

    %Initialize storage vectors
    QF       = [];
    QD       = [];
    Qgamma   = [];
    xxtemp   = [];

    %Initialize variables
    bestever = 1*10^15 ;
    dever    = -1*10^15;
    it_gen   = 0;
    it_trial = 0;
    gammabest= 0;

    %Determine minimum damping in plant
    [planteig,plantdamp]=damp(A);
    plantmindamp = min(plantdamp);

   for gen = 1:MAXGEN;              % Generation Loop start

      zmax  = 1*10^15;              % Initialization max Hinf
      dmax  = -1*10^15;             % Initialization max damping
      dbest = -1*10^15;             % Initialization best damping
      bmax  = zeros(1,PREC*NVARS); % Initialization max damping

      for t = 1:TRIALS                      % Trial Loop Start

        QB    = (rand(1,length(PV)) < PV );  % generate random
trial

        % Split the vector into individual variables
        QB1  = QB(1:PREC);
        QB2  = QB(PREC+1:2*PREC);
        QB3  = QB(2*PREC+1:3*PREC);
        QB4  = QB(3*PREC+1:4*PREC);
        QB5  = QB(4*PREC+1:5*PREC);
        QB6  = QB(5*PREC+1:6*PREC);
        QB7  = QB(6*PREC+1:7*PREC);
        QB8  = QB(7*PREC+1:8*PREC);
        QB9  = QB(8*PREC+1:9*PREC);
```

```
QB10 = QB(9*PREC+1:10*PREC);
QB11 = QB(10*PREC+1:11*PREC);
QB12 = QB(11*PREC+1:12*PREC);
QB13 = QB(12*PREC+1:13*PREC);
QB14 = QB(13*PREC+1:14*PREC);
QB15 = QB(14*PREC+1:15*PREC);
QB16 = QB(15*PREC+1:16*PREC);
QB17 = QB(16*PREC+1:17*PREC);

% Create a weighting vector to generate real numbers
% vector of binary weights

P = (2 .^((PREC-1):-1:0))/2^PREC;

% convert to real numbers in range 0 and 1
qd1  = P*QB1';
qd2  = P*QB2';
qd3  = P*QB3';
qd4  = P*QB4';
qd5  = P*QB5';
qd6  = P*QB6';
qd7  = P*QB7';
qd8  = P*QB8';
qd9  = P*QB9';
qd10 = P*QB10';
qd11 = P*QB11';
qd12 = P*QB12';
qd13 = P*QB13';
qd14 = P*QB14';
qd15 = P*QB15';
qd16 = P*QB16';
qd17 = P*QB17';

% Scale the values
z1   = qd1*(20);                % scale into input
z2   = qd2*(15);                % range 0 to 15
z3   = qd3*(300);               % range 0 to 300
z4   = qd4*(300);               % range 0 to 300
z5   = qd5*(300);               % range 0 to 300
z6   = qd6*(300);               % range 0 to 300
z7   = qd7*(20);                % range 0 to 20
z8   = qd8*(15);                % range 0 to 15
z9   = qd9*(20);                % range 0 to 20
z10  = qd10*(15);               % range 0 to 15
z11  = qd11*(300);              % range 0 to 300
z12  = qd12*(300);              % range 0 to 300
z13  = qd13*(300);              % range 0 to 300
z14  = qd14*(300);              % range 0 to 300
z16  = qd16*(300);              % range 0 to 300
z17  = 0.5+qd17*(1);            % range 0.5 to 1.5
kG   = qd15*(500);              % range 0 to 500

% Create the parameter list to pass to optimization
function
xx1=[z1 z2 z3 z4 z5 z6 z7 z8 z9];
xx2=[z10 z11 z12 z13 z14 kG z16 z17];
xx = [xx1 xx2];

% Augment plant with gamma
[mmm]=pbilex1aug(z17);
```

```
    % Caculate max SV and min damping for given parameters,
    [z,SV,opdamp]=pbilex1obj(xx);

    % Apply optimization rule. Determine whether this is a
    % better solution than the previous in this generation
    if z <= zmax
       if opdamp >= (plantmindamp)
          if(opdamp <= (0.25))
           gammabest= z17;
           zmax=z;
           bmax=QB;
           dmax = opdamp;
           xxtemp=xx;
          end
       end
    end
   end                                    % Trial loop

   % Store the values for plotting for this generation
   QF = [QF,zmax];
   QD = [QD,dmax];
   Qgamma = [Qgamma,gammabest];

   % Update the probability vector based on best solution from
   % generation
   PV = (1-L)*PV + L*bmax;

   % mutate probability vector - prevents convergence to local
   % points
   for k = 1: length(PV)
       if rand < MP
          PV(k) =  (1-L)*PV(k) + L*(rand < 0.5) ;
       end
   end

   % Store best solution ever found in all generations
   if zmax <= bestever
      bestever = zmax;
      dever=dmax;
      xxbest=xxtemp;
      gammabest= z17;
      it_gen =gen;
      it_trial = t;
   end
end                                    % End generation loop
```

E.3 Standard Optimal H_∞ Design Method Matlab Script

The optimal H_∞ design was done using Matlab's Robust Controller Toolbox. The toolbox provides a standard set of procedures to design optimal H_∞ controllers.

E.3.1 Matlab script

The following Matlab script is used to calculate the optimal H_∞ controller.

```
format short e
clear ae be ce de;
% Create a system model for the open loop plant
sys = mksys(A,B,C,D);

% Create system models for the weighting functions
[aw1,bw1,cw1,dw1] = tf2ss(den1,num1); sysw1 =
mksys(aw1,bw1,cw1,dw1);
[aw2,bw2,cw2,dw2] = tf2ss(num2,den2); sysw2 =
mksys(aw2,bw2,cw2,dw2);
[aw3,bw3,cw3,dw3] = tf2ss(den3,num3); sysw3 =
mksys(aw3,bw3,cw3,dw3);

clear aw1 aw2 aw3 bw1 bw2 bw3 cw1 cw2 cw3 dw1 dw2 dw3
disp('Use AUGMENT to construct the augmented plant');

% Augment the open loop plant
sys_=augss(sys,sysw1,sysw2,sysw3,0);

% Extract the matrices of the augmented plant

[Aq,B1q,B2q,C1q,C2q,D11q,D12q,D21q,D22q]=branch(sys_);

% Apply continuous order balanced realization to augmented plant
[aa,bb,cc,mm,tt] = obalreal(Aq,[B1q B2q],[C1q;C2q]);

% Extract the necessary matrices and create a system model
Aq = aa;
B1q = bb(:,1);   B2q = bb(:,2);
C1q = cc(1:3,:); C2q = cc(4,:);
sys_ = mksys(Aq,B1q,B2q,C1q,C2q,D11q,D12q,D21q,D22q,'tss');

disp('Use HINF to design an H_INF controller');
% Run the H-infinity design procedure.
[gamm,ss_cp,ss_cl]=hinfopt(sys_);
% Extract the state-space matrices for the controller
[ae,be,ce,de]=branch(ss_cp);
% Extract the state-space matrices for the closed loop system
[ax,bx,cx,dx]=branch(ss_cl);
disp('Done')
```

E.3.2 Output from Optimal H_∞ Design

The following table is the output generated by the MATLAB function "hopt". Based on the specified tolerance, iteration number 7 is deemed by Matlab to be the best answer.

No	Gamma	D11<=1	P-Exist	P>=0	S-Exist	S>=0	lam(PS)<1	C.L.
1	OK	FAIL	OK	OK	OK	OK	STAB	1
2	OK	OK	OK	OK	OK	OK	STAB	0.5
3	OK	OK	OK	OK	OK	OK	STAB	0.75
4	OK	OK	OK	OK	OK	OK	STAB	0.875
5	OK	OK	OK	OK	OK	OK	STAB	0.9375
6	OK	FAIL	OK	OK	OK	OK	STAB	0.96875
7	OK	OK	OK	OK	OK	OK	STAB	0.95313
8	OK	FAIL	OK	OK	OK	OK	STAB	0.96094

Table 20 Output of Matlab "hopt" function

Appendix F

Chapter 4 Case Study 1

F.1 Plant

The following are the state-space matrices for the 3M9B power system under consideration:

$$A1=\begin{bmatrix}
-3.7232 & 0.0022 & 3.3794 & -0.04 & -0.0279 & -0.0060 & -0.025 & -0.018 & 0.0000 & -0.015 \\
0.0934 & -1.271 & -0.314 & 0.4616 & 0.0029 & -0.0030 & 0.0026 & -0.01 & 0.0000 & 0.0134 \\
19.727 & 0.0238 & -21.69 & -0.425 & 0.4792 & 0.1026 & 0.4236 & 0.3108 & 0.0000 & 0.2584 \\
-0.5158 & 7.7452 & 1.6933 & -13.49 & -0.6428 & 0.6563 & -0.568 & 1.9878 & 0.0000 & -2.93 \\
-0.0251 & 0.0039 & 0.0824 & -0.07 & -1.2427 & -0.0422 & 0.6870 & -0.128 & 0.0000 & -0.168 \\
-0.1266 & -0.014 & 0.4156 & 0.2666 & -0.6797 & -8.5599 & -0.6 & 4.3653 & 0.0000 & 0.4266 \\
-0.7742 & 0.1206 & 2.5415 & -2.147 & 27.1430 & -0.2692 & -33.1 & -0.815 & 0.0000 & -5.191 \\
-0.8037 & -0.091 & 2.6387 & 1.6928 & 0.5209 & 11.484 & 0.4603 & -16.87 & 0.0000 & 2.7083 \\
0.0078 & 0.0062 & -0.026 & -0.112 & -0.1413 & 0.0151 & -0.125 & 0.0457 & -0.127 & -0.224 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 376.99 & 0.0000 \\
-0.0383 & 0.0046 & 0.1257 & -0.082 & 0.0840 & -0.0029 & 0.0743 & -0.009 & 0.0000 & 0.1132 \\
-0.1808 & -0.027 & 0.5932 & 0.4929 & 0.0328 & 0.2092 & 0.0289 & 0.6335 & 0.0000 & -0.640 \\
-1.1316 & 0.1355 & 3.7136 & -2.423 & 2.4817 & -0.0848 & 2.1938 & -0.257 & 0.0000 & 3.3437 \\
-0.7000 & -0.105 & 2.2975 & 1.9088 & 0.1269 & 0.8101 & 0.1120 & 2.4536 & 0.0000 & -2.479 \\
0.0031 & 0.0107 & -0.010 & -0.193 & 0.0759 & -0.0504 & 0.0671 & -0.153 & 0.0000 & 0.2577 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
169.28 & 5.5591 & -555.8 & -99.72 & -39.347 & -17.957 & -34.78 & -54.42 & 0.0000 & 9.7300 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
30.678 & -0.144 & -100.7 & 2.2232 & -171.63 & -76.270 & -151.7 & -231.4 & 0.0000 & 34.479 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
38.403 & 0.3610 & -126.0 & -6.616 & -47.529 & -19.603 & -42.01 & -59.4 & 0.0000 & 4.9194
\end{bmatrix}$$

A2=[

-0.0252	-0.006	-0.020	-0.011	0.0000	-0.0164	0.0000	0.1855	0.0000	0.0000
0.0021	-0.004	0.0017	-0.007	0.0000	0.0107	0.0000	0.0000	0.0000	0.0000
0.4340	0.094	0.3484	0.1842	0.0000	0.2824	0.0000	0.0000	0.0000	0.0000
-0.4517	0.774	-0.363	1.5174	0.0000	-2.3303	0.0000	0.0000	0.0000	0.0000
0.0526	-0.01	0.0423	-0.019	0.0000	0.0832	0.0000	0.0000	0.0000	0.1695
0.0690	0.14	0.0555	0.2763	0.0000	-0.2494	0.0000	0.0000	0.0000	0.0000
1.6243	-0.295	1.3039	-0.578	0.0000	2.5670	0.0000	0.0000	0.0000	0.0000
0.4383	0.895	0.3520	1.7542	0.0000	-1.5836	0.0000	0.0000	0.0000	0.0000
0.0262	-0.038	0.0211	-0.075	0.0000	0.1193	0.0000	0.0000	0.0000	0.0000
0.0000	0.000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
-1.2424	-0.065	0.5797	-0.128	0.0000	-0.2195	0.0000	0.0000	0.0000	0.0000
-0.6864	-8.976	-0.549	4.3962	0.0000	1.0026	0.0000	0.0000	0.0000	0.0000
24.844	-0.392	-33.08	-0.769	0.0000	-6.4851	0.0000	0.0000	0.0000	0.0000
0.4495	10.56	0.3608	-16.31	0.0000	3.8829	0.0000	0.0000	0.0000	0.0000
-0.2395	0.072	-0.192	0.1410	-0.138	-0.4455	0.0000	0.0000	0.0000	0.0000
0.0000	0.000	0.0000	0.0000	376.99	0.0000	0.0000	0.0000	0.0000	0.0000
0.0000	0.000	0.0000	0.0000	0.0000	0.0000	-1.000	0.0000	0.0000	0.0000
-37.198	-18.52	-29.85	-36.32	0.0000	0.2120	600.00	-20.00	0.0000	0.0000
0.0000	0.000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	-1.000	0.0000
-38.001	-13.68	-30.50	-26.83	0.0000	-12.016	0.0000	0.0000	600.00	-20.00
0.0000	0.000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
-163.35	-87.20	-131.1	-171.2	0.0000	13.9650	0.0000	0.0000	0.0000	0.0000

]

A3=[

0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0.16978
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
-1	0
600	-20

]

A=[A1 A2 A3];

B=[

0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0.20982
0	0
0	0
0	0
0	0
0	0
0	0
600	0

]

C1=[

-0.064	-0.001	0.2101	0.0110	0.0792	0.0327	0.0700	0.0990	0.0000	-0.008
-0.019	-0.065	0.0616	1.1793	-0.4629	0.3073	-0.4093	0.9307	0.0000	-1.572

]

C2=[

0.27226	0.14548	0.21846	0.28538	0	-0.0233	0	0	0	0	0	0
1.4612	-0.439	1.1726	-0.8601	0	2.7179	0	0	0	0	0	0

]

C=[C1 C2];

D=[

0	0
0	0

]

F.2 Weighting Functions

The weighting functions are defined as:

$$w_1(s) = \frac{0.01(s+70)^2}{(s+0.2199)^2}$$

$$w_2(s) = 0.001$$

$$w_3(s) = \frac{100(s+2.268)^2}{(s+226.757)}$$

F.3 PBIL-based Sub-optimal H_∞ Design/tuning Method (constrained) Matlab Scripts

F.3.1 Augmentation Script

The augmentation script (`pbilaug.m`) is used to augment the plant.

```
% pbilaug.m
% load the plant state-space description contained in
% in the matlab script plant.m
load plant;

% Specify the weighting functions
numw1=1000*conv([0.0144 1],[0.0144 1]);
denw1=conv([4.547 1],[4.547 1]);

numw3=0.01*conv([0.441 1],[0.441 1]);
denw3=conv([0.00441 1],[0.00441 1]);


w1=[numw1;denw1;numw1;denw1];
w2=[0.001;1;0.001;1];
w3=[numw3;denw3;numw3;denw3];

%Augment the plant
[AA,B1,B2,C1,C2,D11,D12,D21,D22] = augtf(A,B,C,D,w1,w2,w3);

% Save the augmented plant. This is used by the optimization
% function
save augplantmat.m
```

F.3.2 Objective function

This function is called for each iteration by the primary optimization script "pbiloptconst.m". It takes values for the unknown controller parameters and calculates the minimum damping and maximum singular value for the closed loop response.

```
% pbilobjconst.m
function [fuop,fuobj,fudamp]=pbilobjconst(xZ)

% load the augmented plant
load augplantmat.m -mat

% define the frequency range for evaluation
w=logspace(0.01,1.5,200);

% define PSS structure
connum=xZ(5)*conv([xZ(1) 1],[xZ(2) 1]);
conden=conv([xZ(3) 1],[xZ(4) 1]);
[E,F,G,H]=tf2ss(connum,conden);

%create closed loop matrices

AT=[(AA+B2(:,1)*H*C2(2,:)) (B2(:,1)*G)
    (F*C2(2,:))             E        ];
BT=[(B2(:,1)*H*D21(1,:))+B1
     F*D21(1,:)              ];
CT=[(C1+D12(:,1)*H*C2(2,:)) (D12(:,1)*G)];
DT=[(D12(:,1)*H*D21(1,:))+D11];

%create objective function
[T12]=max(sigma(AT,BT(:,1),[CT(2,:);CT(6,:)],[DT(2,1);DT(6,1)],w))
;
SV=[T12];

% return the maximum singular value
fuop=max(SV)
fuobj=max(SV);

% calculate the damping
[x,y]=damp(AT);
fudamp=min(y)
```

F.3.3 Main Script

This is the main simulation script (pbiloptconst.m). It uses the data and functions described in the above sections.

```
% pbiloptconst.m
% define the constants for PBIL algorithm
pos     =   0;
L       =  0.1 ;                    % learning rate
MP      =  .02 ;                    % mutation probability
NVARS   =    5 ;                    % number of variables
TRIALS  =  30 ;                     % number of trials per
generation
MAXGEN  =  100 ;                    % number of generations
PREC    =   22 ;                    % precision (number of bits)
rand('seed',sum(100*clock))         % initialize random generator
len = NVARS*PREC;                   % number of bits in string
PV = 0.5*ones(1,len);               % probability vector init. to
0.5

% initialize storage vectors
QF = [];
QD= [];

% initialize variables
bestever = 1*10^15 ;
dever =-1*10^15;
for gen = 1:MAXGEN ;                % generation loop
   % initialize variables
   zmax = 1*10^15;
   dmax =-1*10^15;
   dbest=-1*10^15;
   bmax =  zeros(1,PREC*5) ;

   for t = 1:TRIALS                      % trial loop
      % generate random solution vector
      QB = (rand(1,length(PV)) < PV );

      % extract parameters from solution vector
      QB1 =  QB(1:PREC);
      QB2 =  QB(PREC+1:2*PREC);
      QB3 =  QB(2*PREC+1:3*PREC);
      QB4 =  QB(3*PREC+1:4*PREC);
      QB5 =  QB(4*PREC+1:5*PREC);

      % create weights for each bit (real number between 0 and 1)
      P = (2 .^((PREC-1):-1:0))/2^PREC;

      % convert binary data to a real number by weighting with P
      qd1 = P*QB1';
      qd2 = P*QB2';
      qd3 = P*QB3';
      qd4 = P*QB4';
      qd5 = P*QB5';
```

```
      % scale parameters
      x = qd1*(100);
      y = qd2*(100);
      xp= qd3*(100);
      yp= qd4*(100);
      kG = qd5*(20);

      % construct solution vector for unknown PSS parameters
      xx=[x y xp yp kG];

      % calculate minimum damping (opdamp)
      % and maximum singular value (z, SV)
      [z,SV,opdamp]=pbilobjconst(xx);

      % see whether the solution is better than the last
      if z < zmax
         if opdamp> 0
            if opdamp > dmax
               zmax=z;
               bmax=QB;
               dmax = opdamp;
            end
         end
      end
   end                                    % end of trial loop


   % store the best solution for this generation
   QF = [QF,zmax];
   QD = [QD,dmax];

   % update probability vector
   PV = (1-L)*PV + L*bmax;

   % mutate probability vector
   for k = 1: length(PV)
      if rand < MP
         PV(k) =  (1-L)*PV(k) + L*(rand < 0.5) ;
      end
   end

   % keep best solution found for in generations
   % this is not necessarily the converged solution
   if zmax < bestever
      if dmax >dever
         bestever = zmax;
         dever=dmax;
         xxbest=xx;
      end
   end
end                                       % end of generation loop
```

Appendix G

Chapter 4 Case Study 2

G.1 Plant

The following are the state-space matrices for the 3M9B power system under consideration:

$$A1 = \begin{bmatrix}
-3.7232 & 0.0022 & 3.3794 & -0.04 & -0.0279 & -0.0060 & -0.025 & -0.018 & 0.0000 & -0.015 \\
0.0934 & -1.271 & -0.314 & 0.4616 & 0.0029 & -0.0030 & 0.0026 & -0.01 & 0.0000 & 0.0134 \\
19.727 & 0.0238 & -21.69 & -0.425 & 0.4792 & 0.1026 & 0.4236 & 0.3108 & 0.0000 & 0.2584 \\
-0.5158 & 7.7452 & 1.6933 & -13.49 & -0.6428 & 0.6563 & -0.568 & 1.9878 & 0.0000 & -2.93 \\
-0.0251 & 0.0039 & 0.0824 & -0.07 & -1.2427 & -0.0422 & 0.6870 & -0.128 & 0.0000 & -0.168 \\
-0.1266 & -0.014 & 0.4156 & 0.2666 & -0.6797 & -8.5599 & -0.6 & 4.3653 & 0.0000 & 0.4266 \\
-0.7742 & 0.1206 & 2.5415 & -2.147 & 27.1430 & -0.2692 & -33.1 & -0.815 & 0.0000 & -5.191 \\
-0.8037 & -0.091 & 2.6387 & 1.6928 & 0.5209 & 11.484 & 0.4603 & -16.87 & 0.0000 & 2.7083 \\
0.0078 & 0.0062 & -0.026 & -0.112 & -0.1413 & 0.0151 & -0.125 & 0.0457 & -0.127 & -0.224 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 376.99 & 0.0000 \\
-0.0383 & 0.0046 & 0.1257 & -0.082 & 0.0840 & -0.0029 & 0.0743 & -0.009 & 0.0000 & 0.1132 \\
-0.1808 & -0.027 & 0.5932 & 0.4929 & 0.0328 & 0.2092 & 0.0289 & 0.6335 & 0.0000 & -0.640 \\
-1.1316 & 0.1355 & 3.7136 & -2.423 & 2.4817 & -0.0848 & 2.1938 & -0.257 & 0.0000 & 3.3437 \\
-0.7000 & -0.105 & 2.2975 & 1.9088 & 0.1269 & 0.8101 & 0.1120 & 2.4536 & 0.0000 & -2.479 \\
0.0031 & 0.0107 & -0.010 & -0.193 & 0.0759 & -0.0504 & 0.0671 & -0.153 & 0.0000 & 0.2577 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
169.28 & 5.5591 & -555.8 & -99.72 & -39.347 & -17.957 & -34.78 & -54.42 & 0.0000 & 9.7300 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
30.678 & -0.144 & -100.7 & 2.2232 & -171.63 & -76.270 & -151.7 & -231.4 & 0.0000 & 34.479 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
38.403 & 0.3610 & -126.0 & -6.616 & -47.529 & -19.603 & -42.01 & -59.4 & 0.0000 & 4.9194
\end{bmatrix}$$

A2=[

-0.0252	-0.006	-0.020	-0.011	0.0000	-0.0164	0.0000	0.1855	0.0000	0.0000
0.0021	-0.004	0.0017	-0.007	0.0000	0.0107	0.0000	0.0000	0.0000	0.0000
0.4340	0.094	0.3484	0.1842	0.0000	0.2824	0.0000	0.0000	0.0000	0.0000
-0.4517	0.774	-0.363	1.5174	0.0000	-2.3303	0.0000	0.0000	0.0000	0.0000
0.0526	-0.01	0.0423	-0.019	0.0000	0.0832	0.0000	0.0000	0.0000	0.1695
0.0690	0.14	0.0555	0.2763	0.0000	-0.2494	0.0000	0.0000	0.0000	0.0000
1.6243	-0.295	1.3039	-0.578	0.0000	2.5670	0.0000	0.0000	0.0000	0.0000
0.4383	0.895	0.3520	1.7542	0.0000	-1.5836	0.0000	0.0000	0.0000	0.0000
0.0262	-0.038	0.0211	-0.075	0.0000	0.1193	0.0000	0.0000	0.0000	0.0000
0.0000	0.000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
-1.2424	-0.065	0.5797	-0.128	0.0000	-0.2195	0.0000	0.0000	0.0000	0.0000
-0.6864	-8.976	-0.549	4.3962	0.0000	1.0026	0.0000	0.0000	0.0000	0.0000
24.844	-0.392	-33.08	-0.769	0.0000	-6.4851	0.0000	0.0000	0.0000	0.0000
0.4495	10.56	0.3608	-16.31	0.0000	3.8829	0.0000	0.0000	0.0000	0.0000
-0.2395	0.072	-0.192	0.1410	-0.138	-0.4455	0.0000	0.0000	0.0000	0.0000
0.0000	0.000	0.0000	0.0000	376.99	0.0000	0.0000	0.0000	0.0000	0.0000
0.0000	0.000	0.0000	0.0000	0.0000	0.0000	-1.000	0.0000	0.0000	0.0000
-37.198	-18.52	-29.85	-36.32	0.0000	0.2120	600.00	-20.00	0.0000	0.0000
0.0000	0.000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	-1.000	0.0000
-38.001	-13.68	-30.50	-26.83	0.0000	-12.016	0.0000	0.0000	600.00	-20.00
0.0000	0.000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
-163.35	-87.20	-131.1	-171.2	0.0000	13.9650	0.0000	0.0000	0.0000	0.0000

]

A3=[

0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0.16978
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
-1	0
600	-20

]

A=[A1 A2 A3];

B=[
0 0
0 0
0 0
0 0
0 0
0 0
0 0
0 0
0 0
0 0
0 0
0 0
0 0
0 0.20982
0 0
0 0
0 0
0 0
0 0
0 0
600 0
]

C1=[
-0.064 -0.001 0.2101 0.0110 0.0792 0.0327 0.0700 0.0990 0.0000 -0.008
-0.019 -0.065 0.0616 1.1793 -0.4629 0.3073 -0.4093 0.9307 0.0000 -1.572
]

C2=[
0.27226 0.14548 0.21846 0.28538 0 -0.0233 0 0 0 0 0 0
1.4612 -0.439 1.1726 -0.8601 0 2.7179 0 0 0 0 0 0
]

C=[C1 C2];

D=[
0 0
0 0
]

G.2 Weighting Functions

The weighting functions are defined as:

$$w_1(s) = \frac{0.01(s+70)^2}{(s+0.2199)^2}$$
$$w_2(s) = 0.001$$
$$w_3(s) = \frac{100(s+2.268)^2}{(s+226.757)}$$

G.3 PBIL-based Sub-optimal H_∞ Design/tuning Method (unconstrained) Matlab Scripts

G.3.1 Augmentation Script

The augmentation script (`pbilaug.m`) is used to augment the plant.

```
% pbilaug.m
% load the plant state-space description contained in
% in the matlab script plant.m
load plant;

% Specify the weighting functions
numw1=1000*conv([0.0144 1],[0.0144 1]);
denw1=conv([4.547 1],[4.547 1]);

numw3=0.01*conv([0.441 1],[0.441 1]);
denw3=conv([0.00441 1],[0.00441 1]);


w1=[numw1;denw1;numw1;denw1];
w2=[0.001;1;0.001;1];
w3=[numw3;denw3;numw3;denw3];

%Augment the plant
[AA,B1,B2,C1,C2,D11,D12,D21,D22] = augtf(A,B,C,D,w1,w2,w3);

% Save the augmented plant. This is used by the optimization
% function
save augplantmat.m
```

G.3.2 Objective function

This function is called for each iteration by the primary optimization script "pbiloptunc.m". It takes values for the unknown controller parameters and calculates the minimum damping and maximum singular value for the closed loop response.

```
%pbilobjunc.m
function [fuop,fuobj,fudamp]=pbilobjunc(xZ)

%load the augmented plant
load augplantmat.m -mat

% define the frequency range for evaluation
w=logspace(-2.5,2.2,20000);

% define PSS structure
connum=xZ(5)*conv([xZ(1) 1],[xZ(2) 1]);
conden=conv([xZ(3) 1],[xZ(4) 1]);
[E,F,G,H]=tf2ss(connum,conden);

%create closed loop matrices

AT=[(AA+B2(:,1)*H*C2(2,:)) (B2(:,1)*G)
    (F*C2(2,:))             E        ];
BT=[(B2(:,1)*H*D21(1,:))+B1
     F*D21(1,:)              ];
CT=[(C1+D12(:,1)*H*C2(2,:)) (D12(:,1)*G)];
DT=[(D12(:,1)*H*D21(1,:))+D11];

%create objective function
[T12]=max(sigma(AT,BT(:,1),[CT(2,:);CT(6,:)],[DT(2,1);DT(6,1)],w))
;
SV=[T12];

% return the maximum singular value
fuop=max(SV)
fuobj=max(SV);

% calculate the damping
[x,y]=damp(AT);
fudamp=min(y)
```

This is the main simulation script (pbiloptunc.m). It uses the data and functions described in the above sections.

```
%pbiloptunc.m
%
% Define the constants for PBIL algorithm
pos      =  0;
L        =  0.1 ;                  % learning rate
MP       =  0.02 ;                 % mutation probability
NVARS    =  5 ;                    % number of variables
TRIALS   =  30 ;                   % number of trials per
generation
MAXGEN   =  100 ;                  % number of generations
PREC     =  22 ;                   % precision (number of bits)
rand('seed',sum(100*clock))        % initialize random generator
len = NVARS*PREC;                  % number of bits in string
PV = 0.5*ones(1,len);              % probability vector init. to
0.5

% initialize storage vectors
QF = [];
QD = [];
QX = [];

% initialize variables
bestever = 1*10^15 ;
dever =-1*10^15;

for gen = 1:MAXGEN

   zmax = 1*10^15;                 %  initialisation
   dmax =-1*10^15;
   dbest=-1*10^15;
   bmax =  zeros(1,PREC*5) ;

   for t = 1:TRIALS                  % trial loop
      % generate random solution vector
      QB = (rand(1,length(PV)) < PV );

      % extract parameters from solution vector
      QB1 =  QB(1:PREC);
      QB2 =  QB(PREC+1:2*PREC);
      QB3 =  QB(2*PREC+1:3*PREC);
      QB4 =  QB(3*PREC+1:4*PREC);
      QB5 =  QB(4*PREC+1:5*PREC);

      % create weights for each bit (real number between 0 and 1)
      P = (2 .^((PREC-1):-1:0))/2^PREC;

      % convert binary data to a real number by weighting with P
      qd1 = P*QB1';
      qd2 = P*QB2';
      qd3 = P*QB3';
```

```
      qd4 = P*QB4';
      qd5 = P*QB5';


      % scale parameters
      x = qd1*(100);
      y = qd2*(100);
      xp= qd3*(100);
      yp= qd4*(100);
      kG = qd5*(100);

      % construct solution vector for unknown PSS parameters
      xx=[x y xp yp kG];

      % calculate minimum damping (opdamp)
      % and maximum singular value (z, SV)
      [z,opdamp]=pbilobjunc(xx)

      % see whether the solution is better than the last
      if opdamp > 0
         if z>=1
            if z < zmax
               zmax=z;
               bmax=QB;
               dmax = opdamp;
               xokay=xx
            end
         end
      end
   end                                % end of trial loop


   % store the best solution for this generation
   QF = [QF,zmax];
   QD = [QD,dmax];
   QX = [QX,xx];

   % update probability vector
   PV = (1-L)*PV + L*bmax;

   % mutate probability vector
   for k = 1: length(PV)
      if rand < MP
         PV(k) =  (1-MP)*PV(k) + MP*(rand < 0.5) ;
      end
   end

   % keep best solution found for in generations
   % this is not necessarily the converged solution
   if zmax < bestever
      bestever = zmax;
      xxbest=xx
   end
end
```

Appendix H

Chapter 5 Case Study 1

H.1 Plant

The following are the state-space matrices for the Eskom power network under consideration:

A1=[

-0.711	0.001	0.548	0.011	0.000	0.034	0.004	0.003	0.003	0.016
0.383	-32.12	0.948	22.180	0.000	1.658	-0.408	0.025	-0.310	0.164
81.13	-0.054	-101.40	-0.406	0.000	11.83	1.298	0.875	0.986	5.694
0.263	81.18	0.649	-132.70	0.000	45.120	-11.11	0.690	-8.437	4.476
0.030	0.019	0.074	0.152	0.040	-0.217	0.037	-0.006	0.028	-0.039
0.000	0.000	0.000	0.000	314.200	0.000	0.000	0.000	0.000	0.000
0.014	-0.017	0.035	-0.132	0.000	0.131	-1.985	-0.001	1.479	-0.008
0.255	0.033	0.630	0.260	0.000	-0.658	0.021	-29.97	0.016	19.79
0.199	-0.236	0.491	-1.867	0.000	1.865	27.650	-0.018	-32.16	-0.114
1.357	0.174	3.353	1.386	0.000	-3.504	0.113	16.280	0.087	-32.56
-0.010	-0.007	-0.024	-0.055	0.000	0.076	-0.076	0.016	-0.058	0.104
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
7.952	-1.463	19.650	-11.59	0.000	2.021	1.785	0.408	1.355	2.654
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
1.232	-0.299	3.043	-2.355	0.000	0.970	12.980	2.501	9.852	16.290
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

]

A2=[

0.000	-0.012	-2.914	8.830	0.032	0.000	0.000	0.000
0.000	-0.657	0.000	0.000	0.000	0.000	0.000	0.000
0.000	-4.078	0.000	0.000	0.000	0.000	0.000	0.000
0.000	-17.87	0.000	0.000	0.000	0.000	0.000	0.000
0.000	0.084	0.000	0.000	0.000	0.000	0.000	0.000
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
0.000	-0.289	0.000	0.000	0.000	-2.452	0.506	0.135
0.000	1.945	0.000	0.000	0.000	0.000	0.000	0.000
0.000	-4.098	0.000	0.000	0.000	0.000	0.000	0.000
0.000	10.35	0.000	0.000	0.000	0.000	0.000	0.000
-0.059	-0.195	0.000	0.000	0.000	0.000	0.000	0.000
314.2	0.000	0.000	0.000	0.000	0.000	0.000	0.000
0.000	-0.473	-42.550	0.000	0.000	0.000	0.000	0.000
0.000	0.000	-3.722	-5.556	0.000	0.000	0.000	0.000
0.000	0.000	-19.430	58.860	-0.183	0.000	0.000	0.000
0.000	-0.392	0.000	0.000	0.000	-42.55	0.000	0.000
0.000	0.000	0.000	0.000	0.000	98.620	-25.64	0.000
0.000	0.000	0.000	0.000	0.000	-21.09	4.353	-0.459

]

A=[A1 A2];

$$B=\begin{bmatrix}
0.0000 & 0.0000\\
0.0000 & 0.0000\\
2.9140 & 0.0000\\
0.0000 & 0.0000\\
0.0000 & 0.0000\\
0.0000 & 0.0000\\
0.0000 & 2.4520\\
0.0000 & 0.0000\\
0.0000 & 0.0000\\
0.0000 & 0.0000\\
0.0000 & 0.0000\\
0.0000 & 0.0000\\
3.7220 & 0.0000\\
19.4300 & 0.0000\\
0.0000 & 0.0000\\
0.0000 & -98.6200\\
0.0000 & 21.0900
\end{bmatrix}$$

$$C1=\begin{bmatrix}
-0.308 & -0.196 & -0.7597 & -1.552 & 0.00 & 2.2160 & -0.378 & 0.0616 & -0.287 & 0.4000\\
0.1084 & 0.0769 & 0.2677 & 0.6078 & 0.00 & -0.851 & 0.845 & -0.183 & 0.6415 & -1.163
\end{bmatrix}$$

$$C2=\begin{bmatrix}
0.0000 & -0.8537 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000\\
0.0000 & 2.1750 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000
\end{bmatrix}$$

$$C=[C1\ C2]$$

$$D=\begin{bmatrix}
0 & 0\\
0 & 0
\end{bmatrix}$$

H.2 Weighting Functions

The weighting functions are defined as:

$$w_1(s) = \frac{0.01(s+70)^2}{(s+0.2199)^2}$$
$$w_2(s) = 0.001$$
$$w_3(s) = \frac{100(s+2.268)^2}{(s+226.757)}$$

H.3 Coordinated Tuning Multiple PSS Matlab Script

H.3.1 Augmentation Script

The augmentation script (`plantaug.m`) is used to augment the plant.

```
% plantaug.m
% load the plant state-space description contained in
% in the matlab script plant.m
load plant;

% Specify the weighting functions
numw1=1000*conv([0.0144 1],[0.0144 1]);
denw1=conv([4.547 1],[4.547 1]);

numw3=0.01*conv([0.441 1],[0.441 1]);
denw3=conv([0.00441 1],[0.00441 1]);


w1=[numw1;denw1;numw1;denw1];
w2=[0.001;1;0.001;1];
w3=[numw3;denw3;numw3;denw3];

%Augment the plant
[AA,B1,B2,C1,C2,D11,D12,D21,D22] = augtf(A,B,C,D,w1,w2,w3);

% Save the augmented plant. This is used by the optimization
% function
save augplantmat.m
```

H.3.2 Objective function

This function is called for each iteration by the primary optimization script "pbilcoordobj.m". It takes values for the unknown controller parameters and calculates the minimum damping and maximum singular value for the closed loop response.

```
%pbilcoordobj.m
function [sv11,sv12,sv21,sv22,fudamp]=pbilcoordobj(pss1,pss2)

%load the augmented plant
load plantaugmat.m -mat

% define the frequency range for evaluation
w=logspace(-2.5,2.2,20000);

% define PSS structure
connum1=pss1(1)*[1 pss1(2)];
conden1=[1 pss1(3)];
connum2=pss2(1)*[1 pss2(2)];
conden2=[1 pss2(3)];
[E,F,G,H]=tf2ss(connum1,conden1);
[I,J,K,L]=tf2ss(connum2,conden2);

%create closed loop matrices
sizeE = size(E);
z1matrix=zeros(sizeE(1),sizeE(2));
%create closed loop matrices
sizeI = size(I);
z1matrix=zeros(sizeI(1),sizeI(2));

A11 = (AA+B2(:,1)*H*C2(1,:)+B2(:,2)*L*C2(2,:));
A12 = (B2(:,1)*G);
A13 = (B2(:,2)*K);
A21 = (F*C2(1,:))
A22 = E
A23 = z1matrix;
A31 = (J*C2(2,:));
A32 = z2matrix;
A33 = I;

AT=[  A11 A12 A13
      A21 A22 A23
      A31 A32 A33];

B11 = B1(:,1)+(B2(:,1)*H*D21(1,1))+(B2(:,2)*L*D21(2,1));
B12 = B1(:,2)+(B2(:,1)*H*D21(1,2))+(B2(:,2)*L*D21(2,2));
B21 = F*D21(1,1);
B22 = F*D21(1,2);
B31 = J*D21(2,1);
B32 = J*D21(2,2);

BT=[B11 B12
    B21 B22
    B31 B32];
```

```
CT11 = (C1+D12(:,1)*H*C2(1,:)+D12(:,2)*L*C2(2,:));
CT12 = (D12(:,1)*G);
CT13 = (D12(:,2)*K);
CT=[C11 C12 C13];

DT11 = D11(:,1)+(D12(:,1)*H*D21(1,1))+(D12(:,2)*L*D21(2,1));
DT12 = D11(:,2)+(D12(:,1)*H*D21(1,2))+(D12(:,2)*L*D21(2,2));
DT= [D11 D12];

%create objective function
[T11]=max(sigma(AT,BT(:,1),[CT(1,:);CT(5,:)],[DT(1,1);DT(5,1)],w))
;
[T12]=max(sigma(AT,BT(:,1),[CT(2,:);CT(6,:)],[DT(2,1);DT(6,1)],w))
;
[T21]=max(sigma(AT,BT(:,2),[CT(1,:);CT(5,:)],[DT(1,2);DT(5,2)],w))
;
[T22]=max(sigma(AT,BT(:,2),[CT(2,:);CT(6,:)],[DT(2,2);DT(6,2)],w))
;

% return the maximum singular value
sv11=max(T11);     sv22=max(T22);
sv12=max(T12);     sv21=max(T21);

% Calculate the damping factor
[x,y]=damp(AT);    fudamp=min(y);
```

H.3.3 Main Script

This is the main simulation script (coordopt.m). It uses the data and functions described in the above sections.

```
%coordopt.m
% Define the constants for PBIL algorithm
pos      =   0;
L        =   0.1 ;                % learning rate
MP       =   .02 ;                % mutation probability
NVARS    =   6 ;                  % number of variables
TRIALS   =   80 ;                 % number of trials per
                                  % generation
MAXGEN   =   80 ;                 % number of generations
PREC     =   22 ;                 % precision (number of bits)
rand('seed',sum(100*clock))       % initialize random generator
len = NVARS*PREC;                 % number of bits in string
PV = 0.5*ones(1,len);             % probability vector init. to 0.5

global w AA B1 B2 C1 C2 D12 D21 D22 D11;
load plantaugmat.m -mat;
w=logspace(-2,2,2000);

% initialize storage vectors
QF = [];
QD= [];

% initialize variables
zever = 1*10^15 ;
dever =-1*10^15;

dmax =0;                          % initialization
zmax = 1*10^15;

for gen = 1:MAXGEN

   bmax =  zeros(1,PREC*NVARS) ;
   dmax = 0.01;

   for t = 1:TRIALS               % trial loop

      % generate random solution vector
      QB   = (rand(1,length(PV)) < PV );
      % extract parameters from solution vector
      QB1  =  QB(1:PREC);
      QB2  =  QB(PREC+1:2*PREC);
      QB3  =  QB(2*PREC+1:3*PREC);
      QB4  =  QB(3*PREC+1:4*PREC);
      QB5  =  QB(4*PREC+1:5*PREC);
      QB6  =  QB(5*PREC+1:6*PREC);

      % create weights for each bit (real number between 0 and 1)
      P    = (2 .^((PREC-1):-1:0))/2^PREC;
      % convert binary data to a real number by weighting with P
```

```
        qd1  = 10^-5+P*QB1';
        qd2  = 10^-5+P*QB2';
        qd3  = 10^-5+P*QB3';
        qd4  = 10^-5+P*QB4';
        qd5  = 10^-5+P*QB5';
        qd6  = 10^-5+P*QB6';

        % scale parameters
        pss11 = qd1*(100);
        pss12 = qd2*(20);
        pss13 = qd3*(20);

        pss21 = qd4*(100);
        pss22 = qd5*(20);
        pss23 = qd6*(20);

        % construct solution vector for unknown PSS parameters
        pss1=[pss11 pss12 pss13];
        pss2=[pss21 pss22 pss23];

        %Objective Function
        [sv11,sv12,sv21,sv22,opdamp] = pbilcoordobj(pss1,pss2);
        z = max([sv11 sv22]);

        if opdamp>= dmax %dmax
              cmax = [pss1 pss2]
              bmax = QB;
              dmax = opdamp
              zmax = z
          end
    end
    QF = [QF,zmax];
    QD = [QD,dmax];
    PV = (1-L)*PV + L*bmax;   % update probability vector

    % mutate probability vector
    for k = 1: length(PV)
        if rand < MP
           PV(k) =  (1-L)*PV(k) + L*(rand < 0.5) ;
        end
    end

    % maintain best solution in all generations
    if dmax >=dever
        dever = dmax;
        zever = zmax;
        xxbest=[pss1 pss2];
    end
end
```

Appendix I

Chapter 5 Case Study 2

I.1 Plant

I.1.1 Speed Governor System Modeling

The turbine-governor models give representations of the effects of power plants on power system stability. However, the models are not intended to be used in studies of the detailed behavior of individual plants. A functional diagram of the representation used and its relationship to the generator is shown in Figure 63.

Because of the wide variety in the details of individual turbine control, the PSS/E models do not attempt to give a high degree of exactness for any given plant; rather represent the principal effects inherent in conventional steam turbine, gas turbine, nuclear, and hydro plants.

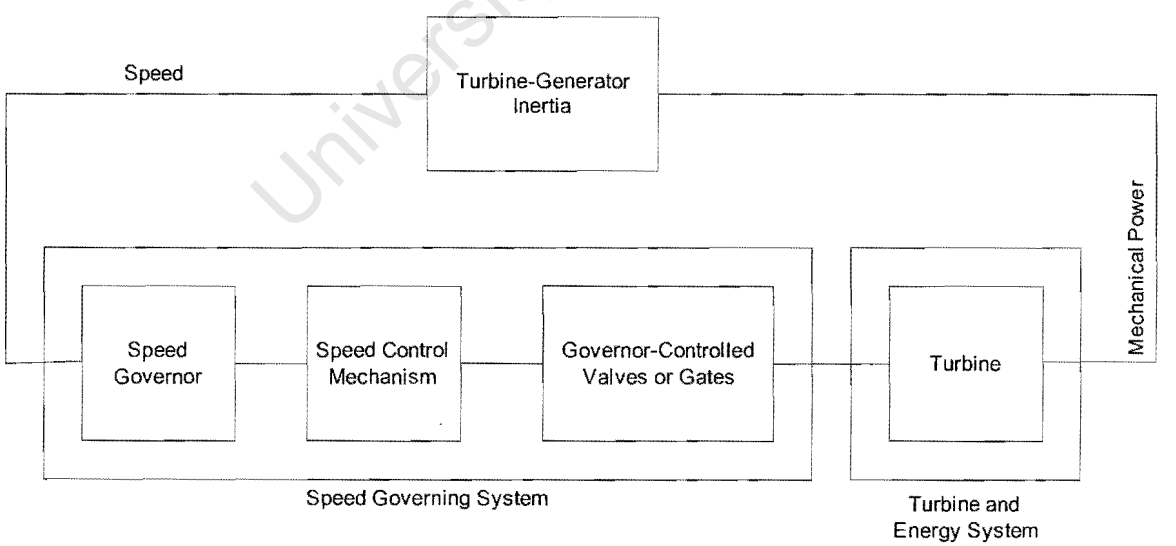

Figure 63 Speed governor and turbine in relationship to generator

In PSS/E, several models are available for hydro electric plant simulation. The model HYGOVM has been chosen to demonstrate how PBIL-based sub-optimal H_∞ method works on PSS-Speed governor coordination.

Figure 64 gives the block diagram of HYGOVM governor system.

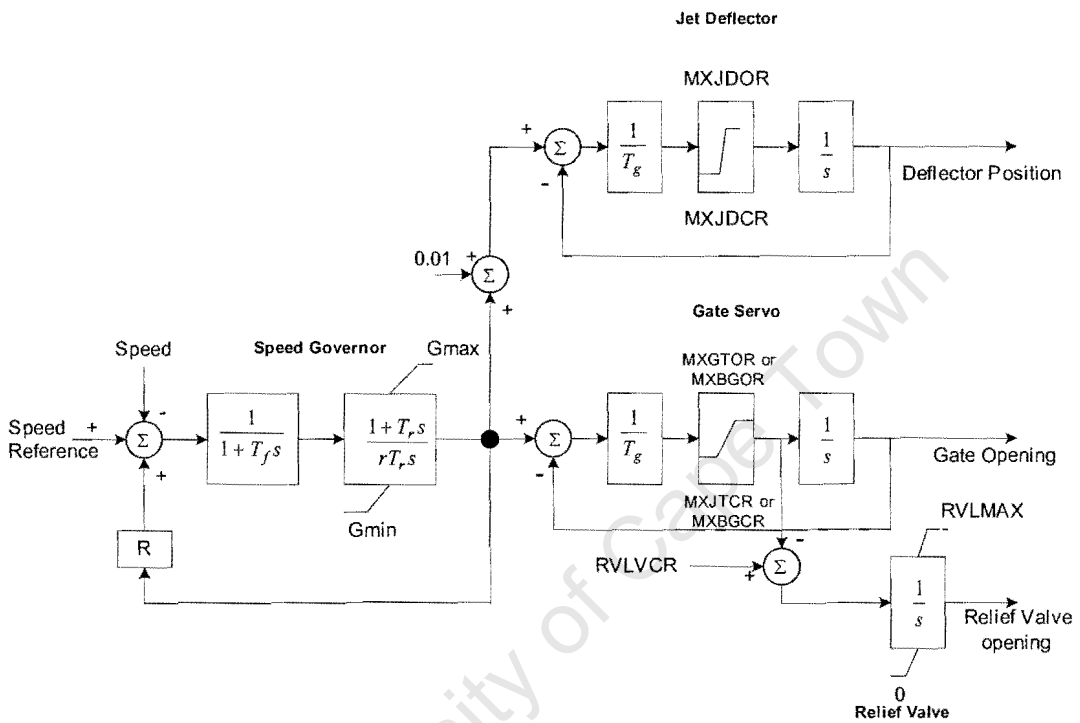

Figure 64 HYGOVM governor system

The parameters are:

R	Permanent droop	0.05 pu	*MXBGOR/MXBGCR*	Max. buffered gate opening/closing rate	+0.1/-0.05 pu/sec.
r	Temporary droop	Needs to be tuned	*MXGTOR/MXGTCR*	Max. gate opening/closing rate	+0.1/0.125 pu/sec.
T_f	Filter time constant	0.05 sec.	*RLVMAX*	Max. relief valve limit	1.0 pu
T_r	Governor time constant	Needs to be tuned	*RLVVCR*	Relief valve closing rate	-1/70 pu/sec.
T_g	Gate servo time constant	0.5 sec.	*MXJDOR*	Max. jet deflector opening rate	+0.5 sec.
G_{max}/G_{min}	Maximum/Minimum gate	1.0/0.0 pu	*MXJDCR*	Max. jet deflector closing rate	-0.5 sec.

I.1.2 State-space Matrices

The followings are the state-space matrices for the Eskom power network under consideration:

A1=[

-0.711	0.001	0.548	0.011	0.000	0.034	0.004	0.003	0.003	0.016
0.383	-32.12	0.948	22.180	0.000	1.658	-0.408	0.025	-0.310	0.164
81.13	-0.054	-101.40	-0.406	0.000	11.83	1.298	0.875	0.986	5.694
0.263	81.18	0.649	-132.70	0.000	45.120	-11.11	0.690	-8.437	4.476
0.030	0.019	0.074	0.152	0.040	-0.217	0.037	-0.006	0.028	-0.039
0.000	0.000	0.000	0.000	314.20	0.000	0.000	0.000	0.000	0.000
0.014	-0.017	0.035	-0.132	0.000	0.131	-1.985	-0.001	1.479	-0.008
0.255	0.033	0.630	0.260	0.000	-0.658	0.021	-29.97	0.016	19.79
0.199	-0.236	0.491	-1.867	0.000	1.865	27.650	-0.018	-32.16	-0.114
1.357	0.174	3.353	1.386	0.000	-3.504	0.113	16.280	0.087	-32.56
-0.010	-0.007	-0.024	-0.055	0.000	0.076	-0.076	0.016	-0.058	0.104
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
7.952	-1.463	19.650	-11.59	0.000	2.021	1.785	0.408	1.355	2.654
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
1.232	-0.299	3.043	-2.355	0.000	0.970	12.980	2.501	9.852	16.290
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

]

A2=[

0.000	-0.012	-2.914	8.830	0.032	0.000	0.000	0.000
0.000	-0.657	0.000	0.000	0.000	0.000	0.000	0.000
0.000	-4.078	0.000	0.000	0.000	0.000	0.000	0.000
0.000	-17.87	0.000	0.000	0.000	0.000	0.000	0.000
0.000	0.084	0.000	0.000	0.000	0.000	0.000	0.000
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
0.000	-0.289	0.000	0.000	0.000	-2.452	0.506	0.135
0.000	1.945	0.000	0.000	0.000	0.000	0.000	0.000
0.000	-4.098	0.000	0.000	0.000	0.000	0.000	0.000
0.000	10.35	0.000	0.000	0.000	0.000	0.000	0.000
-0.059	-0.195	0.000	0.000	0.000	0.000	0.000	0.000
314.2	0.000	0.000	0.000	0.000	0.000	0.000	0.000
0.000	-0.473	-42.550	0.000	0.000	0.000	0.000	0.000
0.000	0.000	-3.722	-5.556	0.000	0.000	0.000	0.000
0.000	0.000	-19.430	58.860	-0.183	0.000	0.000	0.000
0.000	-0.392	0.000	0.000	0.000	-42.55	0.000	0.000
0.000	0.000	0.000	0.000	0.000	98.620	-25.64	0.000
0.000	0.000	0.000	0.000	0.000	-21.09	4.353	-0.459

]

A=[A1 A2];

B=[

0.0000	0.0000
0.0000	0.0000
2.9140	0.0000
0.0000	0.0000
0.0000	0.0980
0.0000	0.0000
0.0000	0.0000
0.0000	0.0000
0.0000	0.0000
0.0000	0.0000
0.0000	0.0000
0.0000	0.0000
3.7220	0.0000
19.4300	0.0000
0.0000	0.0000
0.0000	0.0000
0.0000	0.0000

]

C1=[

-0.308	-0.196	-0.7597	-1.552	0.00	2.2160	-0.378	0.0616	-0.287	0.4000
0.0000	0.0000	0.0000	0.0050	1.0000	0.0000	0.0000	0.0045	0.0000	0.000

]

C2=[

0.0000	-0.8537	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

]

C=[C1 C2]

D=[

0	0
0	0

]

I.2 Weighting Functions

The weighting functions are defined as:

$$w_1(s) = \frac{0.01(s+70)^2}{(s+0.2199)^2}$$
$$w_2(s) = 0.001$$
$$w_3(s) = \frac{100(s+2.268)^2}{(s+226.757)}$$

I.3 Coordinated Tuning PSS-Speed Governor Matlab Script

I.3.1 Augmentation Script

The augmentation script (`plantaug.m`) is used to augment the plant.

```
% plantaug.m
% load the plant state-space description contained in
% in the matlab script plant.m
load plant;

% Specify the weighting functions
numw1=1000*conv([0.0144 1],[0.0144 1]);
denw1=conv([4.547 1],[4.547 1]);

numw3=0.01*conv([0.441 1],[0.441 1]);
denw3=conv([0.00441 1],[0.00441 1]);


w1=[numw1;denw1;numw1;denw1];
w2=[0.001;1;0.001;1];
w3=[numw3;denw3;numw3;denw3];

%Augment the plant
[AA,B1,B2,C1,C2,D11,D12,D21,D22] = augtf(A,B,C,D,w1,w2,w3);

% Save the augmented plant. This is used by the optimization
% function
save augplantmat.m
```

I.3.2 Objective function

This function is called for each iteration by the primary optimization script "pbilcoordobj.m". It takes values for the unknown controller parameters and calculates the minimum damping and maximum singular value for the closed loop response.

```
%pbilcoordobj.m
function [sv11,sv12,sv21,sv22,fudamp]=pbilcoordobj(pss,gov)

%load the augmented plant
load plantaugmat.m -mat

% define the frequency range for evaluation
w=logspace(-2.5,2.2,20000);

% define PSS structure
connum1=pss(1)*conv([1 pss(2)],[1 pss(3)]);
conden1=conv([1 pss(4)],[1 pss(5)]);
connum2=gov(1)*([1 gov(2)]);
conden2=conv([1 gov(3)],[1 gov(4)]);
[E,F,G,H]=tf2ss(connum1,conden1);
[I,J,K,L]=tf2ss(connum2,conden2);

%create closed loop matrices
sizeE = size(E);
z1matrix=zeros(sizeE(1),sizeE(2));
%create closed loop matrices
sizeI = size(I);
z1matrix=zeros(sizeI(1),sizeI(2));

A11 = (AA+B2(:,1)*H*C2(1,:)+B2(:,2)*L*C2(2,:));
A12 = (B2(:,1)*G);
A13 = (B2(:,2)*K);
A21 = (F*C2(1,:))
A22 = E
A23 = z1matrix;
A31 = (J*C2(2,:));
A32 = z2matrix;
A33 = I;

AT=[  A11 A12 A13
      A21 A22 A23
      A31 A32 A33];

B11 = B1(:,1)+(B2(:,1)*H*D21(1,1))+(B2(:,2)*L*D21(2,1));
B12 = B1(:,2)+(B2(:,1)*H*D21(1,2))+(B2(:,2)*L*D21(2,2))1
B21 = F*D21(1,1);
B22 = F*D21(1,2);
B31 = J*D21(2,1);
B32 = J*D21(2,2);
```

```
BT=[B11 B12
    B21 B22
    B31 B32];

CT11 = (C1+D12(:,1)*H*C2(1,:)+D12(:,2)*L*C2(2,:));
CT12 = (D12(:,1)*G);
CT13 = (D12(:,2)*K);
CT=[C11 C12 C13];

DT11 = D11(:,1)+(D12(:,1)*H*D21(1,1))+(D12(:,2)*L*D21(2,1));
DT12 = D11(:,2)+(D12(:,1)*H*D21(1,2))+(D12(:,2)*L*D21(2,2));
DT= [D11 D12];

%create objective function
[T11]=max(sigma(AT,BT(:,1),[CT(1,:);CT(5,:)],[DT(1,1);DT(5,1)],w));
[T12]=max(sigma(AT,BT(:,1),[CT(2,:);CT(6,:)],[DT(2,1);DT(6,1)],w));
[T21]=max(sigma(AT,BT(:,2),[CT(1,:);CT(5,:)],[DT(1,2);DT(5,2)],w));
[T22]=max(sigma(AT,BT(:,2),[CT(2,:);CT(6,:)],[DT(2,2);DT(6,2)],w));

% return the maximum singular value
sv11=max(T11);     sv22=max(T22);
sv12=max(T12);     sv21=max(T21);

% Calculate the damping factor
[x,y]=damp(AT);    fudamp=min(y);
```

I.3.3 Main Script

This is the main simulation script (coordopt.m). It uses the data and functions described in the above sections.

```
%coordopt.m
% Define the constants for PBIL algorithm
pos      =   0;
L        =  0.1 ;                  % learning rate
MP       =  .02 ;                  % mutation probability
NVARS    =   9 ;                   % number of variables
TRIALS   =   80 ;                  % number of trials per generation
MAXGEN   =   80 ;                  % number of generations
PREC     =   22 ;                  % precision (number of bits)
rand('seed',sum(100*clock))        % initialize random generator
len = NVARS*PREC;                  % number of bits in string
PV = 0.5*ones(1,len);              % probability vector init. to 0.5

global w AA B1 B2 C1 C2 D12 D21 D22 D11;
load plantaugmat.m -mat;
w=logspace(-2,2,2000);

% initialize storage vectors
QF = [];
QD= [];
% initialize variables
zever = 1*10^15 ;
dever =-1*10^15;
dmax =0;                    %  initialisation
zmax = 1*10^15;

for gen = 1:MAXGEN
   bmax =  zeros(1,PREC*NVARS) ;
   dmax = 0.01;

   for t = 1:TRIALS                % trial loop

      % generate random solution vector
      QB    = (rand(1,length(PV)) < PV );
      % extract parameters from solution vector
      QB1   =  QB(1:PREC);
      QB2   =  QB(PREC+1:2*PREC);
      QB3   =  QB(2*PREC+1:3*PREC);
      QB4   =  QB(3*PREC+1:4*PREC);
      QB5   =  QB(4*PREC+1:5*PREC);
      QB6   =  QB(5*PREC+1:6*PREC);
      QB7   =  QB(6*PREC+1:7*PREC);
      QB8   =  QB(7*PREC+1:8*PREC);
      QB9   =  QB(8*PREC+1:9*PREC);
```

```
      % create weights for each bit (real number between 0 and 1)
      P    = (2 .^((PREC-1):-1:0))/2^PREC;

      % convert binary data to a real number by weighting with P
      qd1  = 10^-5+P*QB1';
      qd2  = 10^-5+P*QB2';
      qd3  = 10^-5+P*QB3';
      qd4  = 10^-5+P*QB4';
      qd5  = 10^-5+P*QB5';
      qd6  = 10^-5+P*QB6';
      qd7  = 10^-5+P*QB7';
      qd8  = 10^-5+P*QB8';
      qd9  = 10^-5+P*QB9';

      % scale parameters
      pss1 = qd1*(100);            % scale into input
      pss2 = qd2*(100);
      pss3 = qd3*(100);
      pss4 = qd4*(100);
      pss5 = qd5*(100);
      gov1 = qd6*(100);            % scale into input
      gov2 = qd7*(100);
      gov3 = qd8*(100);
      gov4 = qd9*(100);
      % construct solution vector for unknown PSS parameters
      pss=[pss1 pss2 pss3 pss4 pss5 ];
      gov=[gov1 gov2 gov3 gov4 ];
      %Objective Function
      [sv11,sv12,sv21,sv22,opdamp] = pbilcoordobj(pss,gov);
      z = max([sv11 sv22]);

      if opdamp>= dmax %dmax
            cmax = [pss gov]
            bmax = QB;
            dmax = opdamp
            zmax = z
      end
   end
   QF = [QF,zmax];
   QD = [QD,dmax];
   PV = (1-L)*PV + L*bmax;   % update probability vector

   % mutate probability vector
   for k = 1: length(PV)
      if rand < MP
         PV(k) =  (1-L)*PV(k) + L*(rand < 0.5) ;
      end
   end
   if dmax >=dever
      dever = dmax;
      zever = zmax;
      xxbest=[pss gov];
   end
end
```