A PARTICLE SWARM OPTIMIZATION APPROACH FOR TUNING OF SISO PID CONTROL LOOPS

NELENDRAN PILLAY

2008

A PARTICLE SWARM OPTIMIZATION APPROACH FOR TUNING OF SISO PID CONTROL LOOPS

By

Nelendran Pillay

Student Number: 19752683

Thesis submitted in compliance with the requirements for the

Master's Degree in Technology: Electrical Engineering – Light Current

DURBAN UNIVERSITY OF TECHNOLOGY

DEPARTMENT OF ELECTRONIC ENGINEERING

This thesis represents my own work

N. Pillay

APPROVED FOR FINAL SUBMISSION

Supervisor: Dr. P. Govender Date

Dept. of Electronic Engineering

Durban University of Technology

Table of Contents

ABSTRACT

Linear control systems can be easily tuned using classical tuning techniques such as the Ziegler-Nichols and Cohen-Coon tuning formulae. Empirical studies have found that these conventional tuning methods result in an unsatisfactory control performance when they are used for processes experiencing the negative destabilizing effects of strong nonlinearities. It is for this reason that control practitioners often prefer to tune most nonlinear systems using trial and error tuning, or intuitive tuning. A need therefore exists for the development of a suitable tuning technique that is applicable for a wide range of control loops that do not respond satisfactorily to conventional tuning.

Emerging technologies such as Swarm Intelligence (SI) have been utilized to solve many non-linear engineering problems. Particle Swarm Optimization (PSO), developed by Eberhart and Kennedy (1995), is a sub-field of SI and was inspired by swarming patterns occurring in nature such as flocking birds. It was observed that each individual exchanges previous experience, hence knowledge of the "*best position*" attained by an individual becomes globally known. In the study, the problem of identifying the PID controller parameters is considered as an optimization problem. An attempt has been made to determine the PID parameters employing the PSO technique. A wide range of typical process models commonly encountered in industry is used to assess the efficacy of the PSO methodology. Comparisons are made between the PSO technique and other conventional methods using simulations and *real-time* control.

ACKNOWLEDGEMENTS

The work presented in this thesis was carried out under the supervision of Dr. P. Govender. My gratitude and sincere appreciation goes out to him for his assistance, valuable contribution and guidance throughout the study. His challenging questions and imaginative input greatly benefited the work.

I also wish to express my sincere gratitude and appreciation to the following:

The National Research Foundation (NRF) for their financial assistance.

Mr. C. Reinecke from the Durban University of Technology (DUT): Department of Electronic Engineering, for his sound advice and valuable suggestions.

Mr. K. Moorgas for his technical assistance regarding dedicated PC equipment.

A special note of thanks to my parents for their unwavering support and guidance.

Finally, thanks to Theresa Padayachee for your tolerance and patience while I spent many hours glued in front of the PC. You have shown great understanding and given selfless support.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

ACO	Ant Colony Optimization
AH	Åström and Hägglund
AI	Artificial Intelligence
ANN	Artificial Neural Network
CC	Cohen and Coon
DCS	Distributed Control System
DO	De Paor and O’ Malley
EC	Evolutionary Computation
EP	Evolutionary Programming
ES	Evolutionary Strategies
FODUP	First order delayed unstable process
FOPDT	First order plus dead time
GA	Genetic Algorithms
GP	Genetic Programming
IAE	Integral absolute-error criterion
ISE	Integral square-error criterion
ITSE	Integral-of-time multiplied square-error criterion
ITAE	Integral-of-time-multiplied absolute-error criterion
MIMO	Multiple-Input-Multiple-Output
MISO	Multiple-Input-Single-Output
P	Proportional controller

PI	Proportional-integral controller
PID	Proportional-integral-derivative controller
PP	Poulin and Pomerleau
PLC	Programmable Logic Controller
PSO	Particle Swarm Optimization
SOIPDT	Second order integrating plus dead time
SOPDT	Second order plus dead time
SI	Swarm Intelligence
SIMO	Single-Input-Multiple-Output
SISO	Single-Input-Single-Output
VC	Venkatashankar and Chidambaram
ZN	Ziegler and Nichols

LIST OF SYMBOLS

λ	Unstable pole
ε	Self regulating index / Controllability ratio
ϕ_m	Phase margin
$\theta(s)$	Angular displacement
σ	Standard deviation
σ_{ITAE}	Standard deviation of the ITAE index for the trial
ω_c	Gain crossover frequency
ω_p	Phase crossover frequency
$\bar{x}$	Mean
$\bar{x}_{ITAE}$	Mean value of the ITAE index for the trial
$\bar{x}_{Iter}$	Mean number of iterations used to perform a search
$\bar{x}_{Time}$	Mean time taken by PSO to complete a search
χ	Constriction factor
A_m	Gain margin
b	Controller bias
c_1	Cognitive acceleration constants (self confidence)
c_2	Social acceleration constant (swarm confidence)
$C(s)$	Angular displacement of motor shaft
d	Relay amplitude
D(s), $d(t)$	Disturbance
$E(s)$, $e(t)$	Error

e_{ss}	Steady-state error
f	Viscous friction coefficient of the motor and load
$G_c(s)$	Controller transfer function
$G_p(s)$	Process model transfer function
$gbest$	Global best of the population
$gbest_n$	Global best of the population for n dimension
h	Unit step function
$iter$	Current iteration
$iter_{\max}$	Maximum number of iterations
J	Moment of inertia of motor and load
K	Motor torque constant
K_b	Back emf constant
K_c	Proportional gain
K_d	Derivative gain
K_i	Integral gain
K_p	Process gain
K_u	Ultimate gain
L_a	Armature inductance
L_p	Process dead time
$L(s)$	Loop transfer function.
M_r	Maximum peak resonance

M_p (%)	Maximum percentage overshoot
n	Number of dimensions to problem
N	Gear ratio
p	Number of particles in population
PB	Proportional band
$pbest$	Personal best of agent
$pbest_{i,n}$	Personal best of agent i for n dimension
P_u	Ultimate period
q	Number of parameters being optimized by PSO
R_a	Armature resistance
$R(s)$, $r(t)$	Setpoint
$rand_{1,2}$	Random number between 0 and 1
$S(s)$	Sensitivity function
S_s	Swarm size
s^{k+1}	Modified searching point
$s_{i,n}^{k}$	Current position of agent i at iteration k for n dimension
$s_{i,n}^{(k+1)}$	Position of agent i at iteration $(k+1)$ for n dimension
T	Sampling interval
t_c	Period of relay
T_d	Derivative time constant
t_{dist}	Time of unit step disturbance

T_i	Integral time constant
T_p	Process time constant
t_r	Rise-time (10% to 90%)
t_s	Settling time (2%)
t_{ustep}	Time of input unit step
$U(s)$, $u(t)$	Controller output
u_{proc}	Process input
$V_{\max}$	Velocity maximum
v_{pbest}	Velocity based on $pbest$
v_{gbest}	Velocity based on $gbest$
v^{k+1}	Modified velocity
$v_{i,n}^{k}$	Velocity of agent i at current iteration k for n dimension
$v_{i,n}^{(k+1)}$	Velocity of agent i at iteration $(k+1)$ for n dimension
w	Inertia weight
$w_{\max}$	Initial weight
$w_{\min}$	Final weight
$Y(s)$, $y(t)$	Process output

Chapter 1

Introduction and Overview of the Study

1.1 Introduction

The PID controller is regarded as the workhorse of the process control industry (Pillay and Govender, 2007). Its widespread use and universal acceptability is attributed to its simple operating algorithm, the relative ease with which the controller effects can be adjusted, the broad range of applications where it has reliably produced excellent control performances, and the familiarity with which it is perceived amongst researchers and practitioners within the process control community (Pillay and Govender, 2007). In spite of its widespread use, one of its main short-comings is that there is no efficient tuning method for this type of controller (Åström and Hägglund, 1995). Given this brief background, the main objective of this study is to develop a tuning methodology that would be universally applicable to a range of popular processes that occur in the process control industry.

1.2 Motivation for the study

Several tuning methods have been proposed for the tuning of process control loops, with the most popular method being that of Ziegler and Nichols (1942). Other methods include the methods of Cohen and Coon (1953), Åström and Hägglund (1984), De Paor and O'Malley (1989), Zhuang and Atherton (1993), Venkatashankar and Chidambaram (1994), Poulin and Pomerleau (1996) and Haung and Chen (1996). In spite of this large range of tuning techniques, to date there still seems to be no general consensus as to which tuning method works best for most applications (Lipták, 1995). Some methods rely heavily on experience, while others rely more on mathematical considerations (Lipták, 1995).

The Ziegler-Nichols method (1942) is the method most preferred by process control practitioners and alternate methods are often not applied in practice because of the reluctance of control personnel to learn new techniques which they perceive as being complicated, time consuming and laborious to implement (Pillay and Govender, 2007). Also, some commonly used techniques do not perform sufficiently well in the presence of strong nonlinear characteristics within the control channel (Åström and Hägglund, 2004, Shinskey, 1994).

1.3 Focus of the study

This study proposes the development of a tuning technique that would be suitable for optimizing the control of processes operating in a single-input-single-output (SISO) process control loop. The SISO topology has been selected for this study because it is the most fundamental of control loops and the theory developed for this type of loop can be easily extended to more complex loops. The research focuses on utilizing a soft-computing strategy, namely the particle swarm optimization (PSO) technique that was first proposed by Kennedy and Eberhart (1995), as an optimization strategy to determine optimal controller parameters for PID control and its variants. The control performance of loops tuned with the proposed PSO technique will also be compared to that of loops tuned using another soft-computing technique, namely the genetic algorithm (GA) plus the methods mentioned previously in the discussions. The GA was selected for comparison with the PSO because both are population based soft-computing techniques.

1.4 Objectives of the study

The objectives of the study are to:

i) Develop a PSO based PID tuning methodology for optimizing the control of SISO process control loops.

ii) Determine the efficacy of the proposed method by comparing the control performance of loops tuned with the PSO method to that of loops tuned using the GA and the other so-called conventional methods of Ziegler-Nichols (1942), Cohen and Coon (1953), Åström and Hägglund (1984), De Paor and O'Malley (1989), Venkatashankar and Chidambaram (1994) and Poulin and Pomerleau (1996).

1.5 Thesis overview

This document is arranged as follows:

Chapter one gives an introduction and general overview of the study. It focuses on the research problem and motivation for the study.

Chapter two provides a brief outline on PID control and classical control theory.

Chapter three highlights typical process models that are commonly encountered in processes control loops. Typical nonlinear characteristics commonly found in most process control loops are reviewed and their effects on controller tuning and closed-loop performance are also explored in this chapter.

Chapter four reviews selected PID controller tuning algorithms proposed in the literature.

Chapter five discusses soft computing techniques such as evolutionary computation (EC) and compares the intrinsic characteristics of GA's to that of the PSO.

Chapter six discusses the PSO tuning approach.

Chapter seven describes a simulation that study focuses on the effects of PSO parameter variation.

Chapter eight describes a simulation study that compares the control performance of PSO tuned systems to that of systems tuned using methodologies proposed in the literature. This chapter also compares the control performance of PSO tuned systems to GA tuned systems.

In *Chapter nine* the PSO method is applied *offline* to tune process control loops.

Chapter ten describes the *real-time* control of a positional servo-mechanism.

Chapter eleven summarizes the findings of the study and provides direction for further research that could be pursued in the field.

Appendix A provides the PSO source code used in all the experiments.

Appendix B gives details of the experiments conducted in Chapter 9.

Appendix C provides the loop diagram associated with the process control plant and details all the experiments conducted for the PSO and GA tuning methods.

Appendix D presents two conference papers and a draft journal paper arising from the work conducted in this study.

Chapter 2

Overview of PID Control

2.1 Introduction

The PID controller is by far the most commonly used controller strategy in the process control industry (Åström and Hägglund, 1995; Åström *et al.*, 2004). Its widespread use is attributed to its simple structure and robust performance over a wide range of operating conditions (Gaing, 2004). PID control is implemented as either stand-alone control, or on DCS, SCADA and PLC control systems. The popularity and widespread use of PID control in the process control industry necessitates a detailed discussion on the fundamental theory that underpins this type of three-term process control. The dynamics associated with each control mode will also be discussed and the advantages and shortcomings associated with each type of control will also be given.

2.2 Control Effects of Proportional, Integral and Derivative Action

2.2.1 Proportional control

Proportional control is defined as the control action that occurs in direct proportion with the system error. The output of a proportional controller varies proportionally to the system error according to (2.1):

$$u_p(t) = K_c e(t) + b \qquad \text{Equation (2.1)}$$

With regards to (2.1), $u_p(t)$ is the controller output, $e(t)$ is the error, b is the controller bias and K_c is the controller gain (referred to as the proportional gain). Proportional control action responds to only the present error. For a small value of proportional gain, a large error yields a small corrective control action. Conversely, a large proportional gain will result in a small error and hence a large control signal. The controller bias is necessary in order to ensure that a minimum control action is always present in the control loop.

The gain of a proportional controller is usually described in terms of its proportional band (*PB*). The concept of the proportional band is inherited from pneumatic controller and is defined as:

$$PB = \frac{1}{K_c} \times 100\% \qquad \text{Equation (2.2)}$$

From (2.2), a large proportional gain K_c corresponds to a small proportional band *PB*, while a large *PB* implies a small gain K_c. A pure P controller reduces error but does not eliminate it (unless the process has naturally integrating properties). With pure P control an offset between the actual and desired value will normally exist. This is illustrated as follows:

Consider Figure 2.1:

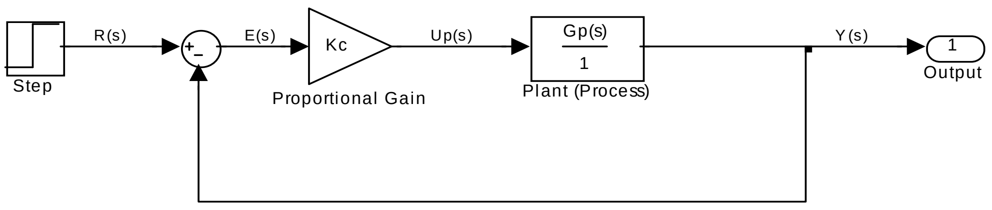


Figure 2.1: Proportional controller within a closed-loop feedback control system

With regards to Fig. 2.1:

The closed-loop transfer function of this control system is represented by (2.3):

$$\frac{Y(s)}{R(s)} = \frac{K_c G_p(s)}{1 + K_c G_p(s)} \qquad \text{Equation (2.3)}$$

where $G_p(s)$ is the transfer function of the process, *R(s)* and *Y(s)* represents the input and output of the process, respectively and the error signal *E(s)* is:

$$E(s) = \frac{R(s)}{1 + K_c G_p(s)} \qquad \text{Equation (2.4)}$$

The action of the proportional controller usually results in an offset i.e. the difference between the desired output and the actual output of the system for processes that do not have any inherent integrating properties. Under these conditions the steady-state error for the control system can be calculated using the final value theorem (2.5):

$$e_{ss}(+\infty) = \lim_{s \to o} [sE(s)] \qquad \text{Equation (2.5)}$$

For a unit step input:

$$e_{ss}(+\infty) = \lim_{s \to o} \left(s \frac{1}{s} \frac{1}{1 + K_c G_p(s)} \right) = \lim_{s \to 0} \left(\frac{1}{1 + K_c G_p s} \right) = \frac{1}{1 + K_c G_p s} \qquad \text{Equation (2.6)}$$

This indicates the presence of a steady state error for $G_p(s) \neq \pm\infty$, which is the case for systems with no inherent integrating properties. From (2.6), the absolute value of the steady-state error can be reduced by sufficiently increasing K_c. However since K_c affects system stability and its dynamics, it will be limited by the stability constraints of the overall control system. A high value of K_c may lead to oscillations and large overshoots which could result in instability (See Figure 2.2).

It is for this reason that proportional control is often combined with integral control in order to eliminate offset, while applying the smaller values of the gain K_c. A typical example of system response using only proportional control is illustrated in Figure 2.2.

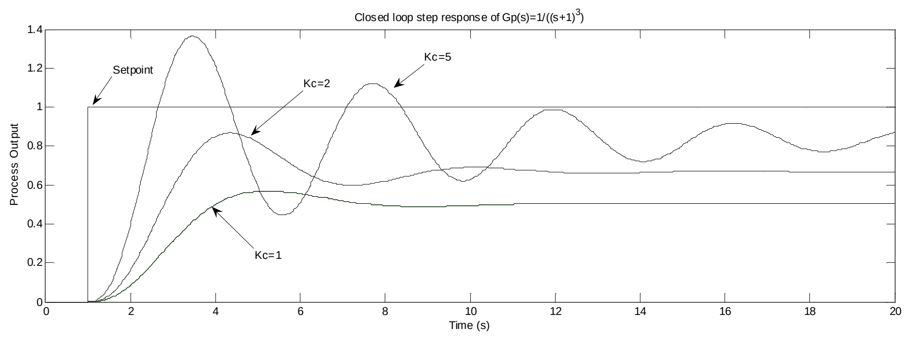


Figure 2.2: Control effect of varying P-action $\left(G_p(s) = \frac{1}{(s+1)^3} \right)$

2.2.2 Integral control (Reset control)

Integral control is used in systems where proportional control alone is not capable of reducing the steady-state error within acceptable bounds. Its primary effect on a process control system is to permanently attempt to gradually eliminate the error. The action of the integral controller is based on the principle that the control action should exist as long as the error is different from zero, and it has the tendency to gradually reduce the error to zero. The integrator control signal ($u_i\ (t)$) is proportional to the duration of the error and is given by:

$$u_i(t) = \frac{K_c}{T_i} \int_{t_i}^{t_f} e(t)\ dt = K_i \int_{t_i}^{t_f} e(t)\ dt \qquad \text{Equation (2.7)}$$

With regards to (2.7): T_i is the integral time constant, K_c is the proportional gain, $K_c / T_i = K_i$ is the gain of the integral controller, $e(t)$ is the instantaneous error signal and the limits t_i and t_f represent the start and end of the error, respectively. The smaller the integral time constant, the more often the proportional control action is repeated, therefore resulting in greater integral contribution toward the control signal. For a large integral time constant, the integral action is reduced. Integral control can be seen as continuously looking at the total past history of the error by continuously integrating the area under the error curve and reducing any offset. The greater the error signal the larger the correcting action from the integral controller will be.

2.2.2.1 Integral action as automatic reset

Integral action may be performed as a kind of automatic reset (see Figure 2.3) and is equivalent to permanently adjusting the bias of the proportional controller.

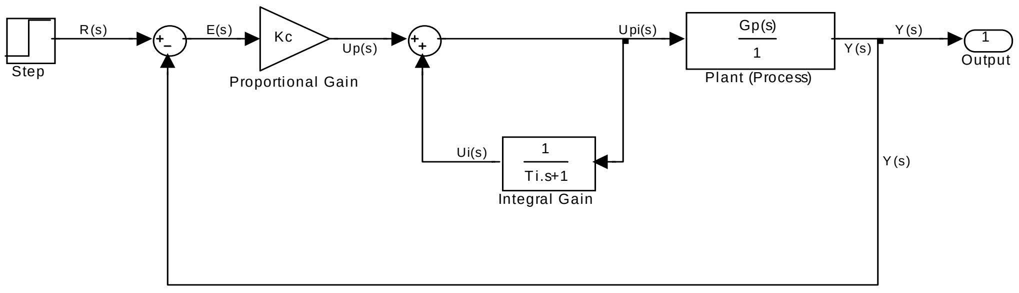


Figure 2.3: Proportional controller with an integrator as automatic reset

With regards to Figure 2.3, the control signal applied to the process is:

$$U_{pi}(s) = K_c E(s) + U_i(s) \qquad \text{Equation (2.8)}$$

and

$$U_i(s) = \frac{U_{pi}(s)}{1 + T_i s} \qquad \text{Equation (2.9)}$$

Substituting (2.9) into (2.8) yields:

$$U_{pi}(s) = K_c E(s) + \frac{U_{pi}(s)}{1+T_i s} = U_{pi}(s) - \frac{U_{pi}(s)}{1+T_i s} = K_c E(s)$$

$$U_{pi}(s)\left(1 - \frac{1}{1+T_i s}\right) = K_c E(s)$$

$$U_{pi}(s)\left(\frac{1+T_i s}{1+T_i s} - \frac{1}{1+T_i s}\right) = K_c E(s)$$

$$\frac{U_{pi}(s)}{E(s)} = K_c\left(\frac{1+T_i s}{T_i s}\right) = \frac{K_c}{T_i s} + \frac{K_c T_i s}{T_i s} = \frac{K_c}{T_i s} + K_c = \frac{K_i}{s} + K_c$$

and

$$U_{pi}(s) = \left(\frac{K_i}{s} + K_c\right)E(s) \qquad \text{Equation (2.10)}$$

where $\frac{K_i}{s}.\,E(s)$ and $K_c E(s)$ represents the control action of the integral and proportional controller on the error signal, respectively.

Proportional action comes into effect immediately as an error different from zero occurs. If the proportional gain is sufficiently high it will drive the error closer to zero. Integral control accomplishes the same control effect as the proportional control but with an infinitely high gain. This results in the offset eliminating property of integral action which can be illustrated by applying the final value theorem to the control structure of Figure 2.3. With regards to Figure 2.3:

$$E(s) = \frac{R(s)}{1+G_p(s)G_c(s)} \qquad \text{Equation (2.11)}$$

where $G_c(s) = K_c + \dfrac{K_i}{s}$ and $R(s) = \dfrac{1}{s}$. From (2.12) the integral controller drives the error to zero:

$$e_{ss}(+\infty) = \lim_{s \to 0}[sE(s)] = \lim_{s \to 0}\left[s \times \frac{1}{s} \times \frac{1}{1 + G_p(s)G_c(s)} \right] = \lim_{s \to 0}\left[\frac{s}{s + (K_c s + K_i)G_p(s)} \right] = 0$$

Equation (2.12)

$e_{ss}(+\infty) = 0$ indicates that the offset is zero and proves that integral action eliminates any offset. The control effects of integral action are illustrated in Figure 2.4. With regards to Figure 2.4, the proportional gain is kept constant ($K_c = 1$) and the integral time is adjusted to illustrate the effects of the integral time constant.

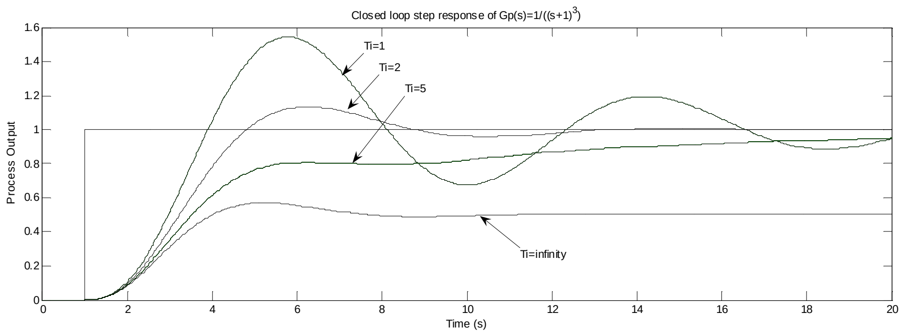


Figure 2.4: Control effects of varying integral action $\left(G_p(s) = \dfrac{1}{(s+1)^3} \right)$

The integral time (*T_i*) constant is varied within the range $T_i = [1,2,5,\infty]$. The case when $T_i = \infty$, corresponds to pure proportional control and is identical to *K=1* in Figure 2.2, where the steady-state error is 50%. The steady-state error is removed when T_i has finite value. For large values of the integration time constant, the response gradually moves towards the setpoint. For small values of T_i, the response is faster but oscillatory.

2.2.2.2 Undesirable effects of Integral Control

Although integral control is very useful for removing steady-state errors it is also responsible for sometimes introducing undesirable effects into the control loop in the form of increase settling time, reduced stability and integral windup (Govender, 1997). A short explanation of each of these undesirable effects is discussed.

Increased settling time: An increase of the closed-loop system settling time is usually caused by the increased oscillations as a consequence of the present integral action.

Reduced stability: The presence of the integral action may lead to increased oscillations within the control loop. These oscillations generally have a tendency to move the system towards the boundary of instability. In some cases these oscillations will result in the loop becoming unstable.

Integral windup: Integrator windup occurs when the integral controller calls for a control action that the process actuator cannot produce because of its saturated state. This so-called *integrator windup* state results in severe overshoots in the controlled variable.

2.2.3 Derivative control (Rate or Pre-Act control)

In linear proportional control the strength of the control action is directly proportional to the magnitude of the error signal and P-action becomes assertive only when a significant error has occurred. The integral controller performs corrective action for as long as an error is present but its gradual ramp shaped response means that significant time expires before it produces a sizeable control response. Both these control modes are incapable of responding to the rate of change of the error signal.

D-control action positively enhances system closed-loop stability (Åström and Hägglund, 1995). When operating in the forward path, the derivative controller responds to the rate at which system error changes according to (2.13a):

$$u_d(t) = K_c T_d \frac{de(t)}{dt} = K_d \frac{de(t)}{dt} \qquad \text{Equation (2.13a)}$$

With regards to (2.13a): $\frac{K_c}{T_d} = K_d$ is the derivative gain, T_d denotes the derivative time constant and $\frac{de(t)}{dt} = De(t)$ is the rate of change of the error feedback signal. From (2.13a) and (2.13b) it is obvious that D-action is only present when the error is changing; for any static error the contribution of the D-controller will be zero. Derivative action on its own will therefore allow uncontrolled steady-state errors. It is for this reason that derivative control is usually combined with either P-control or PI control.

Another shortcoming of the D-controller is its sensitivity. The D-controller can be regarded as a high-pass filter that is sensitive to set-point changes and process noise when operating in the forward path (Lipták, 1995). To reduce this sensitivity, it is quite common to find the D-controller operating in the feedback loop enabling it to act on the feedback signal according to (2.13b):

$$u_d(t) = \frac{K_c}{T_d}\frac{dy(t)}{dt} = K_d\frac{dy(t)}{dt} = K_d Dy(t) \qquad \text{Equation (2.13b)}$$

With regards to (2.13b) $\frac{dy(t)}{dt} = Dy(t)$ represents the rate of change of the feedback signal; all the other terms have the same meaning as was defined for (2.13a).

2.2.3.1 D-Action as Predictive Control

The control action of a PD-controller can be interpreted as a type of predictive control that is proportional to the predicted process error. The prediction is performed by extrapolating the error from the tangent to the error curve in Figure 2.5. PD controllers operate according to control law (2.14):

$$u_{pd}(t) = K_c\left(e(t) + T_d\frac{de(t)}{dt}\right) \qquad \text{Equation (2.14)}$$

A Taylor series expansion of $e(t+T_d)$ gives:

$$e(t+T_d) \approx e(t) + T_d \frac{de(t)}{dt} \qquad \text{Equation (2.15)}$$

The PD control signal is thus proportional to an estimate of the control error at time T_d seconds ahead, where the estimate is obtained through linear extrapolation.

From Figure 2.5, the longer the derivative time constant T_d is set, the further into the future the D-term will predict. Derivative action depends on the slope of the error, hence if the error is constant the derivative action has no effect. The effects of derivative action on control performance are illustrated in Figure 2.6. The controller proportional gain and integrating time constant are kept constant, $K_c = 3$ and $T_i = 2$, and the derivative time is varied according to T_d = [0.1;0.7;4.5]. For T_d = 0 we have a pure PI control.

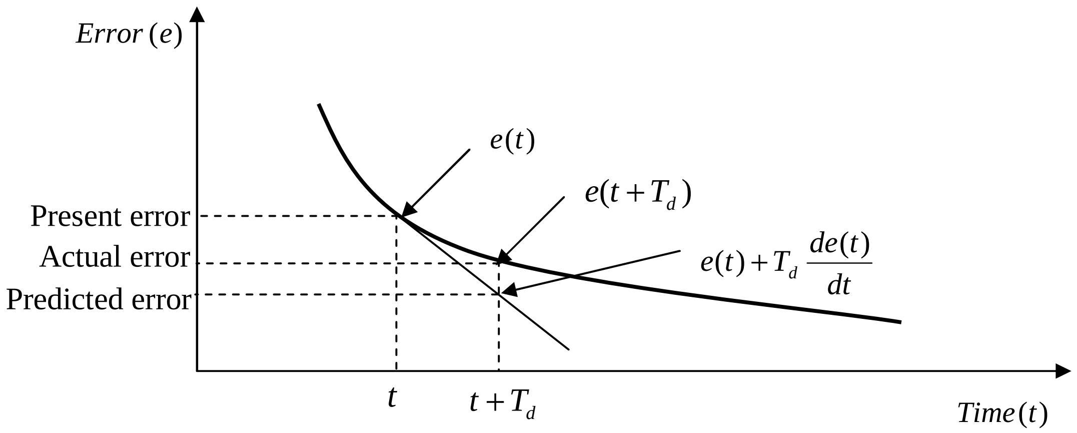


Figure 2.5: Interpretation of derivative action as predictive control

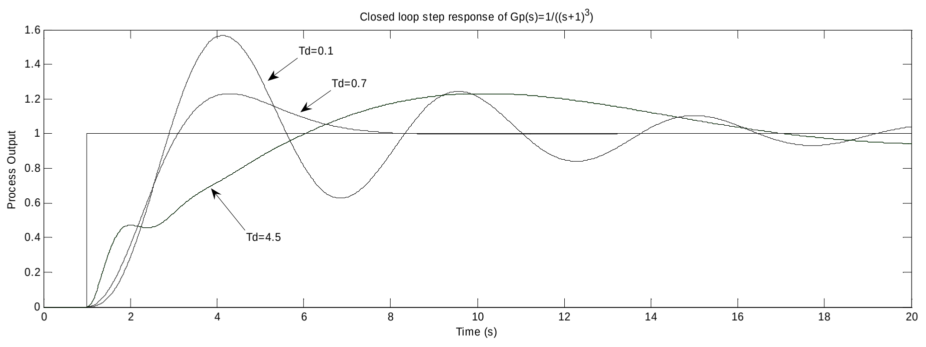


Figure 2.6: Simulation of a closed-loop system with PID control $\left(G_p(s) = \dfrac{1}{(s+1)^3} \right)$

From Figure 2.6, we observe that system response is oscillatory for low values for T_d and highly damped for higher derivative time settings.

2.3 PID Algorithms

The transfer functions for PID algorithms are classified as follows: standard non-interacting (2.16), series interacting (2.17) and parallel non-interacting PID (2.18).

$$\frac{U(s)}{E(s)} = K_c[1 + \frac{1}{T_i s} + T_d s] + b \qquad \text{Equation (2.16)}$$

Most tuning methods are based on (2.16) (Lipták, 1995).

$$\frac{U(s)}{E(s)} = K_c\left[\left(1 + \frac{1}{T_i s}\right)(1 + T_d s)\right] + b \qquad \text{Equation (2.17)}$$

$$\frac{U(s)}{E(s)} = \left(k_c + \frac{k_i}{s} + k_d s\right) + b \qquad \text{Equation (2.18)}$$

With regards to (2.16) – (2.18): $U(s)$ represents the control signal; $E(s)$ is the error signal; K_c denotes the proportional gain; T_i and T_d refers to the integral and derivative time constants; b denotes the controller bias. The implementation strategy for (2.16), (2.17) and (2.18) is shown in Figure 2.7, Figure 2.8 and Figure 2.9.

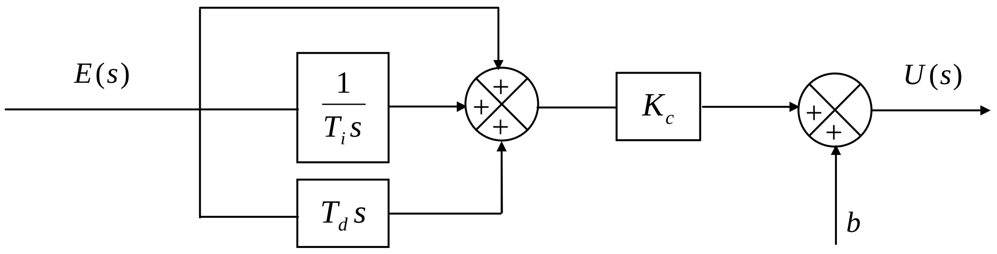


Figure 2.7: Non-interacting PID

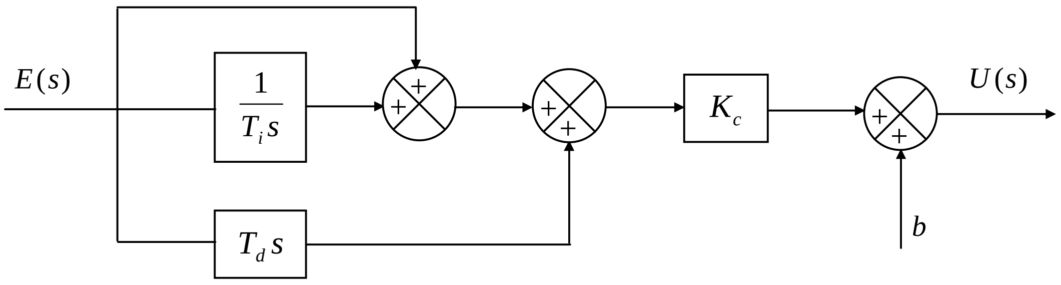


Figure 2.8: Interacting PID

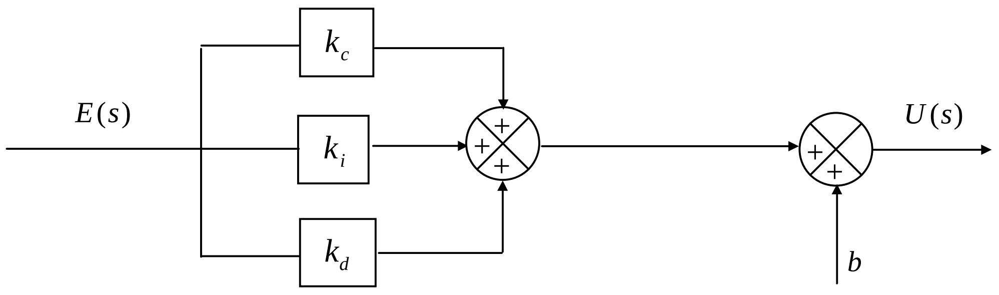


Figure 2.9: Parallel non-interacting PID

Historically, pneumatic controllers based on (2.17) were easier to build and tune (Åström and Hägglund, 1995). Note that the interacting and non-interacting forms are different only when both integral and derivative control actions are used. (2.16) and (2.17) are equivalent when the controller is utilized for P, PI or PD control. It is evident that in the interacting controller the derivative time does influence the integral part, hence the reasoning that it is interacting.

The representation for the parallel non-interacting PID controller is equivalent to the standard non-interacting controller with the exception that the parameters are expressed in a different form. The relationship between the standard and parallel type is given by $k_c = K_c$, $ki = Kc/Ti$ and $k_d = K_cT_d$. The parallel structure has the advantage of often being useful in analytical calculations since the parameters appear linearly. The representation also has the added advantage of being preferred for pure P, I or D control by the selection of finite tuning parameters (Åström, 1995).

2.4 Performance evaluation criteria

Quantification of system performance is achieved through a performance index. The performance selected depends on the process under consideration and is chosen such that emphasis is placed on specific aspects of system performance. Performances indices preferred by the control engineering discipline include the Integral Square-Error (ISE) index (2.19), Integral-of-Time multiplied by Square-Error (ITSE) index (2.20), Integral Absolute-Error (IAE) index (2.21) and the Integral-of-Time multiplied by Absolute-Error (ITAE) index (2.22).

ISE Index:

$$ISE = \int_0^\infty e^2(t)\,dt \qquad \text{Equation (2.19)}$$

An optimal system is one which minimizes this integral. The upper limit ∞ may be replaced by T which is chosen sufficiently large such that $e(t)$ for $T < t$ is negligible and the integral reaches a steady-state. A characteristic of this performance index is that it penalizes large errors heavily and small errors lightly. A system designed by this criterion tends to show a rapid decrease in a large initial error. Hence the response is fast and oscillatory leading to a system that has poor relative stability (Ogata, 1970).

ITSE Index:

$$ITSE = \int_0^\infty te^2(t)\,dt \qquad \text{Equation (2.20)}$$

This criterion places little emphasis on initial errors and heavily penalizes errors occurring late in the transient response to a step input.

IAE Index:

$$IAE = \int_0^\infty |e(t)|\, dt \qquad \text{Equation (2.21)}$$

Systems based on this index penalize the control error.

ITAE Index:

$$ITAE = \int_0^\infty t|e(t)|\, dt \qquad \text{Equation (2.22)}$$

System's designed using this criterion has small overshoots and well damped oscillations. Any large initial error to a step-response is penalized lightly whilst errors occurring later in the response are penalized heavily. The ITAE performance index is used in this study. A summary of the performance indices and their respective properties is shown in Table 2.1.

Performance Index	**Equation**	**Properties**
ISE	$ISE = \int_0^\infty e^2(t)dt$	Penalizes large control errors. Settling time longer than ITSE. Suitable for highly damped systems.
ITSE	$ITSE = \int_0^\infty te^2(t)dt$	Penalizes long settling time and large control errors. Suitable for highly damped systems.
IAE	$IAE = \int_0^\infty \lvert e(t)\rvert dt$	Penalizes control errors.
ITAE	$ITAE = \int_0^\infty t\lvert e(t)\rvert dt$	Penalizes long settling time and control errors.

Table 2.1: Summary of performance indices

2.5 Summary and conclusion

Typical PID algorithms that form the building blocks of controllers have been discussed. The control actions of proportional, integral and derivative terms and some of their adverse effects have also been reviewed. The proportional controller provides a corrective action that is proportional to the size of the error and also has an effect on the speed of a system’s response; integral control provides corrective action proportional to the time integral of the error and is present for the entire duration of the error; the derivative controller provides a corrective action proportional to the time derivative of the error signal and responds to the rate at which the error is changing. The effects of process dynamics on controller tuning are discussed in the next chapter.

Chapter 3

Typical Process Control Models

3.1 Introduction

This chapter presents a discussion on the transfer function models of systems commonly encountered in process control. These plant models will be used to compare the control performance of loops tuned with the PSO versus that of loops tuned using methodologies proposed in the literature. The dynamics associated with each process model is also discussed.

3.2 Dynamics associated with the selected process models

The SISO control loop used in this study is given in Figure 3.1. The SISO configuration has been chosen because it forms the fundamental building block of all process control loops and the dynamics associated with it are universally applicable to configurations such as SIMO, MISO and MIMO control loops.

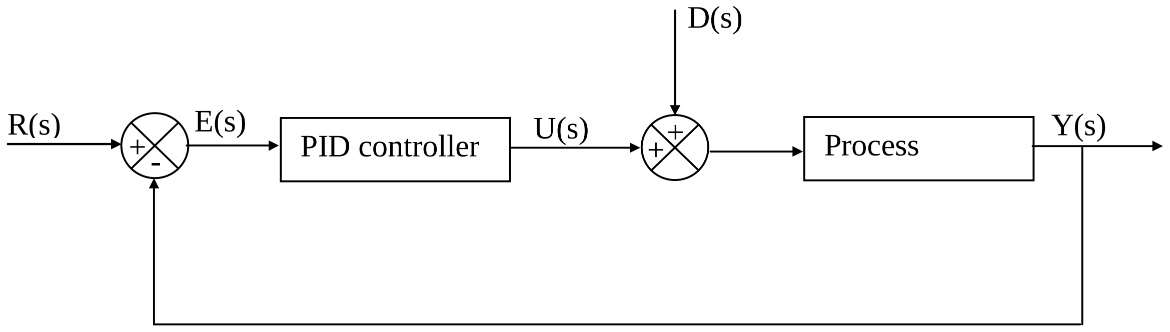


Figure 3.1: SISO system with unity feedback

Typical real-world process models that have been selected for this study are listed in (3.1) to (3.4):

A Stable First Order Plus Dead-Time Process (FOPDT):

$$G_p(s) = \frac{K_p \exp(-L_p s)}{(T_p s + 1)} \qquad \text{Equation (3.1)}$$

A Stable Second Order Plus Dead-Time Process (SOPDT):

$$G_p(s) = \frac{K_p \exp(-L_p s)}{(T_p s + 1)^2} \qquad \text{Equation (3.2)}$$

A Stable Second Order Integrating Process with Dead-Time (SOIPDT):

$$G_p(s) = \frac{K_p \exp(-L_p s)}{s(T_p s + 1)} \qquad \text{Equation (3.3)}$$

A First Order Delayed Unstable Process (FODUP):

$$G_p(s) = \frac{K_p \exp(-L_p s)}{(T_p s - 1)} \qquad \text{Equation (3.4)}$$

Equations (3.1)-(3.4) capture the typical dynamics that are present in most real-world process control systems, with the exception that the $\frac{L_p}{T_p}$ ratios may vary (Åström *et al.*, 2004). Equation (3.2) characterizes systems that are rich in dynamics and include systems such as underdamped, critically damped and overdamped systems. These systems usually follow an *“S-shape”* closed-loop response.

The $\frac{L_p}{T_p}$ ratio, or controllability ratio, is used to characterize the difficulty or ease of controlling a process. Processes having small controllability ratios (i.e. $0 \le \frac{L_p}{T_p} < 1$) are easier to control and the difficulty of controlling the system increases as the controllability ratio increases (Åström and Hägglund, 1995). Processes with $\frac{L_p}{T_p} \ge 1$ correspond to dead-time dominant processes that are difficult to control with conventional PID control (Åström, 1995).

3.3 A brief overview of integrating processes (Self-Regulating Processes)

Most real-world process control systems are characterized by offset or steady-state error which can arise from load friction, intrinsic steady state nonlinearities or uncertainties in modeling (Haung *et al.*, 1996). If the forward branch of a feedback control system contains an integrator, the presence of an error will cause a rate of change of output until the error has been eliminated (Chen *et al.*, 1996; Poulin and Pomerleau, 1996).

The dynamics of certain real-world process control systems are such that an inherent integrating control effect could naturally arise during normal operation of the plant. This "natural integrator" is purely error driven and will ensure that any steady-state error is driven to zero following either a setpoint change or disturbance. There is no static error to a setpoint change for pure proportional control. However this is not the case when nonzero mean disturbances act at the process input. Therefore in order to ensure that there will be no static error, a control with an integrator must be used (Poulin and Pomerleau, 1996).

3.4 Problems experienced with tuning processes having unstable poles and dead-time

Processes having only right-hand poles are inherently unstable under open-loop conditions (Poulin and Pomerleau, 1996; Majhi and Atherton, 1999). The undesirable effects of dead-time will contribute towards the instability inherently present in systems of this nature. The tuning of these open-loop unstable processes having dead-time delay becomes more challenging than for stable processes (Poulin and Pomerleau, 1996). The Ziegler-Nichols (1942) and Cohen-Coon (1953) tuning techniques are unsuitable for tuning loops that have only unstable pole/s plus dead-times because:

The open-loop step response of systems having unstable poles will be unbounded (Poulin and Pomerleau, 1996; Haung *et al.*, 1996). The Ziegler-Nichols and Cohen-Coon open-loop methods rely on a stable open-loop response for determining the controller's tuning parameters.

The Ziegler-Nichols closed-loop method operates the control-loop within the marginally stable region (see Figure 3.2) when the critical gain tuning parameter is being determined. Coupled with this, the destabilizing effects of the system’s right-hand pole/s plus channel dead-time may drive the system’s response into its unstable region of operation following a disturbance input.

Open-loop response trajectories of self-regulating, marginally stable and unstable processes are illustrated in Figure 3.2.

With regards to Figure 3.2:

Trajectory a: Open-loop stable response

Trajectory b: Open-loop marginally stable process. Closed-loop control will push system response into the stable operating region.

Trajectory c: Open-loop unstable process. With closed loop-control and properly tuned control, system response could be forced into the marginally stable or stable operating region.

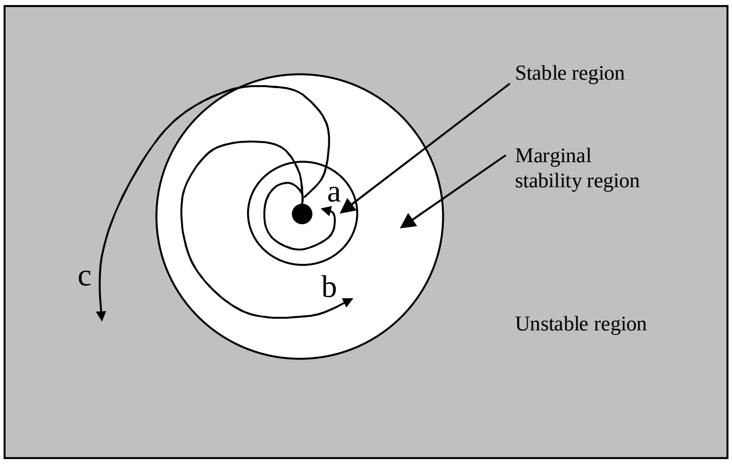


Figure 3.2: Response trajectories for self-regulating (stable), marginally stable and unstable processes

3.5 Performance limitations of PID controllers for unstable processes having $\frac{L_p}{T_p} > 1$

The existence of right-hand pole/s in the open-loop transfer function of a process may lead to limitations in its closed-loop performance (Haung and Chen, 1996). Traditional methods using the sensitivity function $S(s)$ are used to express the limitations caused by the presence of right-hand poles.

If the process has open-loop unstable poles, the response of the closed-loop system will overshoot the setpoint in all cases (Youla *et al.*, 1976). Closed-loop performance is also compromised when the combined effects of long dead-time and unstable pole/s are simultaneously present within the system's control channel (Govender, 2003). Added to this, the ability of a control system to reject load disturbances will degrade if the process contains unstable pole/s (Huang and Chen, 1996). A well-tuned PI controller will stabilize a FODUP process if and only if the controllability ratio $\frac{L_p}{T_p} < 1$ (De Paor and O' Malley, 1989, Venkatashankar and Chidambaram, 1994). For the PID controller the constraint is relaxed to $\frac{L_p}{T_p} < 2$ (Huang and Chen, 1996; Lee *et al.*, 2000).

3.5 Summary and conclusion

This chapter has focused on typical processes and their respective dynamics. Some of the challenges experienced when tuning open-loop processes and processes having a high controllability ratio have also been mentioned. There is no universal tuning algorithm or tuning methodology that is suitable for all processes. This is evident by virtue of the fact that following Ziegler and Nichols (1942), various researchers have proposed tuning approaches that are applicable to specific process types (Cohen-Coon, 1953; Åström and Hägglund, 1984; De Paor and O'Malley, 1989; Zhuang and Atherton, 1993; Venkatashankar and Chidambaram, 1994; Poulin and Pomerleau, 1996; Huang and Chen, 1996). The next chapter discusses selected tuning methodologies applicable to this study.

Chapter 4

PID Tuning

4.1 Introduction

The dynamical nature of process control loops leads to changes of operating conditions within the loop, and hence loop performance. Changes in system performance may be attributed to the presence of process nonlinearities within the control channel, process aging, production strategy changes, modifications to the properties of raw materials, and changes over equipment maintenance cycles (Pomerleau and Poulin, 1996). Given these dynamical conditions, loop tuning is necessary to ensure the continued satisfactory performance of the control loop.

The goal of PID controller tuning is to determine parameters that meet closed loop system performance specifications, and the robust performance of the control loop over a wide range of operating conditions should also be ensured. Practically, it is often difficult to simultaneously achieve all of these desirable qualities. For example, if the PID controller is adjusted to provide better transient response to set point change, it usually results in a sluggish response when under disturbance conditions. On the other hand, if the control system is made robust to disturbance by choosing conservative values for the PID controller, it may result in a slow closed loop response to a setpoint change. A number of tuning techniques that take into consideration the nature of the dynamics present within a process control loop have been proposed (see Ziegler and Nichols, 1942; Cohen and Coon, 1953; Åström and Hägglund, 1984; De Paor and O’Malley, 1989; Zhuang and Atherton, 1993; Venkatashankar and Chidambaram, 1994; Poulin and

Pomerleau, 1996; Huang and Chen, 1996). All these methods are based upon the dynamical behavior of the system under either open-loop or closed-loop conditions. These tuning methods are discussed in the following sections.

4.2 Ziegler-Nichols Tuning

The earliest known and most popular tuning methodology was proposed by Ziegler and Nichols (ZN) in 1942 (Åström and Hägglund, 2004). They proposed the closed-loop (or ultimate sensitivity) method and the open-loop (or process reaction curve) method. The ZN tuning rules has a serious shortcoming in that it uses insufficient process information to determine the tuning parameters (Åström and Hägglund, 2004). This disadvantage leads to system performances that have poor robustness (Åström and Hägglund, 2004).

The Ziegler-Nichols tuning method is based on the determination of processes inherent characteristics such as the process gain (K_p), process time constant (T_p) and process dead time (L_p). These characteristics are used to determine the controller tuning parameters. Although the Ziegler-Nichols methods attempt to yield optimum settings, the only criterion stated is that the response has a decay ratio of quarter (see Figure 4.1) (Ziegler and Nichols, 1942). This is viewed as a shortcoming because a controller tuned with this criterion may not be at its optimal setting (Lipták, 1995).

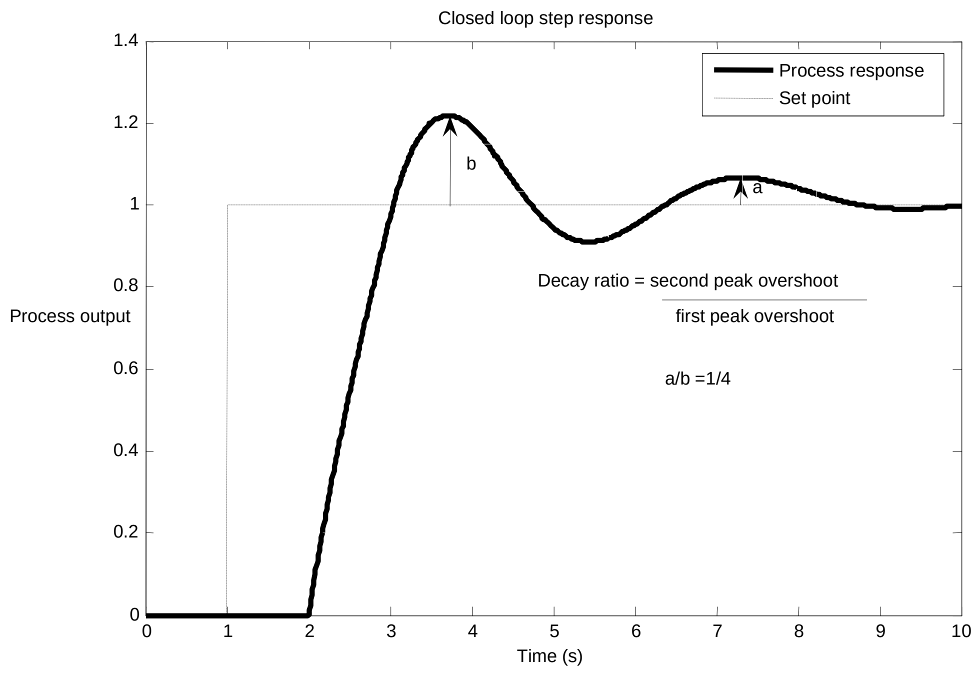


Figure 4.1: Response curve for quarter wave decay ratio

4.2.1 ZN closed-loop tuning method (Ultimate gain and ultimate period method)

The closed-loop tuning method proposed by ZN requires the determination of the ultimate gain and ultimate period. The method can be interpreted as a technique of positioning one point on the Nyquist curve (Åström, 1995). This can be achieved by adjusting the controller gain (K_c) till the system undergoes sustained oscillations (at the ultimate gain or critical gain), whilst maintaining the integral time constant (T_i) at infinity and the derivative time constant (T_d) at zero. Consider Figure 4.2: the closed-loop response is considered stable if there is no encirclement of the point $(-1+j0)$ by the Nyquist plot (Figure 4.2a) of the system (Ogata, 1970). For a proportional gain

(K_c) = 2 the closed-loop response is stable and the Nyquist stability criterion is met (Figure 4.2b). For $K_c = 8$, sustained oscillations are produced since there is an encirclement of the point $(-1 + j0)$ by the Nyquist locus (Figure 4.2c and Figure 4.2d). In both simulations, $T_i = \infty$ and T_d=0 is used with a change only in the proportional gain K_c to move the process closer to the ultimate point.

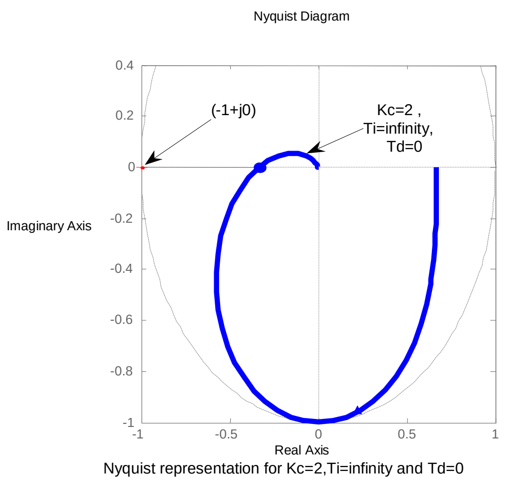


Figure 4.2a

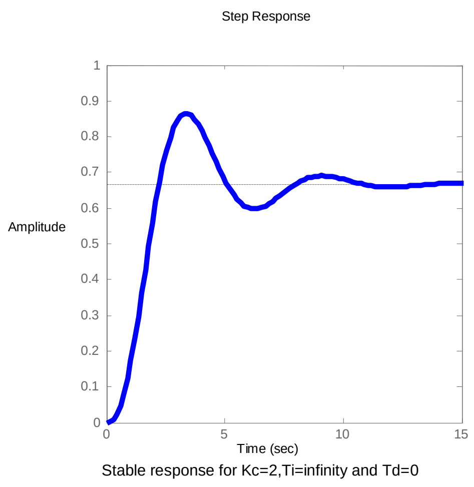


Figure 4.2b

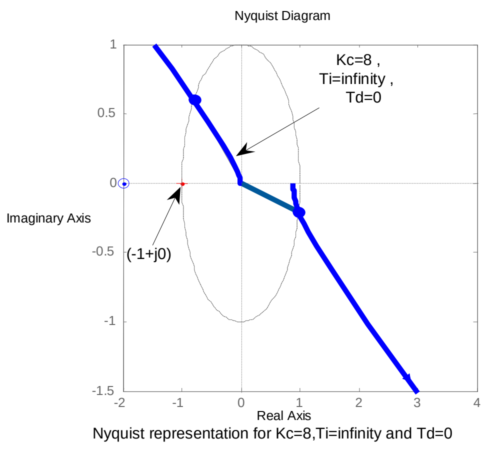


Figure 4.2c

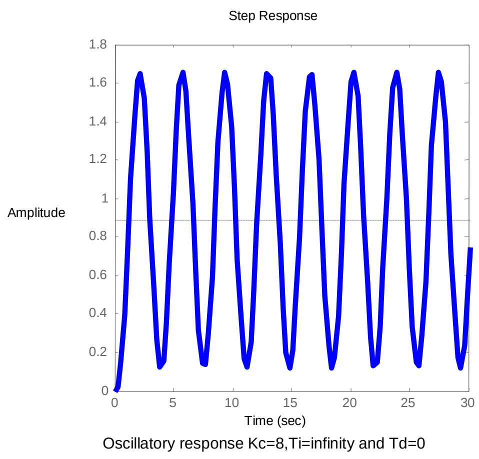


Figure 4.2d

Figure 4.2: Closed-loop step response of $G_p(s) = \frac{1}{(s+1)^3}$ with $K_c = [2,8]$, $T_i = \infty$ and $T_d = 0$

A significant drawback of this closed-loop tuning method is that the ultimate gain has to be determined through trial and error and the system has to be driven to its stability limits. Another disadvantage is that when the process is unknown, the amplitudes of the undampened oscillations can become excessive when using trial and error to determine the ultimate gain of the system. This could lead to unsafe plant conditions. The closed-loop tuning rules for P, PI and PID control are given in Table 4.1.

4.2.2 ZN open-loop tuning method (Process reaction curve method)

This method is based on a registration of the open-loop step response of the system. From Figure 4.3, it can be seen that following a step change, the system's S-shaped response is characterized by three parameters, namely the process static gain K_p, the process time constant T_p and L_p . These parameters are used to determine the controller's tuning parameters (see Table 4.2).

Controller	K_c	T_i	T_d
P	$0.5 K_u$	∞	0
PI	$0.4 K_u$	$0.8 P_u$	0
PID	$0.6 K_u$	$0.5 P_u$	$0.125 P_u$

Table 4.1: Ziegler-Nichols closed-loop tuning parameter (Ziegler and Nichols, 1942)

Controller	K_c	T_i	T_d
P	$\frac{T_p}{L_p K_p}$	∞	0
PI	$0.9\frac{T_p}{L_p K_p}$	$3.33 L_p$	0
PID	$1.2\frac{T_p}{L_p K_p}$	$2L_p$	$0.5L_p$

Table 4.2: Ziegler-Nichols open-loop tuning parameter (Ziegler and Nichols, 1942)

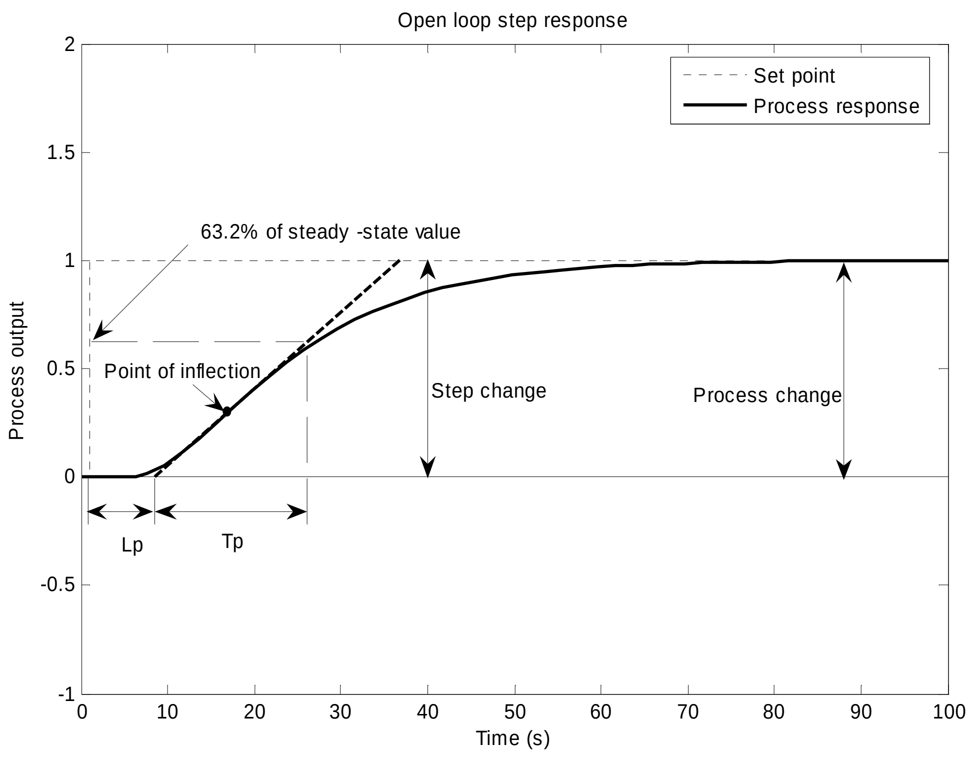


Figure 4.3: Open-loop process reaction curve for a step change

An advantage of the open-loop method is that it is faster and only requires a step change to be applied at the process input in order to determine extract the relevant data for determining the tuning parameters. The method does however suffer from some serious drawbacks namely:

i) The “S-shaped” process reaction curve and its inflection point are difficult to identify when the measurement is noisy and,

ii) A considerable amount of error can be introduced into the tuning calculations if the point of inflection is not determined accurately (Lipták, 1995).

4.2.3 Assessing the efficacy of Ziegler-Nichols tuning rules for dead-time dominant process

ZN tuning yields a poor closed-loop performance for dead-time dominant control loops (Åström, 1995; Shinskey, 1994; Majhi and Atherton, 1999). To illustrate this point, consider the PID controller algorithm of (4.1) being used to control the process defined in (4.2) in a SISO control loop configuration.

$$G_p(s) = K_c[1 + \frac{1}{T_i s} + T_d s] + b \qquad \text{Equation (4.1)}$$

$$G_p(s) = \frac{\exp(-5s)}{(s+1)^3} \qquad \text{Equation (4.2)}$$

For closed-loop tuning, $K_u = 1.25$ and $P_u = 15.7$. The parameters extrapolated from the systems open-loop response to a step input are: $K_p = 1$, $T_p = 3.3$ and $L_p = 5$. Table 4.3 shows the ZN open-loop and closed-loop tuning parameters.

From Figure 4.4, both variants of the ZN tuning methodology methods results in a damped oscillating response. Also, the recovery from a load disturbance is slow since the integral action given by the Ziegler-Nichols method is weak.

ZN tuning method	**K_c**	**T_i**	**T_d**
Closed-loop tuning	0.75	7.9	2
Open-loop tuning	0.8	10	2.5

Table 4.3: Ziegler-Nichols open-loop and closed-loop tuning parameters for
$$G_p(s) = \frac{\exp(-5s)}{(s+1)^3}$$

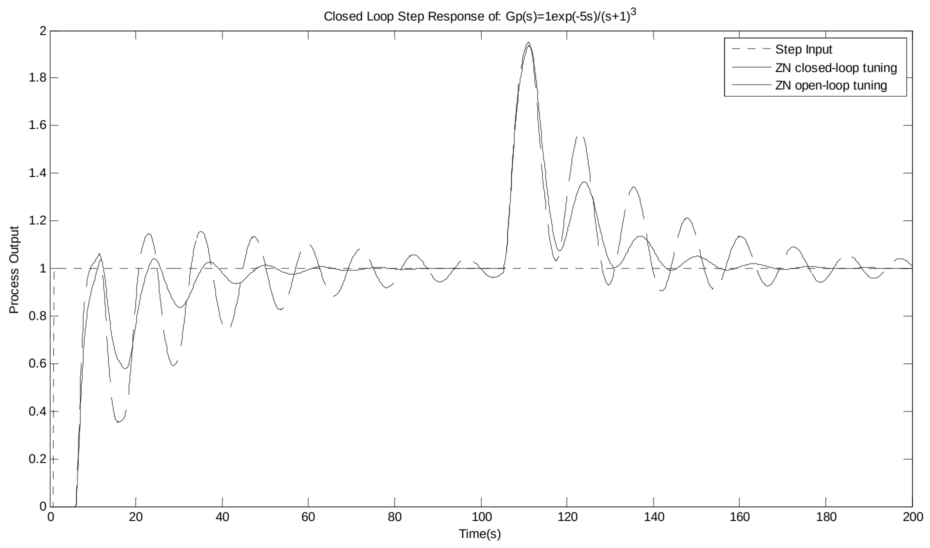


Figure 4.4: Step response using ZN open-loop and closed-loop tuning for a dead-time dominant process $G_p(s) = \dfrac{\exp(-5s)}{(s+1)^3}$

4.3 Cohen-Coon tuning (Open-loop tuning)

The ZN method was designed for a process that cannot regulate itself. To account for self-regulation, Cohen-Coon (CC) introduced the self-regulation index or controllability ratio given by (4.3) (Cohen and Coon, 1953)

$$\varepsilon = \frac{L_p}{T_p} \qquad \text{Equation (4.3)}$$

With regards to (4.3), L_p refers to the process dead time and T_p denotes the process time constant. This method is based on a first-order-plus-dead-time (FOPDT) process models (4.4):

$$G_p(s) = \frac{K_p \exp(-L_p s)}{(T_p s + 1)} \qquad \text{Equation (4.4)}$$

A summary of the CC method is given in Table 4.4.

4.3.1 Comparison between ZN and CC Tuning

A fundamental difference between the ZN and CC methods is as follows: The ZN method associates the integral and derivative constants completely with the process dead-time, whereas the CC method adjusts the integral and derivative time constants according to the particular relationship between the process dead time and the process time constant. For both methods, the controller gain is a function of this relationship. Since processes having different controllability ratios experience different dynamic behaviors, the Cohen-Coon method may perform better than the Ziegler-Nichols method (Lipták, 1995). For example, for dead-time dominant processes i.e. processes having a large controllability ratio, the derivative time constant tends towards zero according to the Cohen-Coon tuning formulae. This is reasonable since the derivative action should not be used when the process contains large process time lag (Åström and Hägglund, 2004, Hägglund, 1992). The method does suffer from the decay ratio being too small. This results in closed-loop systems that are characterized by low damping and high sensitivity (Åström, 1995). Furthermore, the tuning formula tends to produce a very oscillatory set-point change closed-loop response because it was derived to give a quarter wave decay ratio following a load disturbance response (Hang *et al.,* 1991).

Controller	K_c	T_i	T_d
P	$\frac{1}{K_p}\left[0.35+\frac{1}{\varepsilon}\right]$	∞	0
PI	$\frac{1}{K_p}\left[0.083+\frac{0.9}{\varepsilon}\right]$	$\left[\frac{3.3+0.31\varepsilon}{1+2.2\varepsilon}\right]T_p$	0
PID	$\frac{1}{K_p}\left[0.25+\frac{1.35}{\varepsilon}\right]$	$\left[\frac{2.5+0.46\varepsilon}{1+0.61\varepsilon}\right]T_p$	$\left[\frac{3.7}{1+0.19\varepsilon}\right]T_p$

Table 4.4: Cohen Coon tuning formula (Open-loop)

4.4 Åström - Hägglund Gain and Phase Method (Closed-Loop Method)

The tuning method proposed by Åström- Hägglund (1984) is based on the idea of moving the critical point on the process Nyquist curve to a given position. Åström and Hägglund suggested that this point be located at unity gain and at a phase of $(\phi_m - 180°)$ on the Nyquist plot, where ϕ_m denotes the desired phase margin and A_m represents the desired gain margin. The phase and gain margins of a control system are a measure of closeness of the polar plot of the system to the $(-1+j0)$ point. For a system to be stable both the phase and gain margins must be positive. Negative margins indicate instability (Ogata, 1970). For satisfactory performance, the phase margin should be between 30° and 60°, and the gain margin should be greater than 6 dB (Ogata, 1970). The Nyquist plots shown in Figures 4.5a and Figure 4.5b illustrate the phase margin and gain margin of a stable and unstable system, respectively. The phase margin is that amount of additional phase lag at the gain crossover frequency (ω_c) required to bring the system to the verge of instability, where ω_c is defined as the frequency at which $|G(jw)|$ (the magnitude of the open-loop transfer function) is unity. The phase margin (ϕ_m) is 180° plus the phase angle ϕ of the open-loop transfer function at the gain crossover frequency and is defined in (4.5):

$$\phi_m = 180^\circ + \phi \qquad \text{Equation (4.5)}$$

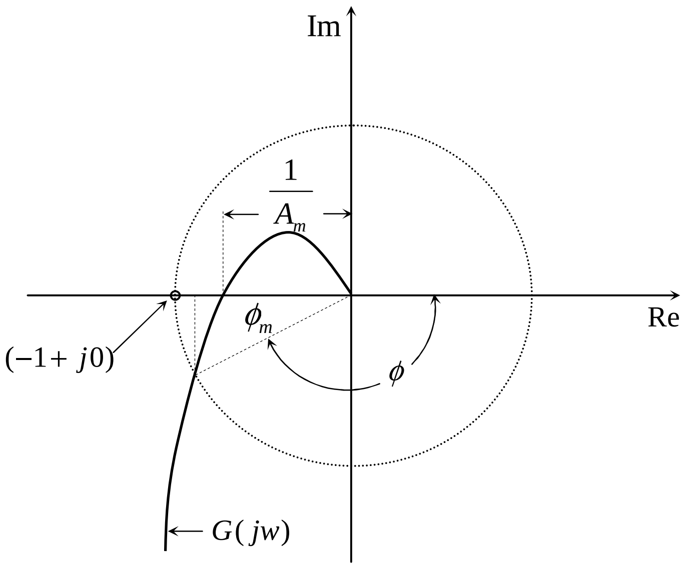


Figure 4.5a: Nyquist plot of stable system showing gain and phase margins

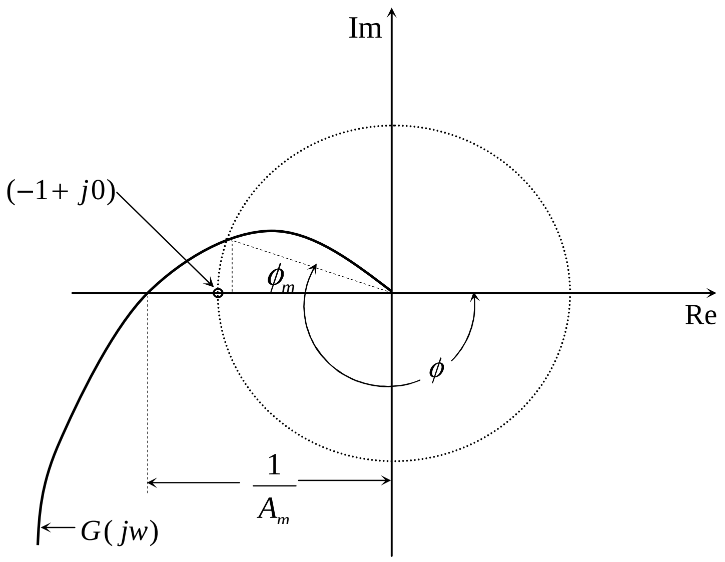


Figure 4.5b: Nyquist plot of unstable system showing gain and phase margins

The phase margin is positive for $\phi_m > 0$ and negative for $\phi_m < 0$. For the system to be stable the phase margin must be positive.

The gain margin is defined as the reciprocal of the magnitude $|G(jw)|$ at the frequency where the phase angle is -180°. Defining the phase crossover frequency (ω_p) to be the frequency at which the phase angle of the open-loop transfer function equals -180° gives:

$$A_m = \frac{1}{|G(j\omega_p)|} \qquad \text{Equation (4.6)}$$

The gain margin is positive if $A_m > 1$ and negative if $A_m < 1$. A positive gain margin indicates that the system is stable and a negative gain margin means that the system is unstable.

A fundamental weakness in the ZN closed-loop method is that the method relies on trial and error adjustments to set the ultimate gain and ultimate period. To overcome this weakness, Åström-Hägglund (1984) proposed their gain and phase method for determining specific points on the Nyquist curve to assist in determining controller pre-tuning parameters. Their approach is based on the use of a simple relay in series with the process (see Figure 4.6). When the switch is in position two, the PID controller is disconnected from the closed-loop and is replaced by the relay. This mode is generally considered a "*pre-tuning*" phase where specific dynamics of the process are determined in the closed-loop.

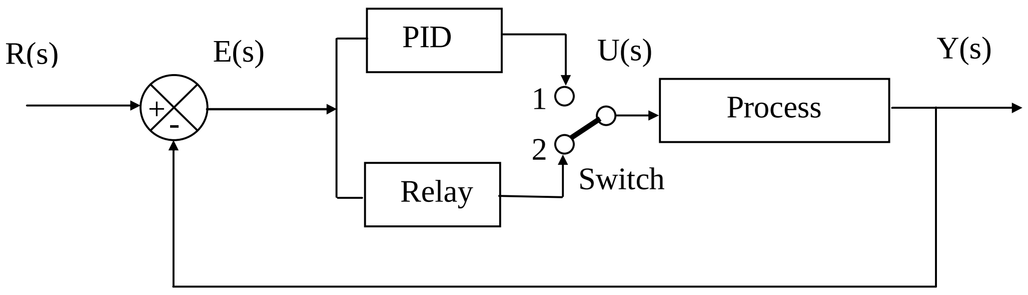


Figure 4.6: Relay feedback system

When the control signal generated by the relay is a square wave, the corresponding process output is similar to a sinusoidal waveform with the process input $U(s)$ and process output $Y(s)$ having opposite phase (Åström, 1995). From Fourier series expansion, the first harmonic of the relay output has amplitude given by $\frac{4d}{\pi}$ where d represents the amplitude of the relay signal and t_c denotes the period of relay switching (Åström and Hägglund, 1984). If the process output is y, then the ultimate gain is thus given as:

$$K_u = \frac{4d}{\pi y} \qquad \text{Equation (4.7)}$$

This result also follows from the describing function approximation for an ideal relay:

$$N(y) = \frac{4d}{\pi y} \qquad \text{Equation (4.8)}$$

For systems designed to perform within gain margin specifications, the proportional gain and derivative time constant is given by Equation (4.9) and Equation (4.10) respectively.

$$K_c = \frac{K_u}{A_m} \qquad \text{Equation (4.9)}$$

$$T_d = \frac{1}{\omega_c^2 T_i} \qquad \text{Equation (4.10)}$$

With reference to equation (4.9) and equation (4.10): A_m is the desired amplitude margin, K_u is the critical gain; the gain crossover frequency (ω_c) is evaluated as:

$$\omega_c = \frac{2\pi}{t_c} \qquad \text{Equation (4.11)}$$

The integration time T_i is arbitrarily chosen (Åström and Hägglund, 1984).

Systems with a prescribed phase margin are obtained by:

$$K_c = K_u \cos\phi_m \qquad \text{Equation (4.12)}$$

$$T_i = \gamma T_d \qquad \text{Equation (4.13)}$$

$$T_d = \frac{\tan\phi_m + \sqrt{\frac{4}{\gamma} + \tan^2\phi_m}}{2\omega_c} \qquad \text{Equation (4.14)}$$

The proportional gain K_c is defined once the ultimate gain K_u is determined. The value $\gamma = 4$ is commonly used to define the relationship between the integral T_i and the derivative T_d time constants (Åström and Hägglund, 1995). The method however, may not be suitable to tune PID controllers for processes with large time delay since this design may result in a very oscillatory closed loop response (Zhuang and Atherton, 1993).

4.5 Poulin-Pomerleau Tuning Method for Second-Order Integrating Process having Dead-Time (SOIPDT) - (Open-Loop Tuning)

Poulin and Pomerleau (1996) proposed a graphical tuning method for integrating processes given by (4.15):

$$G_p(s) = \frac{K_p \exp(-L_p s)}{s(T_p s + 1)} \qquad \text{Equation (4.15)}$$

Their method is based on the Nichol analysis of the open-loop frequency response of the process in series with the controller. With regards to Figure 4.7, the design goal is to position the system on the 3dB ellipse. This is achieved by adjusting the value of the proportional gain (K_c) until the frequency response of the system rests on the stipulated point, which is also known as the maximum peak resonance (M_r). The controller parameters are determined to satisfy the specification on the maximum peak resonance of the closed-loop system.

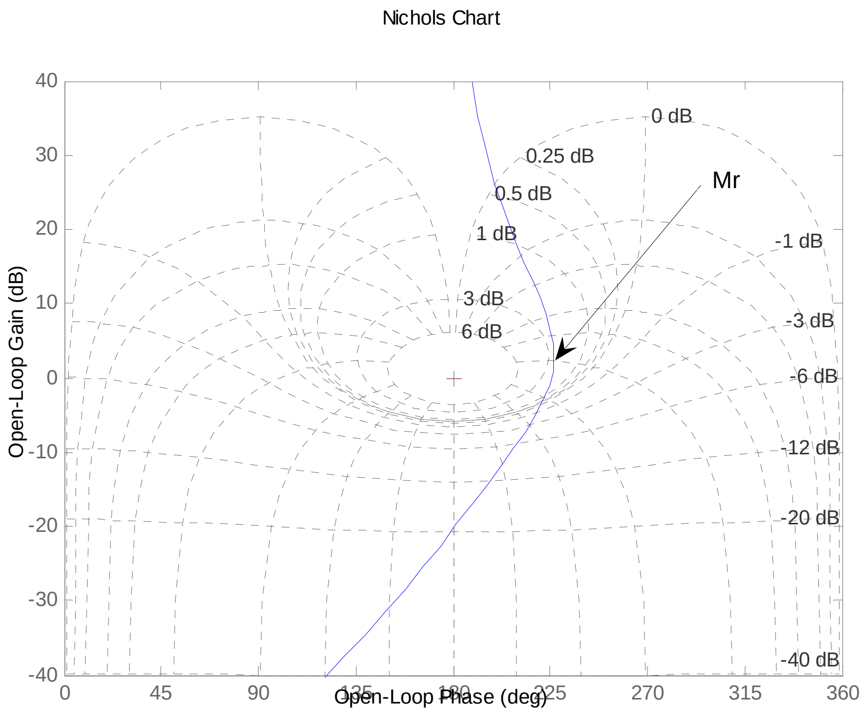


Figure 4.7: Typical open-loop frequency response for second-order integrating process with time delay in cascade with a PI controller

The maximum peak resonance is determined graphically from Figure 4.8. From Figure 4.8, the controller can be tuned for input load disturbance or output load disturbance. In order to determine the optimal M_r, the $\frac{L_p}{T_p}$ ratio must be known. A disadvantage of this tuning method is that the operator requires the maximum peak resonance charts in order to tune the controller. Since this is a graphical technique, it may be difficult for the control practitioner to understand and implement this method (Lee *et al.*, 2000).

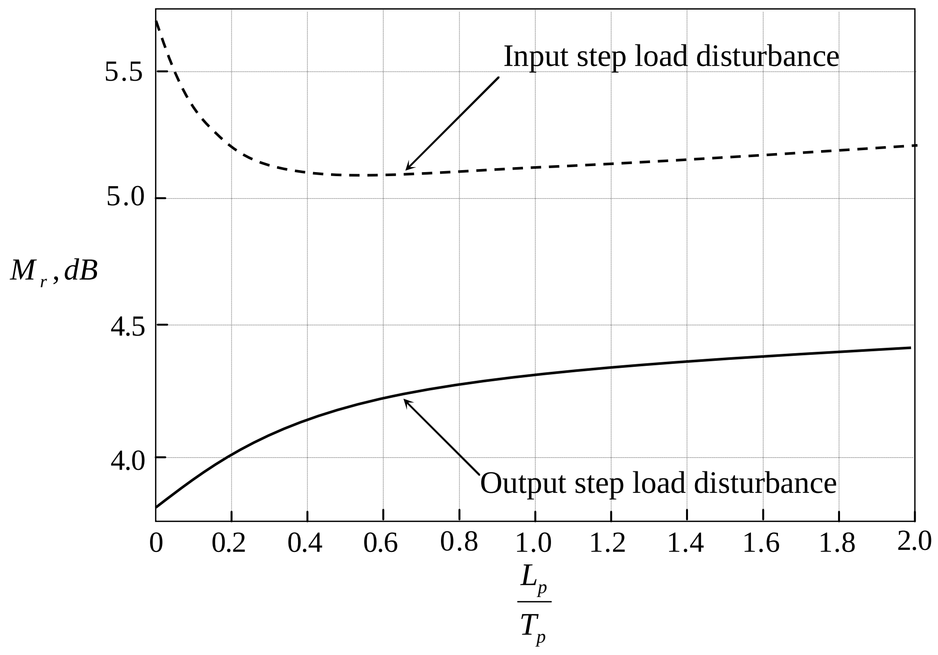


Figure 4.8: Optimal M_r, according to the ITAE criterion for SOIPDT process as a function of the $\frac{L_p}{T_p}$ ratio

From Figure 4.8, M_r is chosen such that the ITAE criterion is minimized for a step load disturbance at the output or the input of the process (Poulin and Pomerleau, 1996). Once the M_r is chosen, the maximum phase value is:

$$\angle G(jw_{\max}) = \arccos\left[\sqrt{10^{0.1M_r} - \frac{1}{10^{0.05M_r}}}\right] - \pi \qquad \text{Equation (4.16)}$$

The frequency $w_{\max}$ at which the phase maximum occurs is:

$$w_{\max} = \sqrt{\frac{1}{T_i(T_p + L_p)}} \qquad \text{Equation (4.17)}$$

The integral time constant that gives the desired $\angle G(jw_{\max})$ is:

$$T_i = \frac{16(T_p + L_p)}{(2\angle G(jw_{\max}) + \pi)^2} \qquad \text{Equation (4.18)}$$

The point $G(jw_{\max})$ is located on the right-most point of the ellipse as specified by M_r. The relationship between $\left|G(jw_{\max})\right|$ and M_r can be visualized on the Nichols chart as shown in Figure 4.7. The maximum gain of the system is given as:

$$\left|G(jw_{\max})\right| = \frac{10^{0.05M_r}}{\sqrt{10^{0.1M_r} - 1}} \qquad \text{Equation (4.19)}$$

The proportional gain that gives the desired $G(jw_{\max})$ is:

$$K_c = \frac{T_i\left|G(jw_{\max})\right|\sqrt{T_p{}^2 w^6_{\max} + w^4_{\max}}}{K_p\sqrt{T_i{}^2 w^2_{\max} + 1}} \qquad \text{Equation (4.20)}$$

The derivative time constant is:

$$T_d = T_p \qquad \text{Equation (4.21)}$$

4.6 De Paor-O’ Malley Tuning for First-Order Open-Loop Unstable Processes having Dead-Time (FODUP)

For De Paor-O’ Malley tuning (1989), controller parameters are derived from a Nyquist analysis of the time delayed process of (4.22) having a single open-loop unstable pole (λ):

$$G_p(s) = \frac{K_p \exp(-L_p s)}{(s-\lambda)} \qquad \text{Equation (4.22)}$$

The Nyquist curve for (4.22) is illustrated in Figure 4.9. Asymptotic stability is obtained if and only if the plot encircles the point $\left(\frac{-1\lambda}{K_c K_p} + j0\right)$ once in an anticlockwise direction (De Paor-O’ Malley, 1989). The design procedures are based upon the classical stability indices of gain and phase margin. The gain margin design does however require a numerical technique for solution of the design problem.

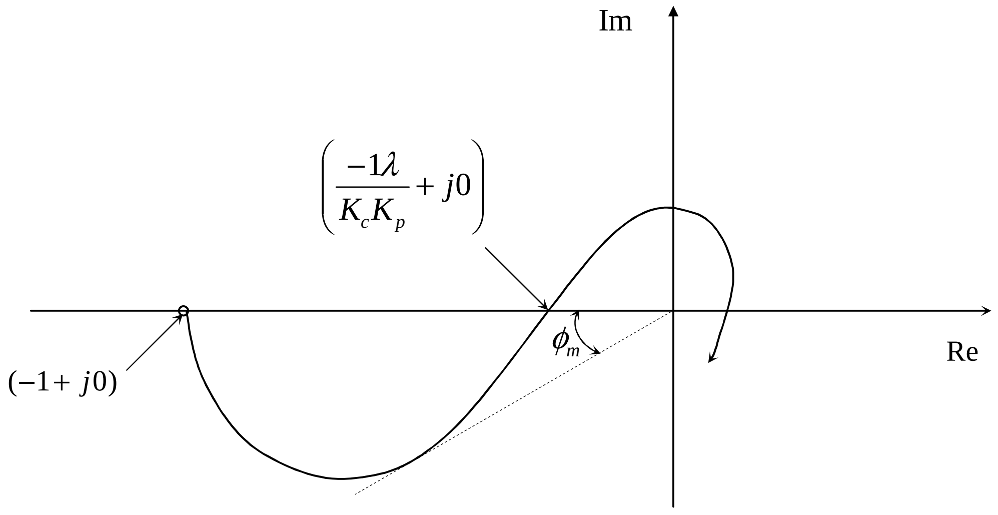


Figure 4.9: Nyquist diagram for open-loop unstable process (Equation 4.22)

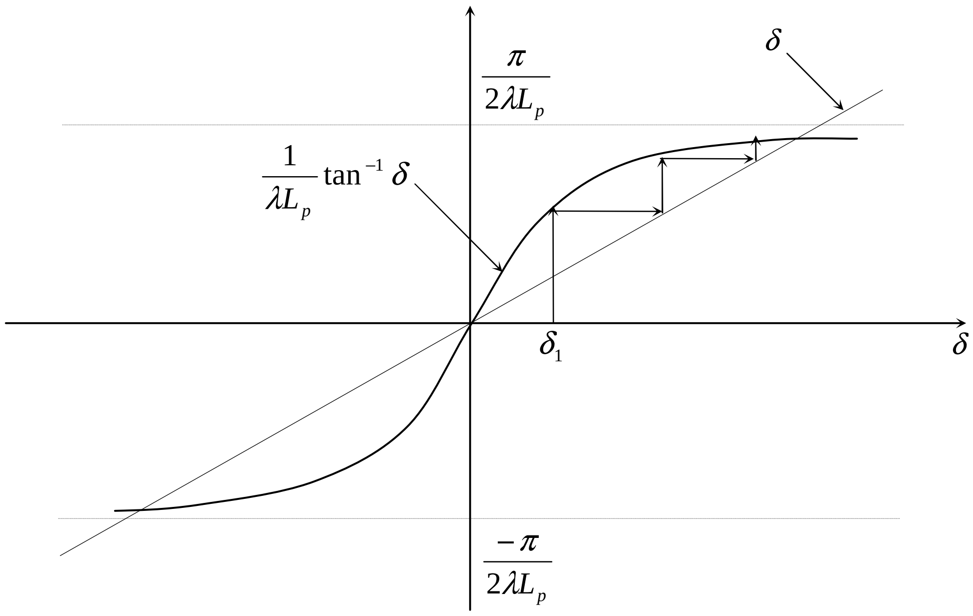
The P-controller parameter to ensure an optimal gain margin is given by (4.23) :

$$K_c = \left[\left(1+\delta_c^2\right)^{1/4}\right]\frac{\lambda}{K_p} \qquad \text{Equation (4.23)}$$

Where, δ_c denotes the smallest positive root and is determined by iterative algorithm (4.24):

$$\delta_{n+1} = \frac{1}{\lambda L_p}\tan^{-1}\delta_n \qquad \text{Equation (4.24)}$$

From Figure 4.10, (4.24) converges to δ_c for any initial guess in the range $0 < \delta_1 < \frac{\pi}{2\lambda L_p}$.



$\frac{\pi}{2\lambda L_p}$
δ
$\frac{1}{\lambda L_p}\tan^{-1}\delta$
δ_1
δ
$\frac{-\pi}{2\lambda L_p}$

Figure 4.10: Iterative algorithm for determination of δ_c

To ensure an optimal phase margin, the tuning parameter for the P-controller is determined from (4.25):

$$K_c = \frac{\lambda}{K_p\sqrt{\lambda L_p}} \qquad \text{Equation (4.25)}$$

and the optimal phase margin is:

$$\phi_m = \tan^{-1}\left(\frac{1-\lambda L_p}{\lambda L_p}\right)^{1/2} - \left(\lambda L_p\left(1-\lambda L_p\right)\right)^{1/2} \qquad \text{Equation (4.26)}$$

For PI control the P-control determined from (4.27):

$$K_c = \cos\left(\left(1-\lambda L_p\right)\lambda L_p\right)^{1/2} + \left(\frac{1-\lambda L_p}{\lambda L_p}\right)^{1/2} \sin\left(\left(1-\lambda L_p\right)\lambda L_p\right)^{1/2} \qquad \text{Equation (4.27)}$$

and the integral term calculated from (4.28):

$$T_i = \left(\left(\left(\frac{1-\lambda L_p}{\lambda L_p}\right)^{1/2} \tan\frac{\phi_m}{2}\right)\lambda\right)^{-1} \qquad \text{Equation (4.28)}$$

Equations (4.27), (4.29) and (4.30) are used to determine the tuning parameters for three term control.

$$T_i = \left(\left(\left(\frac{1-\lambda L_p}{\lambda L_p} \right)^{1/2} \tan\left(\frac{3\phi_m}{4} \right) \right) \lambda \right)^{-1} \qquad \text{Equation (4.29)}$$

$$T_d = \left(\frac{\lambda L_p}{\left(1-\lambda L_p\right)\lambda T_i} \right) \lambda^{-1} \qquad \text{Equation (4.30)}$$

De Paor-O' Malley have derived their stability criterion for processes having control ratios of $\frac{L_p}{T_p} < 1$. The results of the controlled system tuned using the method is highly oscillatory with unacceptable overshoots even for $\frac{L_p}{T_p} = 0.6$ (Venkatashankar and Chidambaram, 1994). Their results on the controller gain (K_c) and the integral time constant (T_i) ratio reflect that for $\frac{L_p}{T_p} > 0.7$, the value for the integral time constant is very large. Hence the integral action in a PI controller would be eliminated making it a simple proportional controller (Venkatashankar and Chidambaram, 1994). For unstable processes the proportional controller would result in an overshoot of 200% making it unsuitable for processes of this nature (Lee *et al.*, 2000).

4.7 Venkatashankar-Chidambaram Tuning Method for First-Order Open-Loop Unstable Processes Having Dead-Time (FODUP)

Venkatashankar and Chidambaram (1994) derived approximate analytical tuning formulae based on the De Paor and O' Malley method (1989). The method was developed using P and PI controllers for FODUP processes of the type represented by (4.31):

$$G_p(s) = \frac{K_p \exp(-L_p s)}{(T_p s - 1)} \qquad \text{Equation (4.31)}$$

The work proposed a method to only tune the P and PI controller and does not support tuning of the PID controller. This is viewed as a significant drawback since unstable processes controlled by a PID controller may provide better closed-loop performance in comparison to PI type (Anandanatarajan *et al.*, 2006). In addition, stability analysis of the PI controller tuned using this method requires that the controllability ratio for the process model be less than 0.775 (Venkatashankar and Chidambaram, 1994).

According to the Venkatashankar and Chidambaram method for PI controllers, the proportional control is bounded within the range $K_{c_{\min}} < K_c < K_{c_{\max}}$ to ensure closed-loop stability for systems with controllability ratio of $\frac{L_p}{T_p} < 0.775$. For systems approaching $\frac{L_p}{T_p} = 0.25$ $K_{c_{\min}}$ is determined from (4.32):

$$K_{c_{\min}} = (0.98)\left[1 + T_p^2 \omega_c^2\right]^{0.5} \qquad \text{Equation (4.32)}$$

where the gain crossover frequency (ω_c) is defined as:

$$\omega_c = \frac{2}{(T_p - L_p)}$$ Equation (4.33)

For systems approaching $\frac{L_p}{T_p} = 0.775$, $K_{c_{\max}}$ is calculated from (4.34)

$$K_{c_{\max}} = \frac{(5\alpha)[1 + T_p^2 \omega_c^2 \alpha^2]^{1/2}}{[1 + 25\alpha^2]^{1/2}}$$ Equation (4.34)

where, α is:

$$\alpha = \frac{5}{L_p}(T_p - L_p)\beta$$ Equation (4.34a)

and β is chosen according to (4.34b) and (4.34c):

$$\beta = 1.373 \quad \text{for} \quad \frac{L_p}{T_p} < 0.25$$ Equation (4.34b)

$$\beta = 0.953 \quad \text{for} \quad 0.25 < \frac{L_p}{T_p} < 0.75$$ Equation (4.34c)

Systems with $\frac{L_p}{T_p} \rightarrow 0.775$ results in $\alpha \rightarrow 1$ (Venkatashankar and Chidambaram, 1994). T_i is represented by:

$$T_i = 25(T_p - L_p)$$ Equation (4.35)

4.8 Summary and conclusion

A summary of the tuning rules discussed in this chapter is given in Table 4.5(a) and Table 4.5(b). The Ziegler-Nichols tuning methods result in closed-loop systems with very poor damping, since it was intended for quarter wave damping. Ziegler-Nichols open-loop tuning is only suitable for open-loop stable processes and the closed-loop method is applicable to processes that operate deep within the stable region under closed-loop conditions. Processes operating on the periphery of the stable region will be unsafe to tune using closed loop Ziegler-Nichols tuning. The method also results in poor tuning (Åström, 1995) and fine tuning is usually necessary to improve loop performance. In spite of these shortcomings, the method is still the most preferred by control practitioners. Its popularity is largely due to the fact that it was amongst the first tuning methods to be proposed, and compared to most other tuning techniques it is still the simplest to use.

The ZN and AH methods are unsuitable for dead-time dominant processes (Åström, 1995; Shinskey, 1994; Zhuang and Atherton, 1993). Other methods that were discussed are not often applied in practice because they are perceived as being complicated and time consuming to implement (Pillay and Govender, 2007). For these reasons the research proposes a simple stochastic methodology, based on the PSO computational algorithm, for determining PID tuning parameters. This is discussed in the next chapter.

Tuning rule	Controller	K_c	T_i	T_d	Comments
ZN (Closed-loop)	P	$0.5K_u$	∞	0	K_u determined by trail and error
ZN (Closed-loop)	PI	$0.4K_u$	$0.8P_u$	0	K_u and P_u determined by trail and error
ZN (Closed-loop)	PID	$0.6K_u$	$0.5P_u$	$0.125P_u$	K_u and P_u determined by trail and error
ZN (Open-loop)	P	$\frac{T_p}{L_p K_p}$	∞	0	K_p, T_p and L_p determined by open-loop step response
ZN (Open-loop)	PI	$0.9\frac{T_p}{L_p K_p}$	$3.33L_p$	0	K_p, T_p and L_p determined by open-loop step response
ZN (Open-loop)	PID	$1.2\frac{T_p}{L_p K_p}$	$2L_p$	$0.5L_p$	K_p, T_p and L_p determined by open-loop step response
CC	P	$\frac{1}{K_p}\left[0.35+\frac{1}{\varepsilon}\right]$	∞	0	K_p and ε determined by open-loop step response
CC	PI	$\frac{1}{K_p}\left[0.083+\frac{0.9}{\varepsilon}\right]$	$\left[\frac{3.3+0.31\varepsilon}{1+2.2\varepsilon}\right]T_p$	0	K_p, T_p and ε determined by open-loop step response
CC	PID	$\frac{1}{K_p}\left[0.25+\frac{1.35}{\varepsilon}\right]$	$\left[\frac{2.5+0.46\varepsilon}{1+0.61\varepsilon}\right]T_p$	$\left[\frac{3.7}{1+0.19\varepsilon}\right]T_p$	K_p, T_p and ε determined by open-loop step response

Table 4.5a: Summary of tuning rules

Tuning	Controller	K_c	T_i	T_d	Comments

rule					
AH	PID	$\frac{K_u}{A_m}$	arbitrary	$\frac{1}{\omega_c{}^2 T_i}$	Specified gain margin A_m
AH	PID	$K_u \cos\phi_m$	γT_d	$T_d = \frac{\tan\phi_m + \sqrt{\frac{4}{\gamma} + \tan^2\phi_m}}{2\omega_c}$	Specified phase margin ϕ_m
PP	PID	$\frac{T_i \lvert G(jw_{\max})\rvert \sqrt{T_p{}^2 w^6_{\max} + w^4_{\max}}}{K_p \sqrt{T_i{}^2 w^2_{\max} + 1}}$	$\frac{16(T_p + L_p)}{(2\angle G(jw_{\max}) + \pi)^2}$	T_p	Optimal M_r deduced from Figure 4.8
DO	P	$K_c = \frac{\lambda}{K_p\sqrt{\lambda L_p}}$	∞	0	Based on optimal phase margin ϕ_m
DO	PI	$K_c{}^{(1)}$	$T_i = \left(\left(\left(\frac{1-\lambda L_p}{\lambda L_p}\right)^{1/2} \tan\frac{\phi_m}{2}\right)\lambda\right)^{-1}$	0	Based on optimal phase margin ϕ_m
DO	PID	$K_c{}^{(1)}$	$T_i = \left(\left(\left(\frac{1-\lambda L_p}{\lambda L_p}\right)^{1/2} \tan\left(\frac{3\phi_m}{4}\right)\right)\lambda\right)^{-1}$	$T_d = \left(\frac{\lambda L_p}{(1-\lambda L_p)\lambda T_i}\right)\lambda^{-1}$	Based on optimal phase margin ϕ_m
VC	PI	$K_c = \frac{(5\alpha)[1 + T_p^2\omega_c^2\alpha^2]^{1/2}}{[1+25\alpha^2]^{1/2}}$	$T_i = 25(T_p - L_p)$	0	$\alpha = \frac{5}{L_p}(T_p - L_p)\beta$ where, $\beta = 1.373$ for $\frac{L_p}{T_p} < 0.25$ and $\beta = 0.953$ for $0.25 < \frac{L_p}{T_p} < 0.75$

$$K_c{}^{(1)} = \cos\left((1-\lambda L_p)\lambda L_p\right)^{1/2} + \left(\frac{1-\lambda L_p}{\lambda L_p}\right)^{1/2} \sin\left((1-\lambda L_p)\lambda L_p\right)^{1/2}$$

Table 4.5b: Summary of tuning rules

Chapter 5

Evolutionary Computation and Swarm Intelligence Paradigms

5.1 Introduction

Evolutionary computation (EC) and Swarm Intelligence (SI) fall within the area of artificial intelligence (AI) (Engelbrecht, 2002). EC is founded upon the principles of biological evolution whilst SI techniques are inspired by swarm behavioral patterns occurring in nature (Kennedy *et al.*, 2001). With the increase of computational power, AI has increasingly been used to solve complex linear and nonlinear control problems.

This chapter provides an introduction to EC and focuses specifically on the PSO algorithm for providing an alternative approach to PID controller tuning. A comparison between the PSO and the GA is also included in the study due to the fact that both SI and GA's are based upon a population of so called 'intelligent agents'.

5.2 Evolutionary Computation

EC techniques are inspired by biological concepts such as population mutation, self-organizing and survival of the fittest. They are regarded as general purpose stochastic search methods that simulate the process of natural selection and evolution in the biological world. There are four major evolutionary techniques namely:

5.2.1 Genetic Programming (GP): GP is used to search for the fittest program to solve a specific problem. Individuals are represented as trees and the focus is on the genetic composition of the individual.

5.2.2 Evolutionary Programming (EP): EP is generally used to optimize real-valued continuous functions. EP uses selection and mutation operators and does not use the crossover operator. The focus is on the observed characteristics of the population. The selection operator is used to determine chromosomes (called parents) for mating in order to generate new chromosomes (called offspring.)

5.2.3 Evolutionary Strategies (ES): ES is used to optimize real-valued continuous functions. ES incorporates selection, crossover and mutation operators. ES optimizes both the population and the optimisation process by evolving the strategy parameters (Omran, 2004).

5.2.4 Genetic Algorithms (GA): The GA is a commonly used evolutionary algorithm and has been selected for comparison with the PSO in this thesis. PSO is similar to the GA in the sense that these two evolutionary heuristics are population-based search methods. The GA and its variants have been popular in academia and the industry mainly because of its intuitiveness, ease of implementation and its ability to solve highly non-linear, mixed integer optimization problems that are typical of complex engineering systems (Hassan *et al.*, 2005). GA's have also been successfully utilized to tune PID controllers (see Krohling and Rey, 2001).

EC techniques involves the following steps:

Step 1: Initialise a population of individuals where each individual represents a potential solution to the problem at hand.

Step 2: Apply a fitness function to evaluate the quality of each solution.

Step 3: A selection process is applied during each iteration to form a new population. The selection process is biased toward the fitter individuals to ensure that they will be part of the new population.

Step 4: Individuals are altered using evolutionary operators. The two most frequently used evolutionary operators are mutation and crossover:

Mutation: Mutation introduces diversity to the population by introducing new genes into the genetic pool. During mutation individual agents undergo small random changes that lead to the generation of new individuals. This assists in reducing the possibility of agents being trapped within local optima.

Crossover (or Recombination): This process is synonymous to mating. During crossover two individual agents combine to produce offspring. The main objective of crossover is to explore new areas within the search space.

Step 5: The above-mentioned steps are repeated until the swarm converges on an optimal or sub-optimal solution.

A brief overview of the above-mentioned evolutionary strategies is given in the following section.

5.3 An Overview of Genetic Algorithms

Genetic Algorithms (GA's) are adaptive heuristic search algorithms that follow the Darwinian principle of "survival of the fittest". They are based on the evolutionary ideas of natural selection and genetic inheritance. GA's involve a population of *individuals*, referred to as *chromosomes*, and each chromosome consists of a string of cells called *genes*.

Chromosomes undergo selection in the presence of variation - inducing operators such as *crossover* and *mutation.* Crossover in GA's occurs with a user specified probability called the "*crossover probability*" and is problem dependant. The *mutation operator* is considered to be a background operator that is mainly used to explore new areas within the search space and to add diversity to the population of chromosomes in order to prevent them from being trapped within a local optimum. Mutation is applied to the offspring chromosomes after crossover is performed.

A *selection operator* selects chromosomes for mating in order to generate offspring. The selection process is usually biased toward fitter chromosomes. A so called *fitness function* is used to evaluate chromosomes and reproductive success varies with fitness. Examples of some well-known selection approaches are given below:

Roulette wheel selection: Parent chromosomes are probabilistically selected, based on their fitness. The more fit the chromosome, the higher the probability that it may be chosen for mating.

Rank selection: Roulette wheel selection suffers from the problem that highly fit individuals may dominate in the selection process. When one or a few chromosomes have a very high fitness level compared to the fitness of other chromosomes, the lesser fit chromosomes will have a very slim chance to be selected for mating. This will increase selection pressure, which will cause diversity to decrease rapidly, resulting in premature convergence. To reduce this problem, rank selection sorts the chromosomes according to their fitness. *Base selection* sorts chromosomes based on their rank order.

Tournament selection: In this more commonly used approach a set of chromosomes are randomly chosen (Goldberg, 1989). The fittest chromosomes from the set are then placed in a mating pool. This process is repeated until the mating pool contains a sufficient number of chromosomes to start the mating process.

Elitism: In this approach the fittest chromosome, or a user-specified number of best chromosomes, is copied into the new population. The remaining chromosomes are then chosen using any selection operator.

5.3.1 Premature convergence of Genetic Algorithms

GA's suffer from premature convergence (or stagnation) which occurs when poorly performing individuals attract the population. This attraction is caused by a poor initialization or through selection of an unsuitable local optimum. Convergence prevents further exploration of the search space and reduces search capabilities (Gaing, 2004).

5.4 Swarm Intelligence

Swarm Intelligence (SI) methods are based around the study of collective behavior in decentralized, self-organized systems. SI systems are typically made up of a population of simple agents interacting locally with one another and with their environment. Although there is no centralized control structure dictating how individual agents should behave, local interactions between such agents often lead to the emergence of a global behaviour.

Two of the most successful SI techniques modeled on the behavior of natural systems are ant colony optimization (ACO) proposed by Dorigo and Gambardella (1997) and particle swarm optimization (PSO) proposed by Kennedy and Eberhart (1995). Our research focuses on the PSO method and a detailed description of this method is provided in this chapter. For the sake of completeness a brief description of the ACO approach has also been included.

5.4.1 Ant Colony Optimization

In ACO artificial ants build solutions by traversing a problem space. Similar to real ants, they deposit artificial pheromone on the workspace in a manner that makes it possible for future ants to build better solutions. In real ant colonies the pheromone is used to find the shortest path to food. Using ACO, finite size colonies of artificial ants communicate with each other via artificial pheromones to find quality solutions to optimization problems. ACO has been applied to a wide range of optimization problems such as the traveling salesman problem, and routing and load balancing in packet switched networks (Dorigo and Gambardella, 1997).

5.4.2 Background to Particle Swarm Optimization

The PSO approach utilizes a population based stochastic optimization algorithm proposed by Eberhart and Kennedy (1995). It was inspired from the computer simulation of the social behaviour of bird flocking by Reynolds (1987). Reynolds used computer graphics to model complicated flocking behaviour of birds. He was mainly interested in simulating the flight patterns of birds for visual computer simulation purposes, observing that the flock *appears* to be under central control. Reynolds proceeded to model his flocks using three simple rules, namely *collision avoidance*, *velocity matching* and *flock centering*.

Using these rules Reynolds showed how the behaviour of each agent inside the flock can be modeled with simple vectors. This characteristic is one of the basic concepts of PSO. Boyd and Recharson (1985) examined the decision making process of human beings and developed the concept of individual learning and culture transmission. According to their examination, people utilize two important kinds of information in decision-making processes, namely:

Their own experience: They have tried the choices and know which state has been better so far, and they know how good it was and

Other people's experiences: They have knowledge of how the other agents around them have performed. In other words, they know which choices their neighbours have found positive so far and how positive the best pattern of choice was. Each agent's decisions is based upon his own experience and other people's experience. This characteristic is another basic concept of PSO.

Eberhart and Kennedy (1995) incorporated these ideas into the development of their PSO method and invented simple velocity and position algorithms that mimic natural swarm behaviour. In PSO, a set of randomly generated agents propagate in the design space towards the optimal solution over a number of iterations. Each agent has a memory of its best position and the swarm's best solution.

PSO is similar to EC techniques in a sense that both approaches are population-based and each individual is evaluated according to a specified fitness function. The major difference is that PSO is influenced by the simulation of social behaviour rather than the survival of the fittest (Shi and Eberhart, 2001). Added to this, each individual benefits from its history and its interactions with its peers. PSO is also easy to implement and the fact that no gradient information is required makes it a good candidate for a wide variety of optimization problems (Kennedy *et al.*, 2001). PSO has been successfully applied to solve a broad range of optimization problems ranging from Artificial Neural Network (ANN) training (Salerno, 1997) to reactive power and voltage control (Fukuyama *et al.*, 2000). The PSO method is also computationally less burdening in comparison to other EC techniques such as GA's (Gaing, 2004). A discussion of the basic PSO algorithm is given in the following section.

5.4.2.1 The basic PSO algorithm

In a PSO system, a swarm of individuals (called *particles* or *intelligent agents*) fly through the search space. Each particle represents a candidate solution to the optimization problem. The position of a particle is influenced by the best position visited by itself (i.e. its own experience) and the position of the best particle in its entire population. The best position obtained is referred to as the *global* best particle. The performance of each particle (i.e. how close the particle is from the global optimum) is measured using a fitness function that varies depending on the optimization problem.

Each particle traverses the XY coordinate within a two-dimensional search space. Its velocity is expressed by vx and vy (the velocity along the X-axis and Y-axis, respectively). Modification of the particles position is realized by the position and velocity information (Kennedy *et al.*, 2001). Each agent knows *its best value* obtained so far in the search ($pbest$) and its XY position. This information is an analogy of the personal experiences of each agent. Individual particles also have knowledge about the *best value achieved by the group* (*gbest*) among *pbest*. Each agent uses information relating to: its *current position* (*x,y*), its *current velocities* (*vx,vy*), *distance between its current position* and its $pbest$ *and the distance between its current position and the groups* $gbest$ to modify its position.

The velocity and position of each agent is modified according (5.1) and (5.2) respectively (Kennedy and Eberhart, 1995):

$$v_i^{k+1} = v_i^k + c_1 rand_1 \times (pbest_i - s_i^k) + c_2 rand_2 \times (gbest - s_i^k) \qquad \text{Equation (5.1)}$$

$$s_i^{k+1} = s_i^k + v_i^{k+1} \qquad \text{Equation (5.2)}$$

With regards to (5.1):

v_i^k	= current velocity of agent *I* at iteration *k*
v_i^{k+1}	= new velocity of agent i at iteration k,
c_1	= adjustable cognitive acceleration constants (self confidence),
c_2	= adjustable social acceleration constant (swarm confidence),
$rand_{1,2}$	= random number between 0 and 1,
s_i^k	= current position of agent i at iteration k,
$pbest_i$	= personal best of agent i,
$gbest$	= global best of the population.

For (5.2):

s_i^{k+1} denotes the position of agent i at the next iteration $k+1$,

Figure 5.1 illustrates the concept of modification of a searching point during the PSO process (Eberhart and Kennedy, 1995). With regards to Figure 5.1, s^k and s^{k+1} denote the current and modified search point, respectively; v^k and v^{k+1} respectively represent the current and modified velocity; v_{pbest} and v_{gbest} represents the velocity based upon *pbest* and *gbest*, respectively. Each agent changes its current position using the integration of vectors as shown in Figure 5.1.

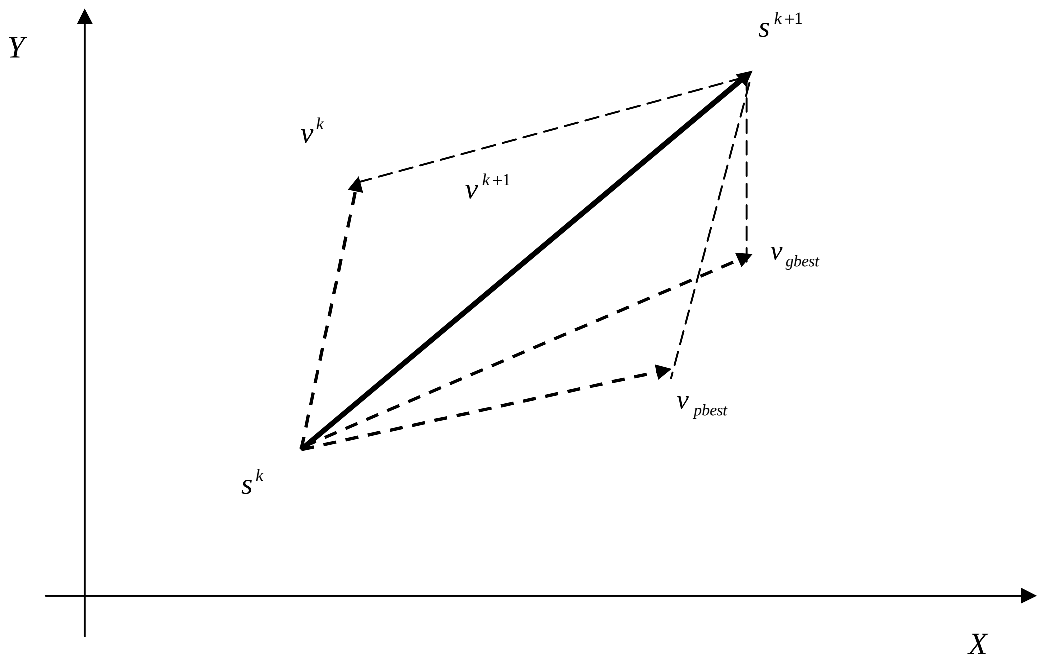


Figure 5.1: Concept of modification of a searching point by PSO (Kennedy and Eberhart, 1995)

A problem with the early version of the PSO algorithm as represented by (5.1) is that the system has a tendency to explode as oscillations become wider (Kennedy *et al.*, 2001). To damp the velocity and limit uncontrollable oscillations of the particles, a method of limiting the velocity to a predetermined value with a maximum velocity parameter ($V_{\max}$) is incorporated into the system (Kennedy *et al.*, 2001). The pseudo-code for limiting particle velocity is as follows (Kennedy *et al.*, 2001):

```
If        $v^{k+1} > V_{max}$     then $v^{k+1} = V_{max}$
Else if   $v^{k+1} > -V_{max}$    then $v^{k+1} = -V_{max}$
```

The effect of this code allows particles to oscillate within bounds with no tendency for the swarm to converge (Kennedy *et al.*, 2001). The $V_{\max}$ parameter thus improves the resolution of the search and arbitrarily limits the velocities of each particle (Carlisle and Dozier, 2001).

5.4.3 Variations to the PSO algorithm

Variations to the conventional PSO algorithm of Eberhart and Kennedy (1995) to control convergence of the swarm have been proposed by Shi and Eberhart (1998) and Clerc (1999). The method proposed by Shi and Eberhart (1998) uses an "inertia weighting" function (see (5.3)) to control the swarm's convergence.

$$w = w_{\max} - \frac{w_{\max} - w_{\min}}{iter_{\max}} \times iter \qquad \text{Equation (5.3)}$$

With regards to (5.3): w = inertia weight, $w_{\max}$ = initial inertia weight, $w_{\min}$ = final inertia weight, $iter_{\max}$ = number of iterations and $iter$ = current iteration.

The inertia weight controls the impact of the previous velocities: a large inertia weight controls the impact of the previous velocity and a small inertia weight favors exploitation (Shi and Eberhart, 1998). Eberhart and Shi (1998) usually implemented the inertia weights ($w_{\max}$ and $w_{\min}$) so that it decreases over time. The effect of the time-decreasing coefficient is to narrow the search to induce a shift from an exploratory to an exploitative mode (Kennedy *et al.*, 2001). The inertia weight is then multiplied by the current velocity component, to give:

$$v_i^{k+1} = wv_i^k + c_1 rand_1 \times (pbest_i - s_i^k) + c_2 rand_2 \times (gbest - s_i^k) \quad \text{Equation (5.4)}$$

All the meanings of the variables in (5.4) are the same as was defined for (5.1). Clerc's method (1999) to control swarm convergence involves a system of "constricted coefficients" applied to various terms of the conventional swarm velocity algorithm (see (5.1)) proposed by Eberhart and Kennedy (1995). This so called *constriction factor* approach proposed by Clerc (1999) controls the swarm convergence so that:

- the swarm does not diverge in a real value region and
- the swarm converges and searches region more efficiently.

The modified velocity update equation (Clerc, 1999) is given in (5.5):

$$v_i^{k+1} = \chi \left[v_i^k + c_1 rand_1 \times (pbest_i - s_i^k) + c_2 rand_2 \times (gbest - s_i^k) \right]$$ Equation (5.5)

With regards to (5.5): χ represents the constriction factor and is defined in (5.6):

$$\chi = \frac{2}{\left| 2 - \varphi - \sqrt{\varphi^2 - 4\varphi} \right|}$$ Equation (5.6)

Where the constant φ is defined in (5.7):

$$\varphi = c_1 + c_2 \ , \ \varphi > 4$$ Equation (5.7)

The meanings of all the other terms in (5.6) and (5.7) are the same as was previously defined for (5.1). The constriction factor results in convergence over time. Unlike the other EC techniques, the constriction factor approach to PSO ensures the convergence of the search procedures based on mathematical theory (Clerc, 1999). The approach ensures that the amplitude of each agent's oscillation decreases as it focuses on a previous best point (Clerc, 1999). Eberhart and Shi (2000) showed empirically that using both the constriction factor and velocity clamping parameter ($V_{\max}$) generally improves both the performance and the convergence rate of the PSO.

5.4.4 Steps in implementing the PSO method.

Figure 5.2 illustrates the general flowchart for the PSO technique. The sequence can be described as follows:

Step 1: Generation of initial conditions of each agent.

Initial searching points (s_i^0) and the velocities (v_i^0) of each agent are usually generated randomly within the allowable range. The current searching point is set to $pbest$ for each agent. The best-evaluated value of $pbest$ is set to $gbest$ and the agent number with the best value is stored.

Step 2: Evaluation of searching point of each agent.

The objective function is calculated for each agent. If the value is better than the current $pbest$ value of the agent, then $pbest$ is replaced by the current value. If the best value of $pbest$ is better than the current $gbest$, the $gbest$ value is replaced by the best value and the agent number with the best value is stored.

Step 3: Modification of each searching point.

The current searching point of each agent is changed, using (5.3), (5.4) and (5.2) for the inertia weight approach. Equations (5.2), (5.5) and (5.6) is used for the constriction factor method.

Step 4: Checking to exit condition.

The terminating criterion is checked to determine whether it has been achieved. If the terminating criterion is not met then the process is repeated from Step 1, otherwise the algorithm is stopped.

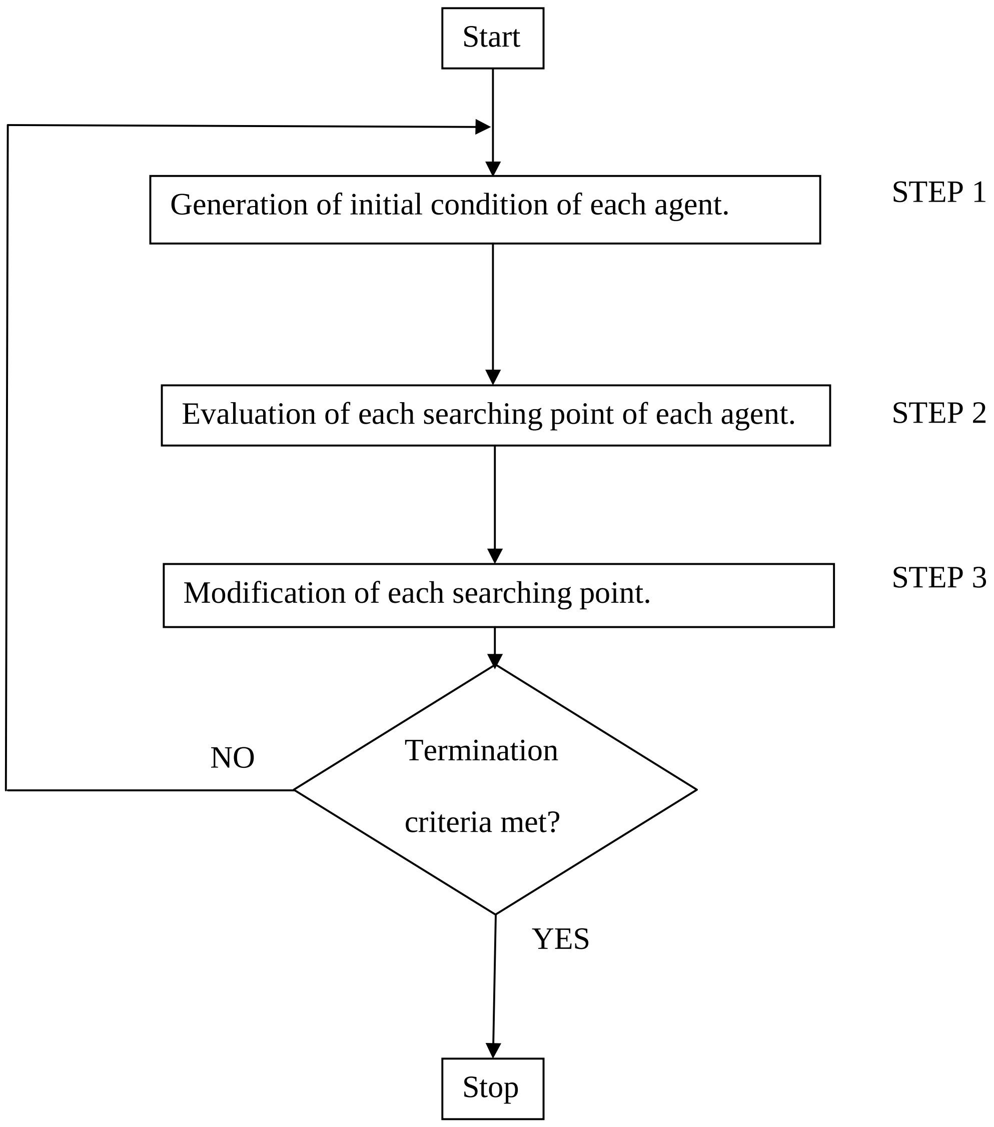

Start
Generation of initial condition of each agent.
STEP 1
Evaluation of each searching point of each agent.
STEP 2
Modification of each searching point.
STEP 3
NO
Termination criteria met?
YES
Stop

Figure 5.2: Steps in PSO (Eberhart and Kennedy, 1995)

5.4.5 Selection of the search method.

The *constriction factor* approach of Clerc (1999) has been chosen over the Eberhart and Kennedy (1995) *inertia weight* technique for this research. This has been done because Clerc’s constriction method has the advantage of being able to recover from a shift to exploratory search back to the exploitative search. Another advantage is that the constriction method converges much faster than the inertia weight approach of Eberhart and Kennedy (1999), and when combined with velocity maximum ($V_{\max}$) may perform well (Kennedy *et al.*, 2001).

5.4.6 Selection of termination method.

A common terminating criterion is to define the maximum amount of iterations that the PSO can perform. Once the PSO reaches the preset maximum iterations the algorithm is automatically terminated according to the flowchart given in Figure 5.2. This terminating criterion has shown to yield poor results since it may produce sub-optimal performance due to premature termination. The *stall-terminating* criterion is therefore utilized in the study. With this approach, if the PSO algorithm stalls continually for any fixed period and the algorithm is then stopped. The fixed period is user defined and can be adjusted to suit a particular application.

5.4.7 Factors affecting PSO performance

The swarm’s size and velocity, plus the behavior of the swarm influence the performance of the PSO process.

i) *Swarm size and Velocity:* The number of particles in the swarm significantly affects the run-time of the algorithm, thus a balance between variety (more particles) and speed (less particles) must be sought. Another important factor in the convergence speed of the algorithm is the maximum velocity parameter ($V_{\max}$). This parameter limits the maximum jump that a particle can make in one step, thus a very large value for this parameter will result in oscillations. On the other hand, a very small value could cause the particle to become trapped within local minima.

ii) *Swarm Behaviour:* The behavior of the swarm is dictated by the summation of the behaviors of individual particles. Each particle 'flies' in the direction of a better solution, weighted by some random factor, maybe overshooting, or potentially finding an individual or global better position. The interaction between the particles in the swarm helps to prevent straying off, whilst keeping close to the optimal solution. This type of behaviour seems ideal when exploring a large search space, especially with a relatively large maximum velocity parameter ($V_{\max}$). Some particles will explore far beyond the current minimum, while the population still remembers the global best.

5.5 Comparison between the GA and PSO

Experiments conducted by Veeramachaneni *et al.* (2003) showed that the PSO method performed better than GA's when applied on some continuous optimization problems. In addition, Eberhart and Shi (1998) compared the PSO to GA's. Their results showed the following, namely:

i) PSO is generally faster, more robust and performs better than GA's especially when the dimension of the problem increases and,

ii) PSO performance is insensitive to the population size (however, the population size should not be too small). Consequently, PSO with smaller swarm sizes perform comparably better than GA's having larger populations.

Although GA's have been widely applied to many control systems, its natural genetic operators would still result in enormous computational efforts. Added to this, the premature convergence of GA's degrades its performance and reduces its search capabilities (Gaing, 2004). On the other hand, the PSO technique can generate a high quality solution within shorter calculation times and more stable convergence characteristics than other stochastic methods (Eberhart and Shi, 1998).

5.6 Summary and conclusion

This chapter presented a brief overview of EC and SI techniques, with special emphasis on the GA and PSO approaches. The GA technique was chosen for comparison because it is similar to the PSO in the sense that they are both population-based computational approaches. Also, both the GA and the PSO approaches depend on information sharing amongst their population members to enhance their search processes using a combination of deterministic and probabilistic rules. The proposed tuning method using the PSO algorithm is discussed in the next chapter.

Chapter 6

PSO Tuned PID Control

6.1 Introduction

This chapter discusses the implementation of the PSO tuning methodology as an optimization strategy to determine the optimal tuning parameters for SISO PID control loops.

6.2 Description of the PSO tuning methodology

Consider Figure 6.1 which represents a 3-dimensional search space being traversed by intelligent agent *"w_1"*. Each dimension's space represents a potential optimal value for K_c, T_i and T_d. The position of a*gent "w_1"* determines the controller's tuning parameters. Modification of an agent's position is realized by responding to velocity and position information according to (5.1) and (5.2). For PI or PD control each agent is given an initial position within a 2-D search space; the same applies to PID control, but within a 3-D search space.

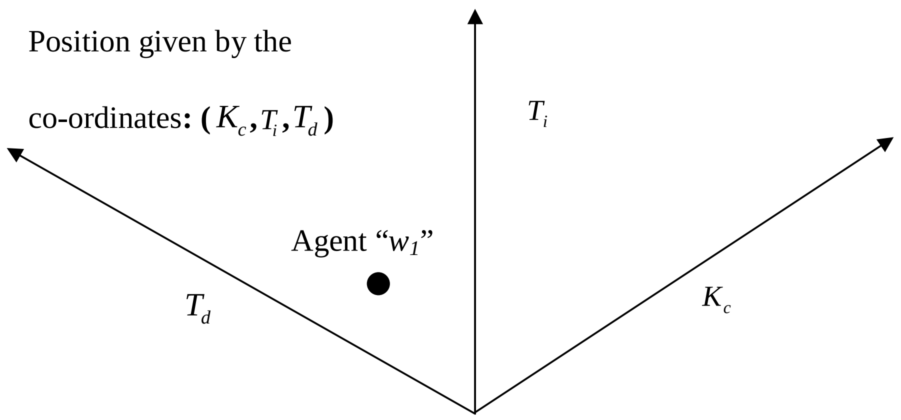


Figure 6.1: Position of swarm agent within a 3-D search space

6.2.1 Application of PSO for PID Tuning

The algorithm proposed by Eberhart and Kennedy (1995) uses a 1-D approach for searching within the solution space. For this study the PSO algorithm will be applied to a 2-D or 3-D solution space in search of optimal tuning parameters for PI, PD and PID control.

Consider position $s_{i,n}$. of the *i*-th particle as it traverses a n-dimensional search space: The previous best position for this *i*-th particle is recorded and represented as $pbest_{i,n}$. The best performing particle among the swarm population is denoted as $gbest_{i,n}$ and the velocity of each particle within the *n*-th dimension is represented as $v_{i,n}$. The new velocity and position for each particle can be calculated from its current velocity and distance with (6.1) and (6.2), respectively:

$$v_{i,n}^{(k+1)} = \chi\left[v_{i,n}^{k} + c_1 rand_1 \times (pbest_{i,n} - s_{i,n}^{k}) + c_2 rand_2 \times (gbest_{i,n} - s_{i,n}^{k})\right] \qquad \text{Equation (6.1)}$$

$$s_{i,n}^{(k+1)} = s_{i,n}^{k} + v_{i,n}^{(k+1)} \qquad \text{Equation (6.2)}$$

With regards to (6.2) and (6.3):

i = number of agents 1,2,….,p;

n = dimension 1,2,3;

$v_{i,n}^{(k+1)}$ = velocity of agent 'i' at iteration $(k+1)$ for n-dimension;

χ = constriction factor;

$v_{i,n}^{k}$ = velocity of agent i at current iteration k for n dimension;

c_1 = cognitive acceleration constants (self confidence);

c_2 = social acceleration constant (swarm confidence);

$rand_{1,2}$ = random number between 0 and 1;

$pbest_{i,n}$ = personal best of agent i for n dimension;

$gbest_{i,n}$ = global best of the population for n dimension;

$s_{i,n}^{k}$ = current position of agent i at iteration k for n dimension;

$s_{i,n}^{(k+1)}$ = position of agent i at iteration $(k+1)$ for n dimension and;

p = number of particles in the population.

For PI, PD and PID control *n = 2, 3* respectively. All other variables have the same meanings as was described in Chapter 5.

6.2.2 Position of the PSO algorithm within the selected control loop

Figure 6.2 illustrates the position of the PSO tuning algorithm within the SISO system used in this study. The steps for PSO tuning were mentioned in Section 5.4.4. The velocity and positional algorithms, namely (6.1) and (6.2), define the search within the solution space. Following each iteration, the impact of each agents position within the search space is evaluated according to the ITAE cost function and the corresponding transient response specifications. The minimization of the ITAE performance index provides a global quantification of overall system performance. The PSO source code used for the tuning is given in Appendix A.

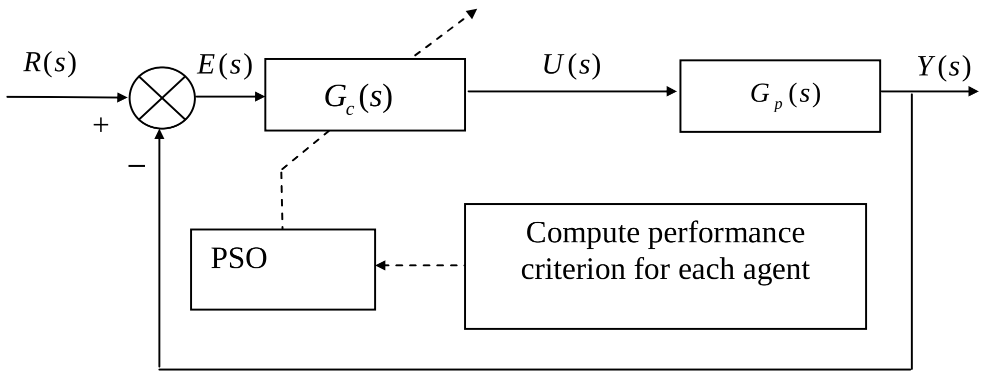


Figure 6.2: Positioning of the PSO optimization algorithm within a SISO system

6.3 Statistical Evaluation of the Dynamical Behaviour of Intelligent Agents

The mean value and standard deviation of the population's position within the search space was calculated in order to evaluate the dynamical behavior and convergence characteristics of the intelligent particles. The mean value (6.3) is used to determine the accuracy of the algorithm, whilst the standard deviation (6.4) measures the convergence speed of the algorithm (Gaing, 2004).

$$mean(\bar{x}) = \frac{\sum_{i=1}^{p} w_i}{p} \qquad \text{Equation (6.3)}$$

$$standard\ deviation\ (\sigma) = \sqrt{\frac{1}{p}\sum_{i=1}^{p}(w_i - \bar{x})^2} \qquad \text{Equation (6.4)}$$

With regards to (6.3) and 6.4):

w_i is the agent's performance index for a particular position, 'p' represents the population size; σ denotes the standard deviation and $\bar{x}$ is the mean value.

6.4 Summary and conclusion

The chapter has indicated the position of the PSO tuning algorithm in a SISO control loop. A particles specific position within the search space represents the tuning parameters of the controller and is evaluated according to the ITAE index. The statistical performance of the tuning approach is measured by determining the corresponding mean and standard deviation of an agent’s position as it traverses the solution space. The next chapter focuses on determining suitable parameters for the PSO algorithm.

Chapter 7

Simulation Study of PSO Performance for Process Control

7.1 Introduction

Swarm particles display distinct behavioral characteristics, namely swarm convergence and particle explosion, as they traverse a system's space searching for an optimal solution. Variations of the swarm's behavior is achieved by adjusting four explicit parameters of the PSO algorithm, namely: *cognitive acceleration (c_1), social acceleration (c_2), maximum velocity (V_{max}),* and *swarm size (S_s)* (Kennedy *et al.*, 2001). These four parameters are set at the beginning of each trial and remain constant throughout.

PSO is a stochastic optimization technique. In view of this, swarm particles start their search for an optimal solution from any region within the system's search space and, the solution reached upon convergence may not necessarily be optimal. For this reason a statistical validation of the PSO's range of solutions is necessary to determine their accuracy. The following indices were used to evaluate the performance of the PSO method over ten trials, and also to provide useful insight into the behavior of the swarm for variations of c_1, c_2, V_{max} and S_s:

Mean value of the ITAE index for the trial ($\bar{x}_{ITAE}$)

Standard deviation of the ITAE index during the trial (σ_{ITAE})

Mean number of iterations utilized by the PSO to perform a search ($\bar{x}_{Iter}$)

Mean time taken by the PSO algorithm to complete a search ($\bar{x}_{Time}$)

7.2 Preliminaries to the evaluation of PSO performance

The performance of the PSO tuning methodology for the selected process models is analyzed by running the algorithm for ten trials. The explicit parameters of the PSO algorithm were adjusted over the following ranges:

$$\text{Swarm Size} = \begin{cases} 2 \\ 5 \\ 10 \\ 20 \\ 40 \\ 50 \end{cases} ;\ V_{max} = 1;\ c_1 = c_2 = 2.05$$

$$\text{Maximum Velocity} = \begin{cases} 0.1 \\ 1 \\ 5 \\ 10 \end{cases} ;\ S_s = 20;\ c_1 = c_2 = 2.05$$

$$\text{Cognitive Acceleration} = \begin{cases} 1 \\ 2.05 \\ 3 \\ 4 \\ 5 \end{cases} ;\ S_s = 20;\ V_{max} = 1;\ c_2 = 2.05$$

$$\text{Social Acceleration} = \begin{cases} 1 \\ 2.05 \\ 3 \\ 4 \\ 5 \end{cases} ;\ S_s = 20;\ V_{max} = 1;\ c_2 = 2.05$$

Table 7.1 shows the heuristically determined PSO parameters that were determined following empirical testing. Each parameter was kept constant whilst the others were

Swarm Parameter	Value
Swarm Size (S_s)	20
Maximum Velocity (V_{max})	1
Cognitive Acceleration (c_1)	2.05
Social Acceleration (c_2)	2.05

Table 7.1: Empirically determined PSO parameters

varied over the previously mentioned ranges. All the tests for the ten trials were conducted under the following conditions: Upper bound of initialization (u_b) = 1; Lower bound of initialization (l_b) = 0. The data extrapolated from the ten trials was used to determine the parameters for the PSO algorithm. The tests were conducted using the control loop shown in Figure 6.2.

7.3 Process models used in the simulation tests

The process models used in the simulation studies are shown in Table 7.2(a) to Table 7.2(d). These models, and variants thereof, are representative of real world models encountered in most process control applications (Åström and Hägglund, 2004; Zhuang and Atherton, 1993).

	Model	**L_p/T_p**
Case 1	$G_p(s) = \frac{\exp(-1s)}{(10s+1)}$	0.1
Case 2	$G_p(s) = \frac{\exp(-1s)}{(s+1)}$	1
Case 3	$G_p(s) = \frac{\exp(-1s)}{(0.1s+1)}$	10

Table 7.2a: FOPDT models

	Model	**L_p/T_p**
Case 4	$G_p(s) = \frac{\exp(-1s)}{(10s+1)^2}$	0.1
Case 5	$G_p(s) = \frac{\exp(-1s)}{(s+1)^2}$	1
Case 6	$G_p(s) = \frac{\exp(-1s)}{(0.1s+1)^2}$	10

Table 7.2b: SOPDT models

	Model	**L_p/T_p**
Case 7	$G_p(s) = \frac{\exp(-0.2s)}{s(0.9s+1)}$	0.06
Case 8	$G_p(s) = \frac{\exp(-0.5s)}{s(0.5s+1)}$	0.5
Case 9	$G_p(s) = \frac{\exp(-0.9s)}{s(0.1s+1)}$	4.5

Table 7.2c: SOIPDT models

	Model	**L_p/T_p**
Case 10	$G_p(s) = \frac{\exp(-1s)}{(10s-1)}$	0.1
Case 11	$G_p(s) = \frac{\exp(-1s)}{(2s-1)}$	0.5
Case 12	$G_p(s) = \frac{\exp(-1s)}{(s-1)}$	1

Table 7.2d: FODUP models

7.4 Results of PSO parameter variation

The results of the simulations for ten trials are illustrated in Figures 7.1 to Figure 7.4 and are referenced in Table 7.3.

PSO parameter being varied	**Case No.**	**Figure Ref. No.**
Swarm Size (S_s): 2;5;10;20;40;50	Case 1, Case 4, Case 7, Case 10	Figure 7.1(a) to Figure 7.1(d)
	Case 2, Case 5, Case 8, Case 11	Figure 7.1(e) to Figure 7.1(h)
	Case 3, Case 6, Case 9, Case 12	Figure 7.1(i) to Figure 7.1(l)
Velocity Maximum ($V_{\max}$): 0.1;1;5;10	Case 1, Case 4, Case 7, Case 10	Figure 7.2(a) to Figure 7.2(d)
	Case 2, Case 5, Case 8, Case 11	Figure 7.2(e) to Figure 7.2(h)
	Case 3, Case 6, Case 9, Case 12	Figure 7.2(i) to Figure 7.2(l)
Cognitive Acceleration (c_1): 1;2.05;3;4;5	Case 1, Case 4, Case 7, Case 10	Figure 7.3(a) to Figure 7.3(d)
	Case 2, Case 5, Case 8, Case 11	Figure 7.3(e) to Figure 7.3(h)
	Case 3, Case 6, Case 9, Case 12	Figure 7.3(i) to Figure 7.3(l)
Social Acceleration (c_2): 1;2.05;3;4;5	Case 1, Case 4, Case 7, Case 10	Figure 7.4(a) to Figure 7.4(d)
	Case 2, Case 5, Case 8, Case 11	Figure 7.4(e) to Figure 7.4(h)
	Case 3, Case 6, Case 9, Case 12	Figure 7.4(i) to Figure 7.4(l)

Table 7.3: Figure references to show the effects of varying S_s, V_{max}, c_1 and c_2 parameters for the selected processes

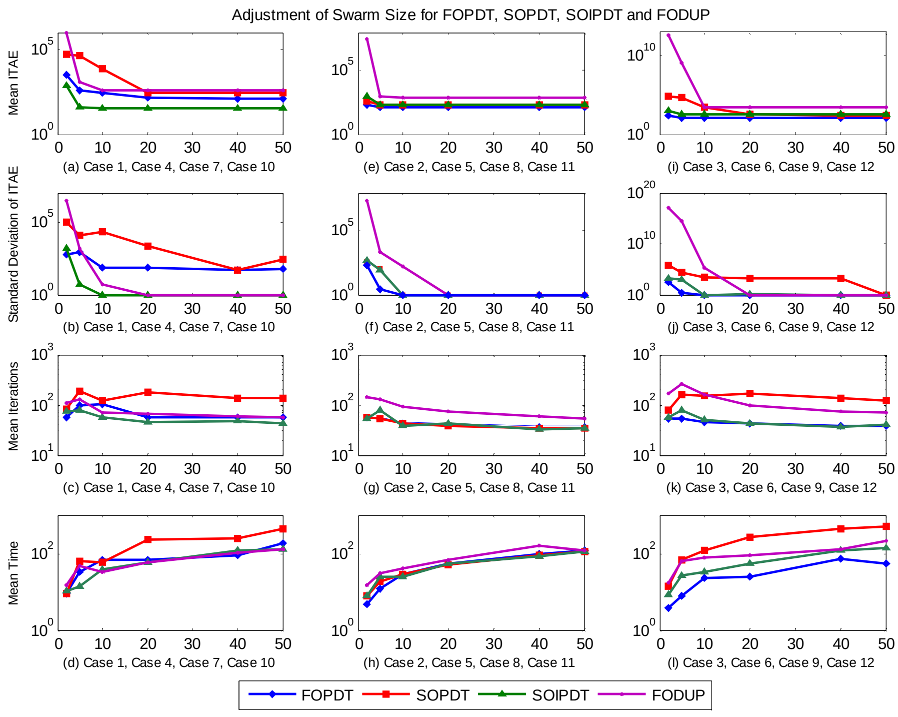


Figure 7.1: Adjustment of Swarm Size (2; 5; 10; 20; 40; 50)

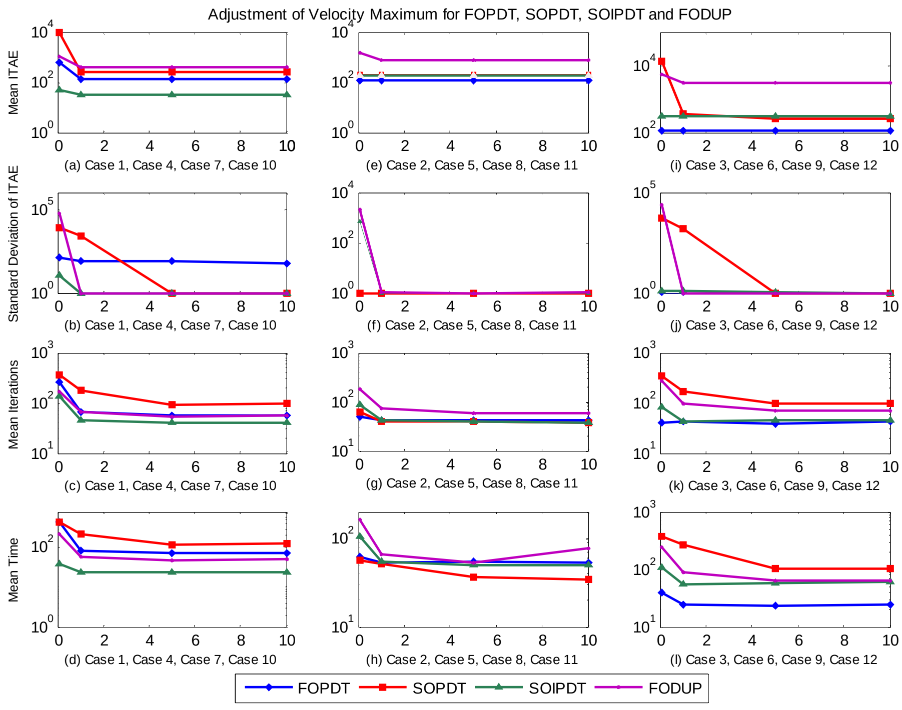


Figure 7.2: Adjustment of Velocity Maximum (0.1; 1; 5; 10)

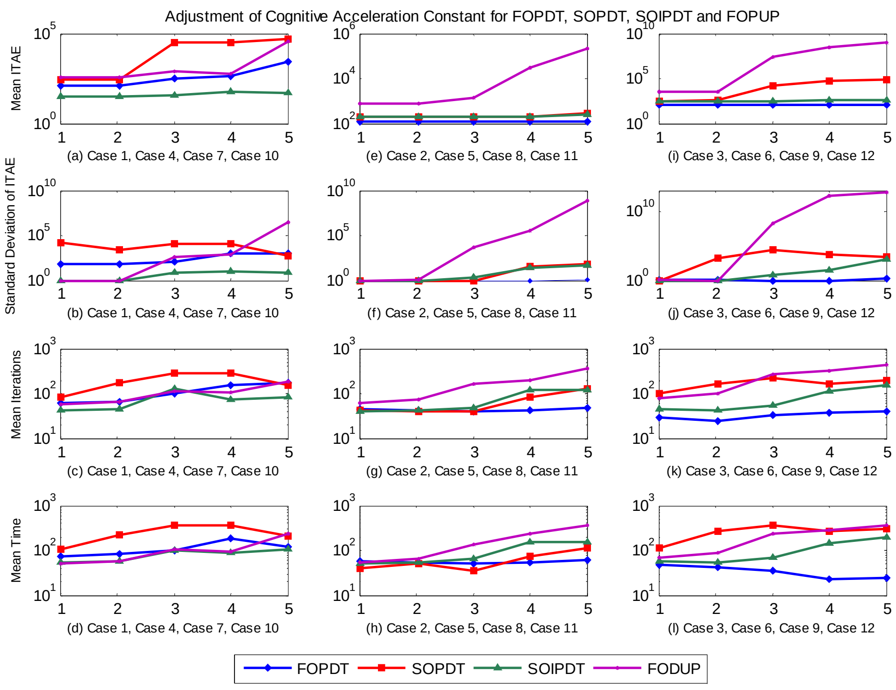


Figure 7.3: Adjustment of Cognitive Acceleration (1; 2.05; 3; 4; 5)

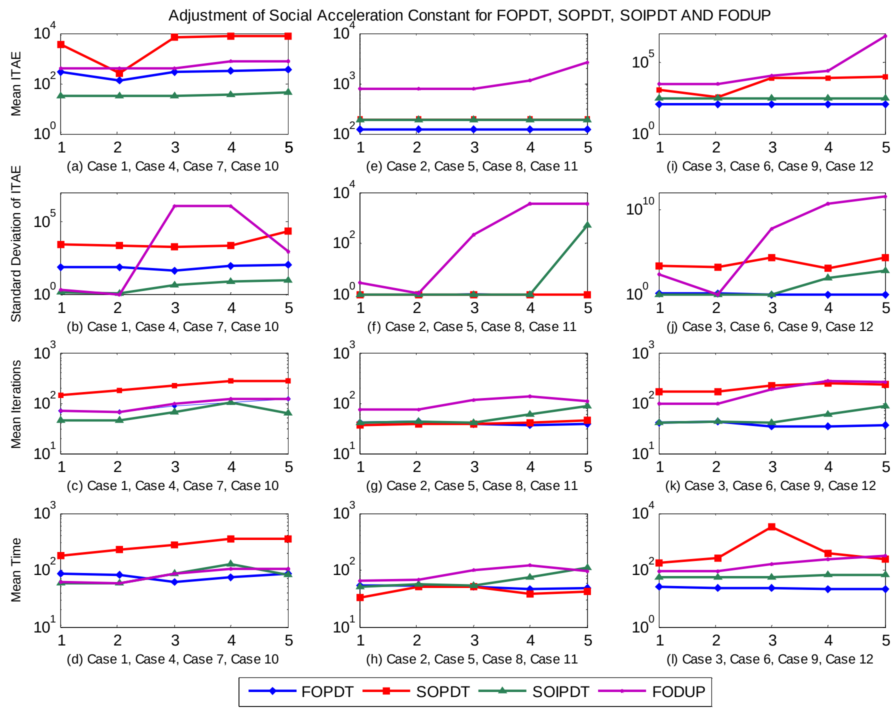


Figure 7.4: Adjustment of Social Acceleration (1; 2.05; 3; 4; 5)

7.5. Observing the effects of varying PSO parameters

7.5.1 Variation in Swarm Size (See Figure 7.1)

Swarm size > 20 agents:

A marginal improvement was noted in the repeatability of the results but this was achieved at the cost of an increased computational burden. The number of iterations necessary before the large swarm convergences decreases again at the cost of higher computational burden. This is due to the fact that more agents find the optimal solution over a lesser number of iterations, albeit at the cost of additional computational time.

Swarm size = 20 agents:

A swarm size of twenty particles works well in all cases whereas a low population size (say two agents) produced poor results. This corresponds to the results of Eberhart and Kennedy (2001). Based upon these observations it was decided that a swarm size of twenty agents was ideal for yielding high quality solutions whilst consuming only minimal computational power.

7.5.2 Variation of Velocity Maximum (See Figure 7.2)

The PSO algorithm yields satisfactory results when $V_{max} = 1$. Low values of V_{max} (say $V_{max} = 0.1$) yielded an unacceptable performance during most of the tests. This can be attributed to the constraints placed on the particle’s motion trajectory, reducing its chances of moving into an ‘optimal solution’ region. Varying the maximum velocity from 1 to 10 resulted in no improvement in the PSO’s search since the constriction factor comes into effect before the V_{max} parameter has an opportunity to limit the particle's velocity (Clerc, 1999).

7.5.3 Variation of Social and Cognitive Acceleration Constants (See Figure 7.3 and Figure 7.4)

The value for cognitive and social acceleration constants has a significant impact on the dynamic performance of the PSO algorithm. This is compounded by the use of Clerc's constriction factor approach (1999) (see (5.4.3)), which inhibits particles oscillations as it focuses on a best point within the solution space. Observations of swarm behavior revealed that a constriction factor of $\chi = 0.73$ with $c_1 + c_2 = 4.1$ produced the best results. Conversely, large magnitudes of c_1 and c_2 results in $\chi < 0.73$ and leads to a pronounced damping effect on particle movement. The unacceptable convergences of the algorithm are illustrated in the results when using high values for cognitive and social acceleration constants. For all cases, the value of $c_1 = 2.05$ and $c_2 = 2.05$ produced the best results.

7.6 Summary and conclusion

Experiments to analyze the effects of variations in PSO parameters for different process models have been described. Observations of the test results show that the optimal performance of the algorithm is limited to only certain values of the PSO parameters. These values work well for FOPDT, SOPDT, SOIPDT and FODUP process models over a range of controllability ratios. A comparison of the proposed method with the other tuning techniques that were discussed in the previous chapters follows next.

Chapter 8

Simulation Studies to Compare the Performance of PSO vs. Other Tuning Techniques

8.1 Introduction

This chapter discusses the simulation experiments that were conducted to compare the control performance of a conventionally tuned PID controller to that of one tuned using the PSO method. The comparison is based upon the loop’s transient response characteristics and the ITAE performance index. The tuning techniques considered in this study includes the Ziegler-Nichols (1942), Cohen-Coon (1953), Åström-Hägglund (1984), De Paor-O’Malley (1989), Venkatashankar-Chidambaram (1994) and Poulin-Pomerleau (1996) tuning methods. The PSO tuned control loop will also be compared to that of loops tuned with the GA algorithm for the following reasons, namely:

i) The GA and the PSO are regarded as soft-computing techniques having strong roots in evolutionary computing,

ii) GA’s and the PSO display stochastic behavioral characteristics,

iii) Both methods are population based search techniques with the ability to handle arbitrary non-linear cost functions and,

iv) PSO and GA’s do not require gradient information of the objective function being optimized.

8.2 Preliminaries to the experiments

A SISO negative unity feedback control system, with the controller and process connected in cascade in the forward path of the control loop will be considered for all the experiments in this study. The control performance of each tuning method will be tested using the process models mentioned in Chapter 3. These process models are representative of typical processes found in most process control applications (Åström and Hägglund, 2004; Zhuang and Atherton, 1993).

The SISO control loop used for all the experiments described in this chapter is given in Figure 8.1. With regards to Figure 8.1: *r(t)* denotes the step input signal, disturbance input signal *d(t)* acts as an additive with the controller output signal *u(t)* such that the process input is governed by $u_{proc}(t) = u(t) + d(t)$. The process output *y(t)* is fed back to the input of the controller to form the error signal *e(t)*. The comparison between the PSO tuning methodology and other selected tuning methods is discussed in Experiments 8.1 to Experiment 8.8.

For the PSO and the GA, the conditions under which the experiments were conducted are the same as for the simulation experiments discussed in Chapter 7, namely: Upper bound of initialization (u_b) = 1, Lower bound of initialization (l_b) = 0; termination is reached after 10 successive stalls of the algorithm. The PSO parameters used in the experiments are the same as those previously mentioned in Table 7.1 and these are repeated in Table 8.1 for convenience. The parameters used for all the experiments using the GA is given in

Table 8.2. The tuning methodology used for the GA technique is similar in nature to the proposed PSO method.

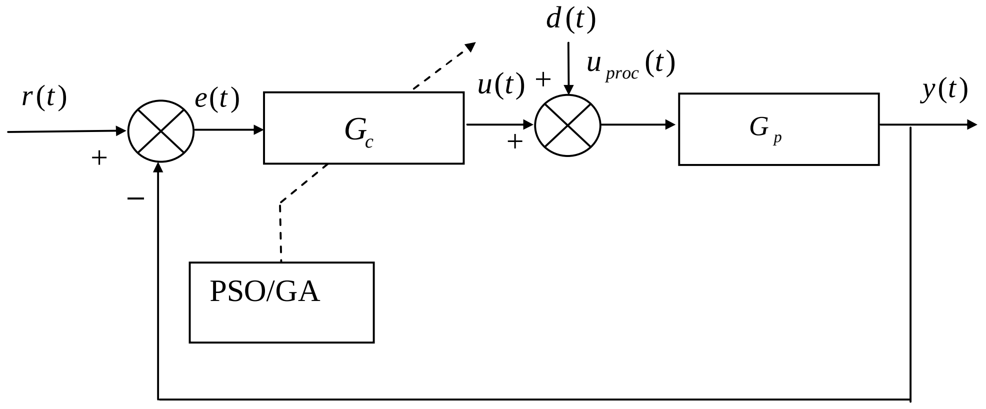


Figure 8.1: Process control loop used in the experiments

Parameter	**Value**
Swarm Size (S_s)	20
Maximum Velocity (V_{max})	1
Cognitive Acceleration (c_1)	2.05
Social Acceleration (c_2)	2.05

Table 8.1: PSO parameters

The GA parameters in Table 8.2 were kept constant for all the simulation experiments and follow standard implementations from the literature (Krohling and Rey, 2001). As a consequence of their stochastic nature, the PSO and the GA yields different controller parameter solutions for each trial. For this reason both optimization methods were each run for ten trials and the average values of these ten trials was then used as the controller tuning parameters. The details of the trial runs conducted for each experiment discussed in this chapter is given in Appendix B1. All the simulation experiments were conducted using a standard Pentium 4 personal computer having 1 giga-byte of random access memory.

Parameter	**Value/Type**
Population Size	20
Selection Method	Tournament
Crossover Method	Heuristic
Crossover Probability	0.35
Mutation Probability	0.02

Table 8.2: GA parameter settings

8.3 Experiments

This section discusses the simulation experiments that were conducted to compare the performance of the PSO tuning methodology to the selected tuning techniques discussed in Chapter 4.

8.3.1 Experiment 8.1: Tuning of FOPDT process for optimal setpoint tracking

8.3.2 Objective

The objective of this experiment is to compare the control performance of a PSO tuned loop to a loop tuned using the GA, ZN and CC tuning methods. The ZN and CC have been included in this experiment because they were originally proposed for FOPDT processes (Ziegler and Nichols, 1942; Cohen and Coon, 1953).

8.3.3 Methodology

The FOPDT process used in this experiment is given in (8.1):

$$G_p(s) = \frac{\exp\,(-0.2s)}{(s+1)} \qquad \text{Equation (8.1)}$$

The PID controller is tuned for setpoint tracking.

8.3.4 Observations and analysis of results

The PID tuning parameters and the dynamic closed-loop performance specifications are shown in Table 8.3. Figure 8.2 shows the closed-loop responses for Experiment 8.1. With regards to Figure 8.2, the ZN and CC methods deliver marginally quicker rise time, but at the expense of larger overshoot and longer settling times. The CC method positions dominant poles that yield a quarter-wave amplitude decay ratio, resulting in oscillation and an increased settling time. The GA tuned system delivers a sluggish response which is evident from the long rise time. Overall the PSO tuned controller delivers an improved response when compared to the other methods.

Tuning Method	**PID Parameters**			**Dynamic Performance Specifications**			**Performance Index**
	K_c	T_i	T_d	t_r	t_s (2%)	M_p (%)	*ITAE*
ZN	6	0.4	0.1	0.1	5.4	78.4	17.6
CC	7.02	0.46	0.07	0.1	8.4	93.1	35.3
GA	0.94	0.67	0.11	1.3	5.4	5.1	17.5
PSO	3.63	0.97	0.07	0.2	1.9	0.1	3.8

Table 8.3: PID parameters and closed-loop response specifications for Experiment 8.1 $\left(G_p(s) = \frac{\exp(-0.2s)}{(s+1)} \right)$

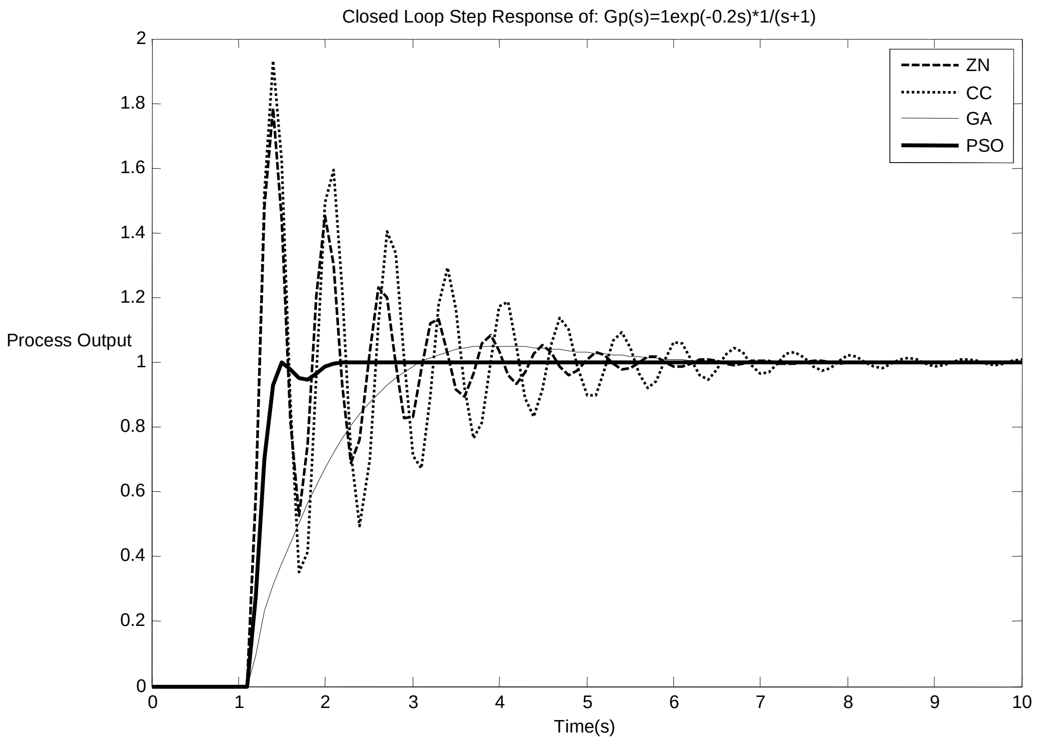


Figure 8.2: FOPDT system response for Experiment 8.1 $\left(G_p(s) = \dfrac{\exp(-0.2s)}{(s+1)} \right)$

Figure 8.3 shows the recorded results of the 10 trial runs for the PSO and GA methods for Experiment 8.1. It is evident from Figure 8.3 that the GA method provides a large level of variance of the PID parameters when compared to the PSO trials. Further analysis of the trial runs are provided in Table 8.4 and show that the PSO technique has better search capability than the GA method, and can also reach an optimal solution within a shorter time and fewer iterations.

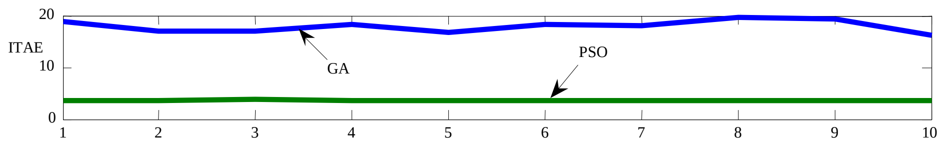


Figure 8.3(a): Performance quantification for 10 trials: PSO vs. GA

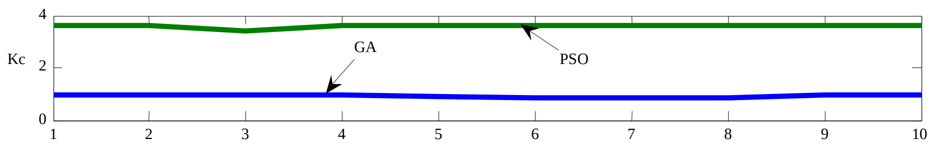


Figure 8.3(b): Search for optimal Kc over 10 trials: PSO vs. GA

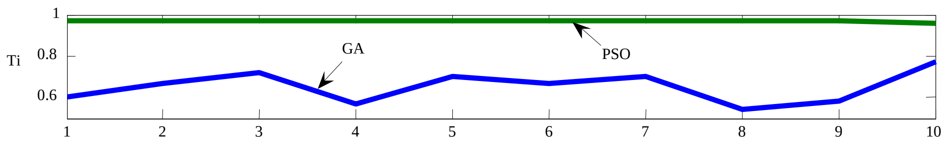


Figure 8.3(c): Search for optimal Ti over 10 trials: PSO vs. GA

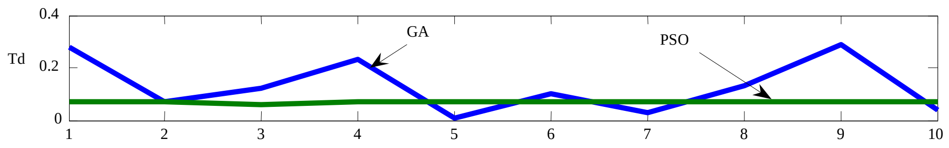


Figure 8.3(d): Search for optimal Td over 10 trials: PSO vs. GA

Figure 8.3: PSO *vs.* GA - Exp. 8.1 results following 10 trials

$$\left(G_p(s) = \frac{\exp(-0.2s)}{(s+1)} \right)$$

	GA						PSO					
	$ITAE$	K_c	T_i	T_d	Time(s)	Iter	$ITAE$	K_c	T_i	T_d	Time(s)	Iter
Mean($\overline{x}$) (After 10 trial runs)	18.15	0.94	0.67	0.11	46.65	80	3.66	3.63	0.97	0.07	20.19	33
Standard Deviation(σ) (After 10 trials)	1.19	0.05	0.08	0.1	15.26	26.8	0.02	0.07	0	0	2.07	3.98

Table 8.4: PSO *vs.* GA – Exp. 8.1 statistical analysis following 10 trials

$$\left(G_p(s) = \frac{\exp(-0.2s)}{(s+1)} \right)$$

On average the PSO required 20.19 seconds to perform a search whilst the GA took 46.65 seconds to reach either a near optimal or an optimal solution. Furthermore, the PSO yields a higher quality solution with low variation on the ITAE and PID parameters for each trial run. The PSO algorithm required only 33 iterations to complete the task of finding the solution to the problem, while the GA method needed 80 iterations. Analysis of the standard deviation of the results indicates a higher variation in the PID parameters for the GA technique than the PSO method.

8.4 Experiment 8.2: Tuning of SOPDT process for optimal setpoint tracking

8.4.1 Objective

In this experiment the ZN closed-loop tuning and the Åström and Hägglund (AH) phase margin method will be compared to the PSO methodology for a SOPDT process model. The AH method has been chosen since it is suited for systems having small $\frac{L_p}{T_p}$ ratios. (Åström and Hägglund, 1984) The ZN method is also included in this experiment because of its popularity amongst control practitioners.

8.4.2 Methodology for experiment

The SOPDT model used in the experiment is given by (8.2):

$$G_p(s) = \frac{\exp(-0.5s)}{s^2 + 2s + 1} \qquad \text{Equation (8.2)}$$

The ultimate gain K_u and ultimate period P_u of the SOPDT process are determined through trial and error. With regards to the Åström and Hägglund (AH) tuning method, a phase margin ϕ_m of 45° is used as the design criterion. In addition, a relationship of $\frac{T_i}{T_d} = 4$ is selected since it is a common choice amongst control practitioners (Åström and Hägglund, 1995). The PSO and GA methods are run for 10 trials each. For the process model given by Equation (8.2), the ultimate gain and period is $K_u = 4.7$ and $P_u = 3.3$, respectively.

8.4.3 Observations and analysis of results

The PID tuning parameters and dynamic closed-loop performance specifications are shown in Table 8.5; the closed-loop responses are given in Figure 8.4. The ZN and AH methods delivers a response having large overshoot and marginally quicker rise time than the PSO method.

The AH method also yields an oscillatory response with long settling time. The GA method gives a system with very long rise time and settling time as compared to the other tuning methods. From Table 8.5, we observe that the PSO method yields a system having minimal overshoot and a rapid settling time. The marginally longer rise time is offset by the system's improved performance index over the other methods.

Figure 8.5 and 8.6 illustrate the performance and dynamic characteristics of the PSO and GA methods for the first trial. With regards to Figure 8.5, the GA method demonstrates that it is susceptible to the problem of local minima for problems of this nature. The GA algorithm is trapped within the minima just after initialization. The PSO method on the other hand displays its capability of being resilient and robust in finding near optimal solutions very quickly and efficiently after its commencement. The optimal solution is reached within 32 iterations taking only 29.84 seconds to complete.

Figure 8.6 illustrates the mean and standard deviation of the 20 individuals for each iteration of the PSO and GA. The evaluation value of each individual is compared to the other members of the populations using mean and standard deviation statistical assessments following each iteration. As expected, the mean and standard deviation value of the entire population is initially large for both techniques during the start since they are randomly initialized between the upper and lower limits of the search space.

Tuning Method	**PID Parameters**			**Dynamic Performance Specifications**			**Performance Index**
	K_c	T_i	T_d	t_r	$t_s\,(2\%)$	$M_p\,(\%)$	$ITAE$
ZN	2.82	1.65	0.41	0.7	5.4	34.6	30.2
AH	3.13	2.5	0.63	0.4	7.1	33.6	33.8
GA	0.87	0.95	0.91	2.1	13.2	16.9	84.4
PSO	2.07	1.99	0.54	0.8	4.6	4.9	17.8

Table 8.5: PID parameters and closed-loop response specifications for Experiment 8.2 $\left(G_p(s) = \dfrac{\exp(-0.5s)}{s^2 + 2s + 1}\right)$

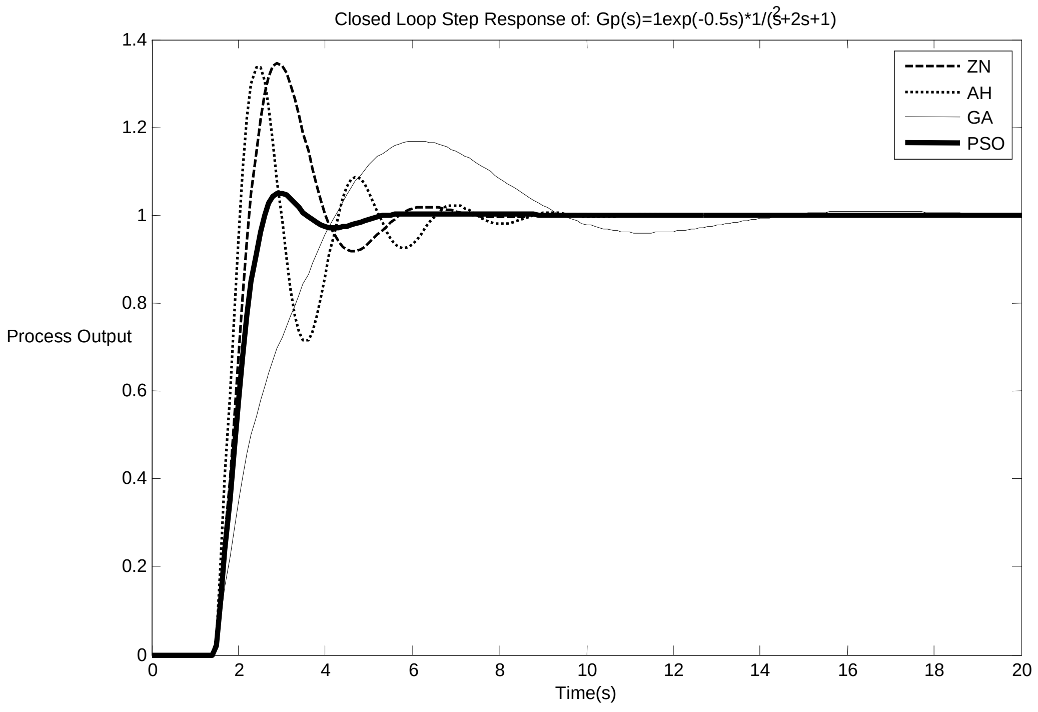


Figure 8.4: SOPDT system responses for Experiment 8.2 $\left(G_p(s) = \frac{\exp(-0.5s)}{s^2 + 2s + 1} \right)$

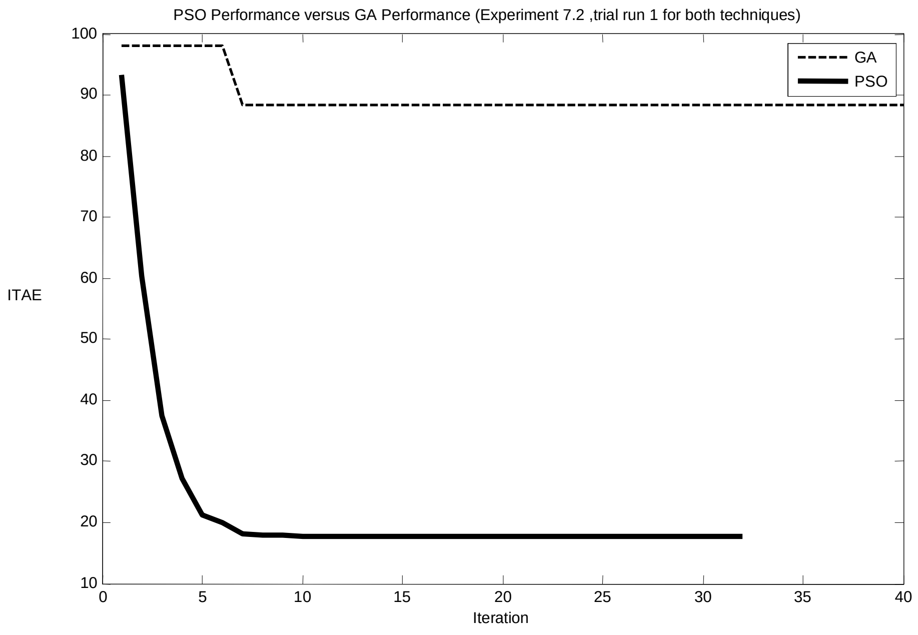


Figure 8.5: ITAE convergence for PSO *vs.* GA $\left(G_p(s) = \frac{\exp(-0.5s)}{s^2 + 2s + 1} \right)$

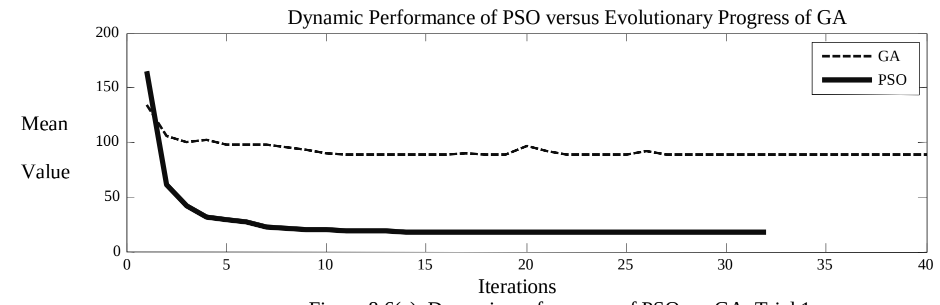


Figure 8.6(a): Dynamic performance of PSO vs. GA: Trial 1

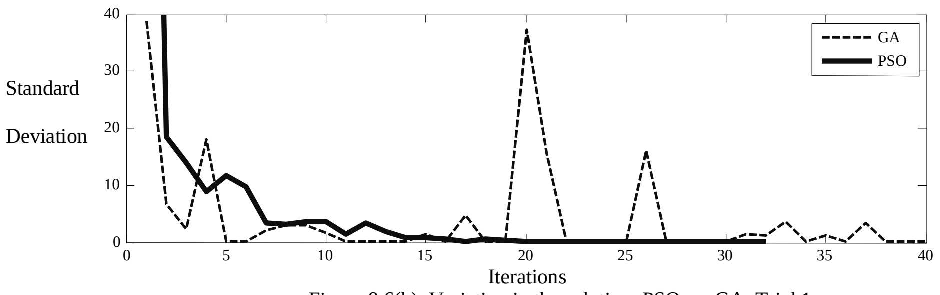


Figure 8.6(b): Variation in the solution: PSO vs. GA: Trial 1

Figure 8.6: Statistical analysis for PSO *vs.* GA $\left(G_p(s) = \dfrac{\exp(-0.5s)}{s^2 + 2s + 1} \right)$

	GA						PSO					
	ITAE	K_c	T_i	T_d	Time(s)	Iter	*ITAE*	K_c	T_i	T_d	Time(s)	Iter
Mean ($\overline{x}$) (After 10 trial runs)	497.8	0.9	0.94	0.86	35	53	12.72	5.16	2.17	0.55	17.5	31
Standard Deviation(σ) (After 10 trial runs)	303.7	0.34	0.22	0.07	17.4	29	0	0.02	0.03	0.01	2.99	5.4

Table 8.6: Statistical analysis of the 10 trial runs for PSO *vs.* GA for Experiment 8.2 $\left(G_p(s) = \dfrac{\exp(-0.5s)}{s^2 + 2s + 1} \right)$

8.5 Experiment 8.3: Tuning of SOIPDT process for optimal setpoint tracking

8.5.1 Objective

The objective of this experiment is to compare the performance of the PSO tuning methodology to that of the Poulin and Pomerleau (PP) and GA method for a SOIPDT process model. The PP method has been chosen in this comparison study because it was developed for integrating processes.

8.5.2 Methodology

The SOIPDT model used in the experiment is given by:

$$G_p(s) = \frac{\exp(-0.2s)}{s(s+1)} \qquad \text{Equation (8.3)}$$

The PID controller is tuned for optimal setpoint tracking.

8.5.3 Observations and analysis of results

The PID tuning parameters and closed-loop dynamic performance specifications are shown in Table 8.7 and Figure 8.7 respectively. The PP tuning method delivers a response that has less overshoot than the PSO technique. The GA method produces a highly oscillatory system with excessive overshoots and undershoots. On the other hand, the PSO tuned PID provides a closed-loop system which delivers improvements in rise and settling time.

With regards to the statistical analysis given in Table 8.8, the PSO optimization algorithm required 31 iterations to find the solution within 17.5 seconds. The GA, on the other hand, required 53 iterations and required twice the amount of time to yield a solution for the same experiment.

Tuning Method	**PID Parameters**			**Dynamic Performance Specifications**			**Performance Index**
	K_c	T_i	T_d	t_r	t_s (2%)	M_p (%)	$ITAE$
PP	0.54	7.09	0.86	2.5	19	17.8	196.3
GA	0.9	0.94	0.86	0.9	40	73.7	996
PSO	5.16	2.17	0.55	0.2	3.8	54.2	12.7

Table 8.7: PID parameters and closed-loop response specifications for Experiment 8.3 $\left(G_p(s) = \dfrac{\exp(-0.2s)}{s(s+1)}\right)$

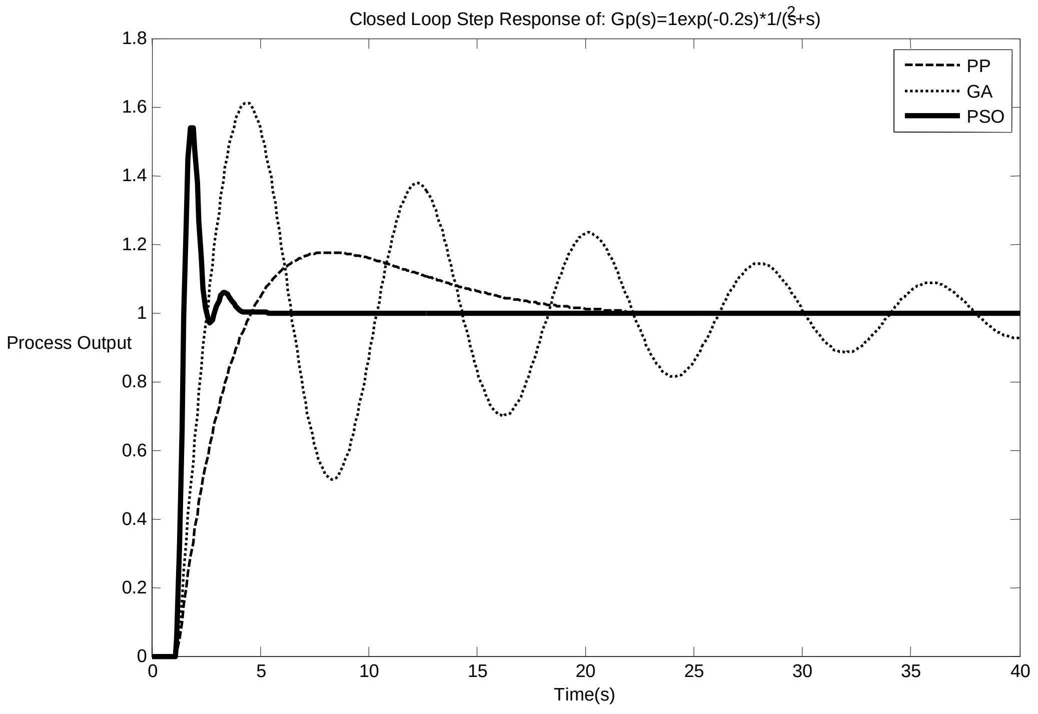


Figure 8.7: SOIPDT system responses for Experiment 8.3 $\left(G_p(s) = \dfrac{\exp(-0.2s)}{s(s+1)} \right)$

	GA						PSO					
	ITAE	K_c	T_i	T_d	Time(s)	Iter	*ITAE*	K_c	T_i	T_d	Time(s)	Iter
Mean($\bar{x}$) (After 10 trial runs)	497.8	0.9	0.94	0.86	35	53	12.72	5.16	2.17	0.55	17.5	31
Standard Deviation(σ) (After 10 trial runs)	303.7	0.34	0.22	0.07	17.4	29	0	0.02	0.03	0.01	2.99	5.4

Table 8.8: Statistical analysis of the 10 trial runs for PSO *vs.* GA for Experiment 8.3 $\left(G_p(s) = \dfrac{\exp(-0.2s)}{s(s+1)} \right)$

8.6 Experiment 8.4: Tuning of FODUP process for optimal setpoint tracking

8.6.1 Objective

The objective of this experiment is to compare the PSO tuning to that of the methods of De Paor and O'Malley (DO) (1989) and Venkatashankar and Chidambaram (VC) (1994) for FODUP. These tuning methods have been chosen because they are based upon controller design for open-loop unstable processes.

8.6.2 Methodology for experiment

The FODUP model considered in the experiment is given by (8.4):

$$G_p(s) = \frac{\exp(-0.2s)}{(s-1)} \qquad \text{Equation (8.4)}$$

A PI controller is utilized to control an open-loop unstable process and is tuned for optimal setpoint tracking.

8.6.3 Observations and analysis of results

The PI tuning parameters and dynamic closed-loop performance specifications are shown in Table 8.9 and Figure 8.8 respectively. Analyses of the results indicate that the tuning methods of DO provide oscillations and excessive overshoot. Conversely, the method of VC delivers no oscillations but suffers severely from longer settling time, that are caused by the weak integral action provided by the tuning algorithm.

Tuning Method	PID Parameters			Dynamic Performance Specifications			Performance Index
	K_c	T_i	T_d	t_r	$t_s(2\%)$	$M_p(\%)$	$ITAE$
DO	1.7	1.35	0	0.3	17	122.7	197.5
VC	2.4	19.6	0	0.4	33.1	58.5	953.7
PSO	3.83	1.36	0	0.2	3.7	97.2	16.9

Table 8.9: PI parameters and closed-loop response specifications for Experiment 8.4 $\left(G_p(s) = \dfrac{\exp(-0.2s)}{(s-1)} \right)$

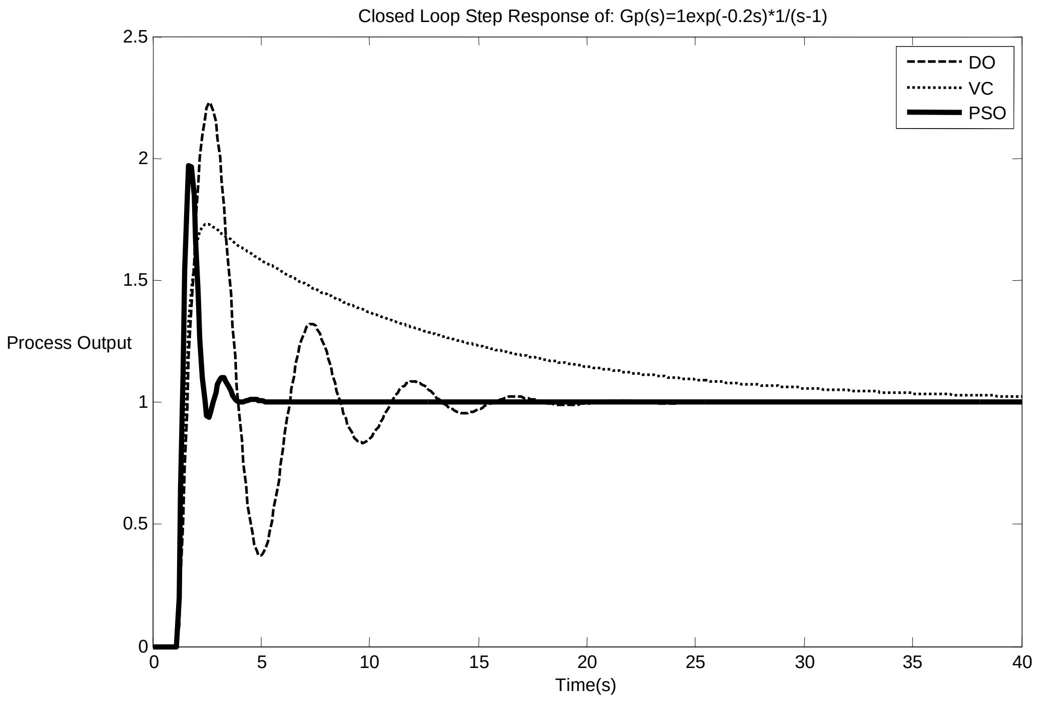


Figure 8.8: FODUP system responses for Experiment 8.4 $\left(G_p(s) = \dfrac{\exp(-0.2s)}{(s-1)} \right)$

The GA tuned system is not shown in Figure 8.8 due to its unstable closed-loop response. Due to the premature convergence characteristic of the GA, the method was unable to find a suitable solution for closed loop stability. The PSO tuned PI controller generates a superior control performance in terms of improved rise time and settling time, with marginally larger overshoot than the VC method.

8.7 Experiment 8.5: Tuning of FOPDT processes for setpoint tracking and disturbance rejection.

8.7.1 Objective

The objective of this experiment is to demonstrate the effectiveness of the PSO method to tune the PID controller for setpoint tracking and disturbance rejection. The ZN, CC and GA tuning methodologies are chosen for comparison with the PSO technique. The ZN and CC methods have been chosen for this experiment since they have been design for load disturbance rejection.

8.7.2 Methodology

The FOPDT model considered in the experiment is given by (8.5):

$$G_p(s) = \frac{\exp(-0.2s)}{(s+1)} \qquad \text{Equation (8.5)}$$

A unit step load disturbance $d(t)$ is introduced into the process input at t_{dist}*=10 seconds*.

8.7.3 Observations and analysis of results

The PID tuning parameters and dynamic closed-loop performance specifications are given in Table 8.10; Figure 8.9 shows the closed-loop response of the system following a step input and disturbance signal. From these results, the Ziegler-Nichols and Cohen-Coon methods produce an oscillatory response with high overshoots and longer settling time following a setpoint change.

Tuning Method	PID Parameters			Dynamic Performance Specifications			Performance Index
	K_c	T_i	T_d	t_r	t_s (2%)	M_p (%)	$ITAE$
ZN	6	0.4	0.1	0.1	12	78.4	29.5
CC	7.02	0.46	0.07	0.1	13.1	93.1	53.1
GA	1.07	0.32	0.07	0.6	14.6	34.4	78.9
PSO	4.92	0.40	0.07	0.1	10.8	50.7	15

Table 8.10: PID parameters and closed-loop response specifications for Experiment 8.5 $\left(G_p(s) = \frac{\exp(-0.2s)}{(s+1)} \right)$

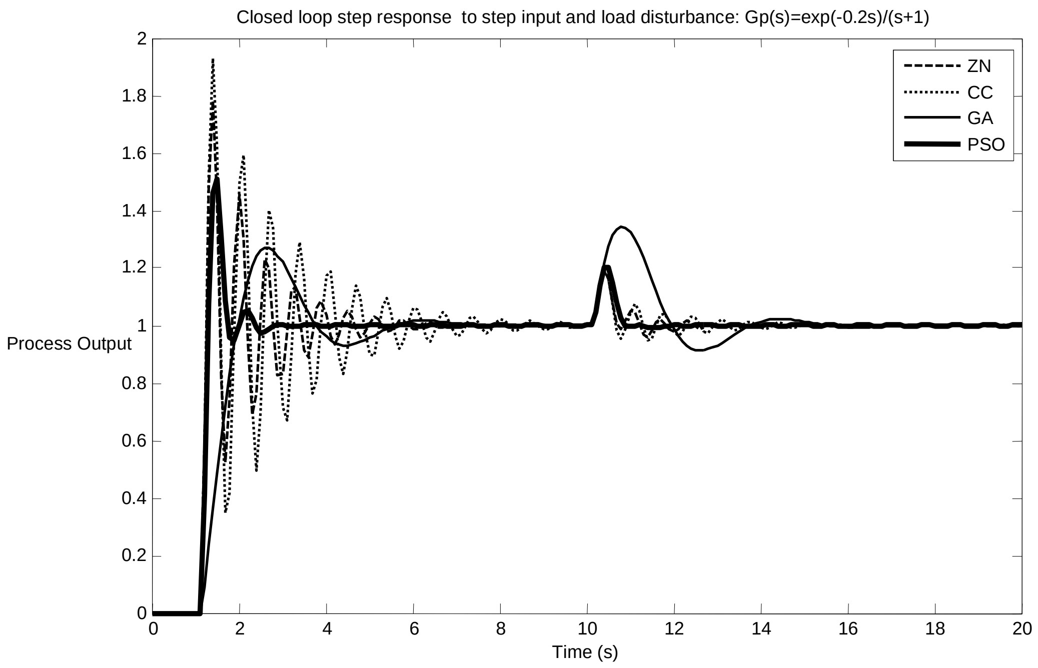


Figure 8.9: FOPDT system responses for setpoint tracking and disturbance rejection (Experiment 8.5) $\left(G_p(s) = \frac{\exp(-0.2s)}{(s+1)} \right)$

ZN and the CC tuning results in oscillatory set-point tracking and a poor recovery from load disturbances. Using the GA and the PSO methodology, we can tune for improved servo tracking and regulatory control, albeit system recovery from disturbances takes longer for system's tuned with the GA.

The statistical evaluation of system performance for PSO and GA tuning is given in Table 8.11. These two methods were selected for this analysis because they are regarded as being computational based evolutionary algorithms. From Table 8.11:

The GA method reaches a mean ITAE of 87.04 within 86 iterations whilst the PSO required 37 iterations for an ITAE of 14.99. In addition, the PSO method delivers tuning parameters that are more repeatable as is evident from the smaller standard deviations for all parameters considered in this analysis.

	GA						**PSO**					
	ITAE	K_c	T_i	T_d	Time(s)	Iter	*ITAE*	K_c	T_i	T_d	Time(s)	Iter
Mean($\bar{x}$) (After 10 trial runs)	87.04	1.07	0.32	0.07	44.63	86	14.99	4.92	0.40	0.07	17.36	37
Standard Deviation(σ) (After 10 trials)	3.77	0.20	0.03	0.07	10.36	17	0.01	0.01	0.01	0.00	3.64	8

Table 8.11: Statistical analysis of the 10 trial runs for PSO and GA for Experiment 8.5 $\left(G_p(s) = \frac{\exp(-0.2s)}{(s+1)} \right)$

8.8 Experiment 8.6: Tuning of SOPDT processes for setpoint tracking and disturbance rejection.

8.8.1 Objective

The objective of this experiment is aimed at comparing the PSO tuning method to that of the ZN, AH and GA for setpoint tracking and disturbance rejection. These methods have been chosen for the reasons previously mentioned.

8.8.2 Methodology

The SOPDT model considered in the experiment is given by (8.6):

$$G_p(s) = \frac{\exp(-0.5s)}{s^2 + 2s + 1} \qquad \text{Equation (8.6)}$$

A unit step load disturbance $d(t)$ is introduced into the process input at t_{dist}*=10 seconds*.

8.8.3 Observations and analysis of results

The PID tuning parameters and dynamic closed-loop performance specifications are given in Table 8.11; Figure 8.10 shows the closed-loop response of the system following a step input and disturbance signal. The PSO tuning method produces a high peak overshoot to setpoint change but delivers the best recovery to load disturbance. In contrast the GA gives a very sluggish response confirming that it may not be suited for processes of this nature. The ZN response is marginally slower compared to the AH and PSO methods but shows less sensitivity to load disturbance to that of the AH tuning.

Tuning Method	PID Parameters			Dynamic Performance Specifications			Performance Index
	K_c	T_i	T_d	t_r	t_s (2%)	M_p (%)	*ITAE*
ZN	2.82	1.65	0.41	0.7	13.8	34.6	101.8
AH	3.13	2.5	0.63	0.4	17	33.6	137.7
GA	1.1	0.77	0.83	8.9	19.4	35.2	293.9
PSO	3.31	1.33	0.53	0.4	13.1	49	86.5

Table 8.11: PID parameters and closed-loop response specifications for Experiment 8.6 $\left(G_p(s) = \dfrac{\exp(-0.1s)}{s^2 + 2s + 1}\right)$

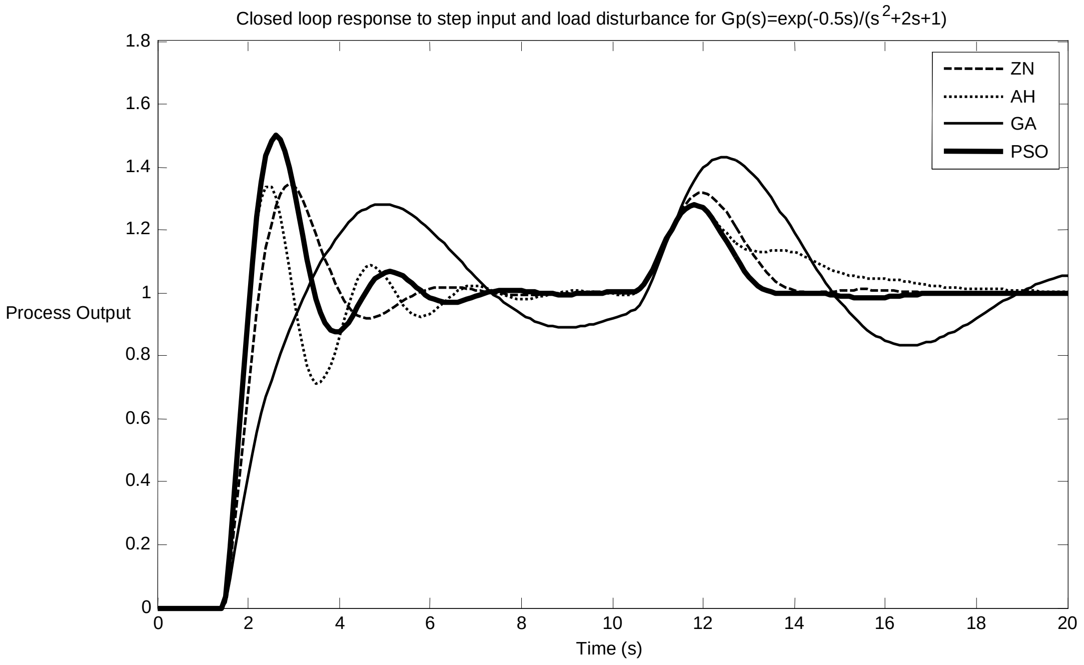


Figure 8.10: SOPDT system responses for Experiment 8.6 $\left(G_p(s) = \dfrac{\exp(-0.5s)}{s^2 + 2s + 1}\right)$

8.9 Experiment 8.7: Tuning of SOIPDT processes for setpoint tracking and disturbance rejection.

8.9.1 Objective

The objective of this experiment is aimed at comparing the PSO tuning method to that of the PP and GA for setpoint tracking and disturbance rejection of SOIPDT process.

8.9.2 Methodology

The SOIPDT model considered in the experiment is given by (8.7):

$$G_p(s) = \frac{\exp(-0.2s)}{(s^2 + s)} \qquad \text{Equation (8.7)}$$

A unit step load disturbance $d(t)$ is introduced into the process input at t_{dist}*=20 seconds*.

8.9.3 Observations and analysis of results

The PID tuning parameters and dynamic closed-loop performance specifications are given in Table 8.12; Figure 8.11 shows the closed-loop response of the system following a step input and disturbance signal. The GA tuning produces an unacceptable response as the system oscillates erratically to system input and load changes. The GA tuned response will eventually reach setpoint if given adequate time. The PP method gives a slow initial response and does not respond well to load disturbance since it results in a very high overshoot. The PSO tuning method outperforms the other methods in this experiment for all the performance specifications considered.

Tuning Method	PID Parameters			Dynamic Performance Specifications			Performance Index
	K_c	T_i	T_d	t_r	t_s (2%)	M_p (%)	$ITAE$
PP	0.58	5.91	0.83	2.2	37	142.5	2899
GA	0.72	0.85	0.73	1.5	40	135.7	5201.8
PSO	7.64	0.85	0.47	0.2	21.6	96.6	52.5

Table 8.12: PID parameters and closed-loop response specifications for Experiment 8.7 $\left(G_p(s) = \dfrac{\exp(-0.2s)}{(s^2+s)} \right)$

2

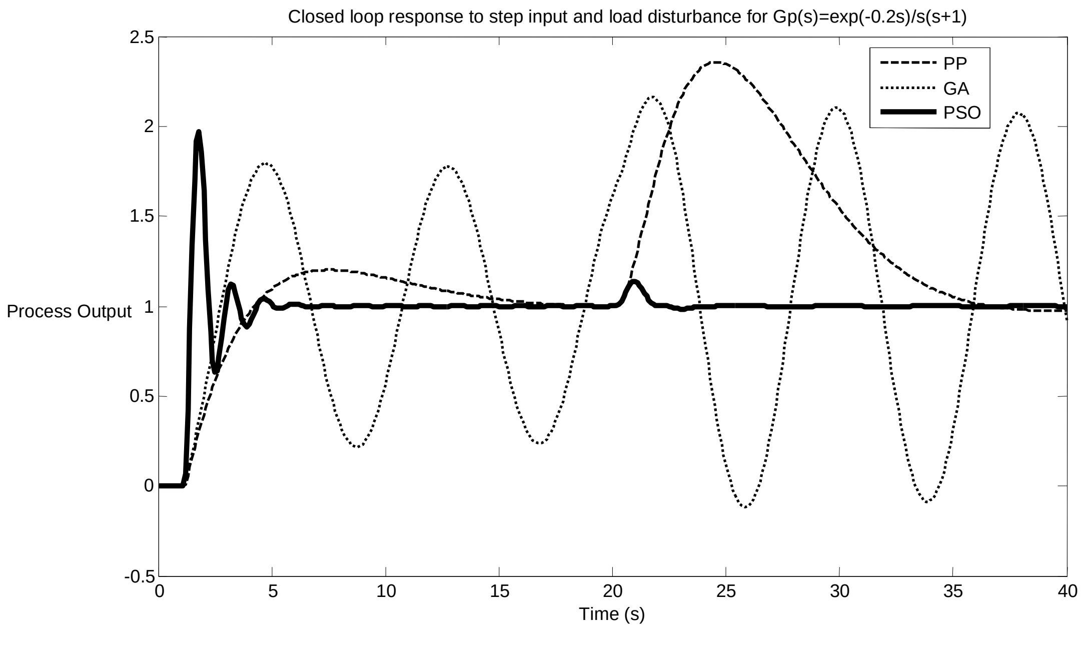


Figure 8.11: SOIPDT system responses for Experiment 8.7 $\left(G_p(s) = \dfrac{\exp(-0.2s)}{(s^2+s)} \right)$

8.10 Experiment 8.8: Tuning of FODUP processes for setpoint tracking and disturbance rejection.

8.10.1 Objective

The objective of this experiment is aimed at comparing the PSO tuning method to that of the DO, VC and GA for setpoint tracking and disturbance rejection of FODUP process.

8.10.2 Methodology

The FODUP model considered in the experiment is given by (8.8):

$$G_p(s) = \frac{\exp(-0.2s)}{(s-1)} \qquad \text{Equation (8.8)}$$

A unit step load disturbance $d(t)$ is introduced into the process input at t_{dist}=*20 seconds*. A PI controller is utilized to control an open-loop unstable process.

8.10.3 Observations and analysis of results

The PID tuning parameters and dynamic closed-loop performance specifications are given in Table 8.13; Figure 8.12 shows the closed-loop response of the system following a step input and disturbance signal. The GA tuning did not result in a stable system and is not chosen for this experiment. The VC tuned response was expected because of its high integral time constant and the DO gave an oscillatory response. Overall, the PSO yielded the best system recovery from load disturbance as is evident from its faster settling time shown in Table 8.12.

Tuning Method	**PID Parameters**			**Dynamic Performance Specifications**			**Performance Index**
	K_c	T_i	T_d	t_r	t_s (2%)	M_p (%)	*ITAE*
DO	1.7	1.35	0	0.3	33.7	122.7	774
VC	2.4	19.6	0	0.4	40	58.5	2829
PSO	4.41	1.23	0	0.2	22.3	108.9	80.9

Table 8.12: PID parameters and closed-loop response specifications for Experiment 8.8 $\left(G_p(s) = \dfrac{\exp(-0.2s)}{(s-1)}\right)$

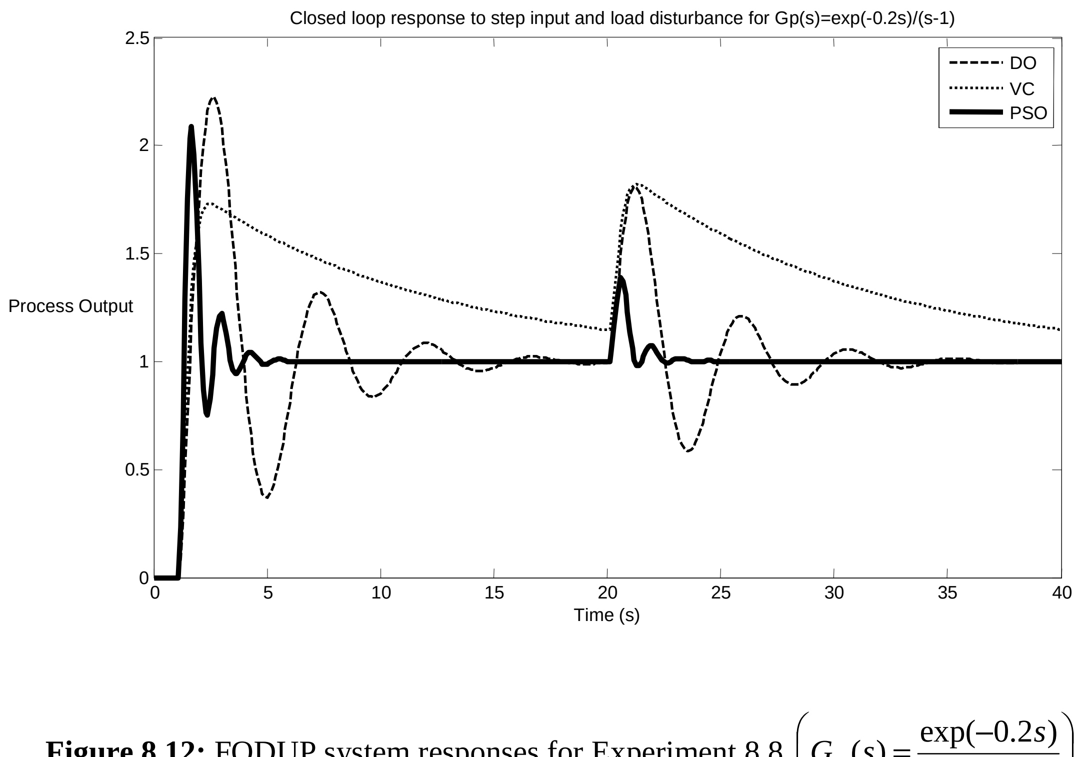


Figure 8.12: FODUP system responses for Experiment 8.8 $\left(G_p(s) = \dfrac{\exp(-0.2s)}{(s-1)}\right)$

8.11 Summary and conclusion

This chapter has presented a comparative study of closed-loop system performance for FOPDT systems that were tuned using the proposed PSO methodology *vs.* ZN, CC and GA tuning. The dynamical performance of the PSO tuned system outperforms that of the same system tuned with ZN, CC and the GA for the following reasons:

i) The ZN and CC methods provide only initial tuning parameters. Fine tuning for an improved response depends on the experience and intuition of the control practitioner.

ii) The PSO method does not suffer from premature convergence – this is not true for the GA.

iii) The high degree of stochasticity that the GA suffers from means that there is a strong possibility of the algorithm yielding poor results over a small number of iterations. Improvements in tuning performance can be achieved if the GA is run for a greater number of iterations – this comes at the cost of increased computational computation burden and process delays.

iv) The GA depends on genetic operators. This implies that even weak solutions could contribute to the composition of future candidate solutions.

v) GA's operate according to a sharing mechanism during their evolutionary process whereby the previous solutions are potentially lost whilst the PSO relies on a memory based progression (Engelbrecht, 2002). This ability to 'remember' its previous best solution means that the PSO can converge much faster than the GA on an optimal solution.

The subsequent chapters will discuss the application of the PSO tuning methodology on real-life systems.

Chapter 9

Offline Tuning for Process Control

9.1 Introduction

This chapter describes tests that were conducted to assess the effectiveness of *offline* PSO tuning for process control. In this approach the transfer function of the process was determined and utilized for the PSO tuning using simulation. The P, I and D tuning parameters determined from the PSO tuning methodology was applied to a process plant that is housed in the instrumentation laboratory at the Durban University of Technology. The main variables under control are water flow, pump discharge pressure and tank level.

9.2 Basic description of the Process Control Plant used in the study

The plant used for this study is given in Figure 9.1.and its corresponding P&ID schematic is shown in Figure 9.2. The plant consists of a storage tank, process tank, feed water pump, control valve and pressure, level and flow transmitters. A feed-pump supplies water from the storage tank to the process tank. The pressure transmitter (PT03) and the flow meter (FT01), which is situated downstream of the pump, provides an indication of pump discharge pressure and volume of water moved by the pump. Control valve (CV01) is situated between the flow transmitter and the process tank to manipulate the flow and thus control the discharge pressure and the level of water in the process tank. Control of the water flow rate (FT01), line pressure (PT03) and process tank level (LT02) is achieved separately using control valve (CV01) during each control session. A current-to-pressure (I/P) converter is used to provide correct signal interface to the control valve (CV01), which operates from a 4 bar air supply.

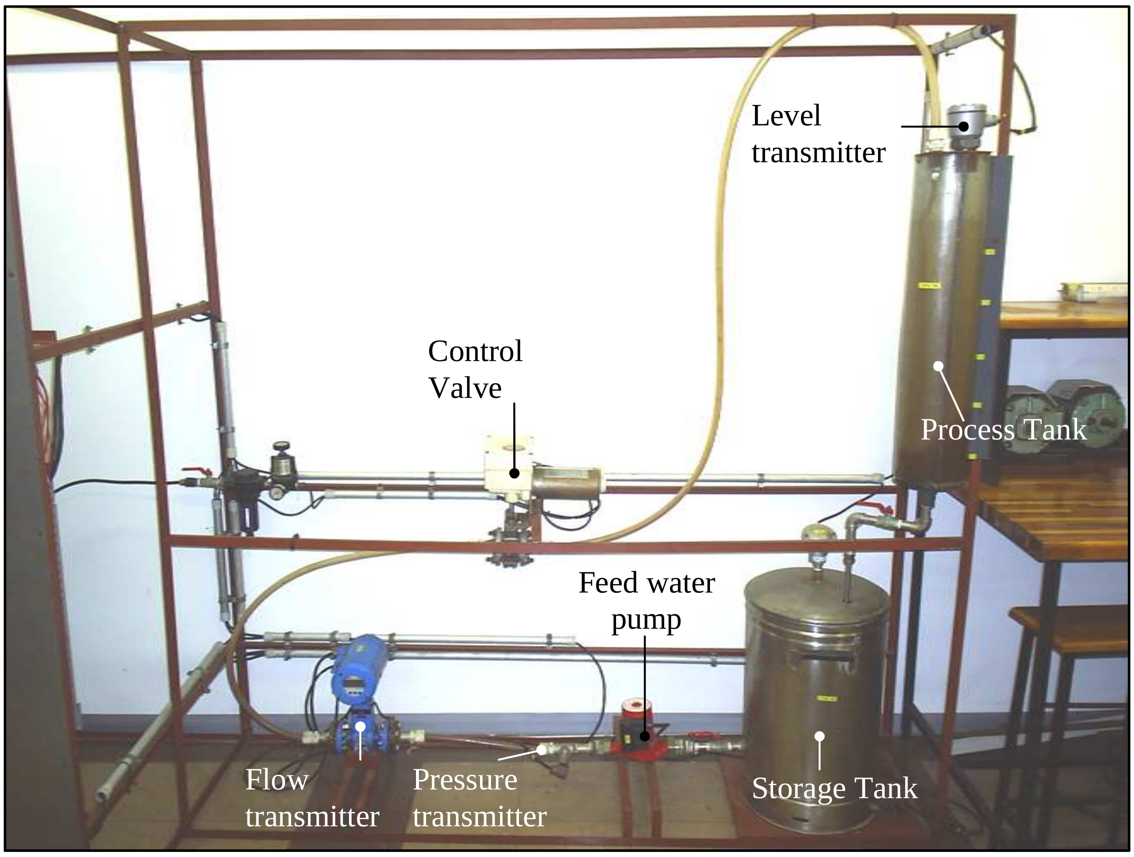


Figure 9.1: Process plant used for the tests

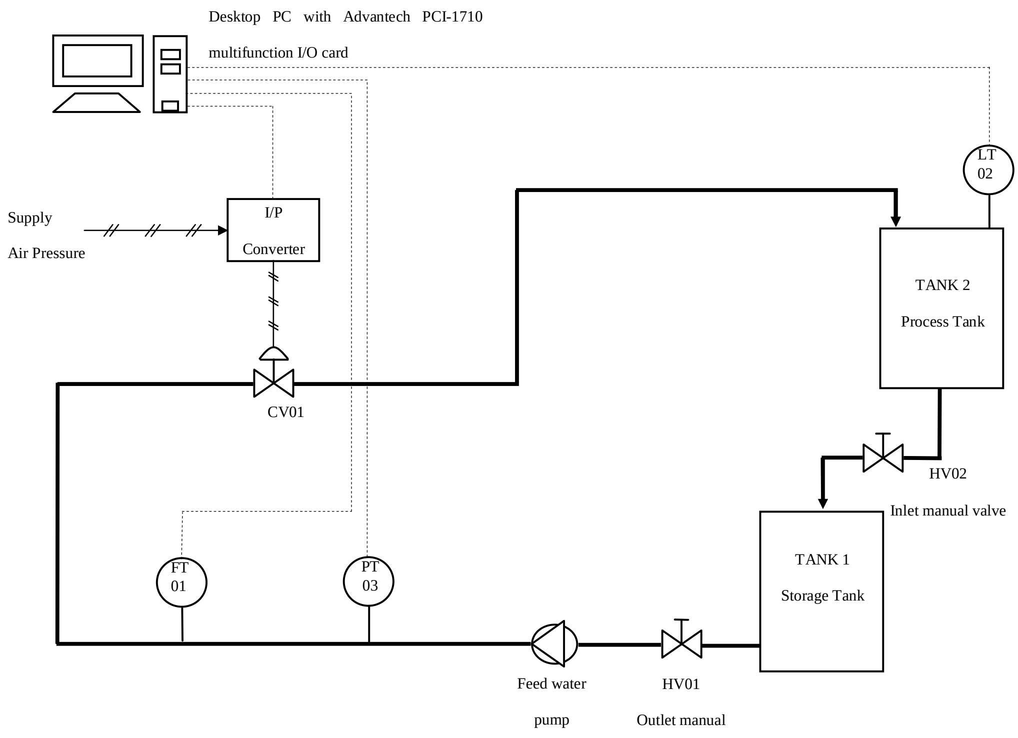


Figure 9.2: P&ID of the plant under study

9.3 Interfacing the plant to the PC based controllers

The PID control algorithm was implemented on a standard Pentium 4 desktop PC. The PC is interfaced to the process control rig using an *Advantech* PCI-1710, 12-bit multifunction I/O card. The PCI-1710 I/O card is supported by MATLAB® 7.3, *Real Time Workshop Toolbox version 6.3*. The PCI-1710 card is capable of 12-bit A/D conversion with up to 100 kHz sampling rate. Electrical connection to each of the devices in the plant is illustrated in Figure 9.3.

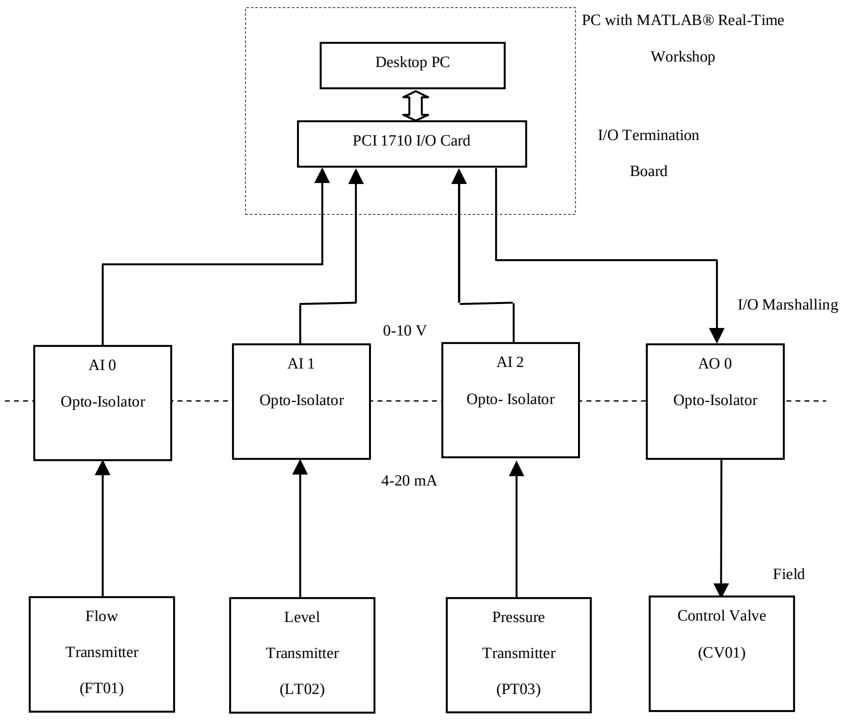


Figure 9.3: Interface between plant and PC

The interaction with the I/O card drivers was done within the MATLAB® Simulink environment and the parameter dialog boxes provided in the *Real-Time Workshop* I/O library. The Simulink model of the control application is shown in Figure 9.4. The analogue I/O channels of the PCI-1710 card are represented by device driver blocks. The rate transition is inserted between the device driver blocks and the 'standard' Simulink blocks to ensure proper data transfer between blocks.

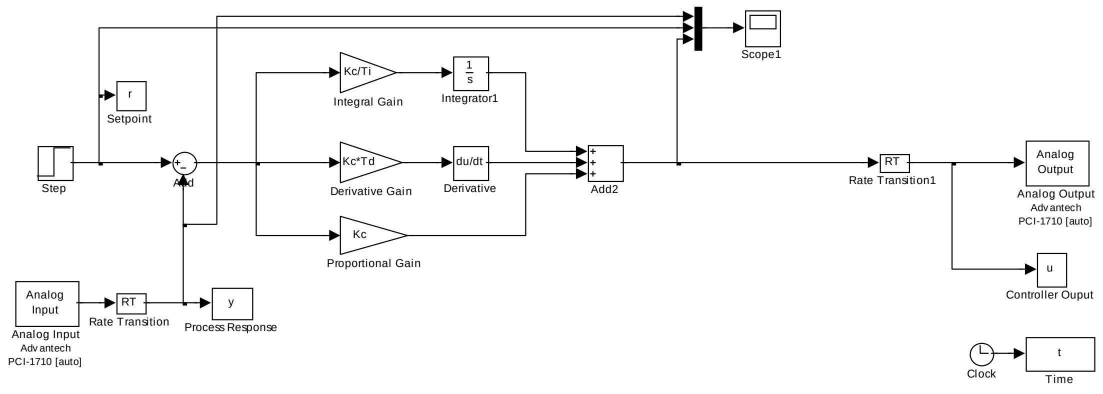


Figure 9.4: MATLAB® Simulink based PID controller for *real-time* control

All the details of the control-loop for flow, level and pressure control are given in Appendix C1.1.

9.4 Preliminaries for the real-time experiments

The process models used in the experiments for the pressure, flow and level control loops are given in (9.1), (9.2) and (9.3) respectively. Models (9.1) – (9.3) were identified with the MATLAB® System *Identification Tool Box*, ver. 6.12. (9.1) - (9.3) were used to determine the tuning parameters by means of the ZN, CC, AH, PP, GA and PSO tuning methodologies.

Pressure control system - FOPDT: $$G_{p_{pressure}}(s) = \frac{0.62\exp(-0.1s)}{(0.5s+1)}$$ Equation (9.1)

Flow control system - SOPDT: $$G_{p_{flow}}(s) = \frac{0.5\exp(-6.5s)}{(1.24s^2+3.5s+1)}$$ Equation (9.2)

Level control system - SOIPDT: $$G_{p_{level}}(s) = \frac{0.02\exp(-5s)}{s(0.76s+1)}$$ Equation (9.3)

Plant models (9.1), (9.2) and (9.3) were used to determine the tuning parameters for each respective control loop. These parameters were then applied to the actual process plant in order to determine the dynamical closed-loop performance of the plant. The PSO and GA methods were each run over ten trials. Details of these trials are provided in Appendix C2.1 to Appendix C2.3. The closed-loop performance for each tuning method was evaluated using its transient response characteristics and the ITAE performance index.

9.5 Pressure control loop

The controllability ratio for the pressure control system in (9.1) is $\frac{L_p}{T_p} = 0.2$. A PI controller was used because experimental results showed that the derivative action caused erratic movement of the control valve due to the presence of valve noise present within the control channel. The tuning parameters for PI control were obtained using ZN (open-loop tuning), CC, GA and PSO tuning techniques and are shown in Table 9.1.

Tuning Method	**PI Parameters**	
	K_c	T_i
ZN	7.26	0.33
CC	7.39	0.24
GA	0.86	0.02
PSO	4.53	0.43

Table 9.1: Tuning parameters for the pressure control loop

$$G_{p\,pressure}(s) = \frac{0.62\exp(-0.1s)}{(0.5s+1)}$$

The statistical analysis over the ten trials for PSO and GA tuning is given in Table 9.2. The reasons for choosing only these two methods were mentioned in Chapter 8.

	GA						PSO					
	ITAE	K_c	T_i	T_d	Time(s)	Iter	*ITAE*	K_c	T_i	T_d	Time(s)	Iter
Mean($\bar{x}$) (After 10 trial runs)	30.79	0.86	0.02	0.00	35.26	52	2.41	4.53	0.43	0.00	8.27	20
Standard Deviation(σ) (After 10 trial runs)	15.85	0.09	0.05	0.00	10.27	14	0.01	0.05	0.01	0.00	1.36	3

Table 9.2: Statistical analysis over the 10 trials for PSO and GA for pressure control loop $\left(G_{p_{pressure}}(s) = \dfrac{0.62\exp(-0.1s)}{(0.5s+1)} \right)$

9.5.1 Results and observations

From Table 9.2, the PSO tuned system displays a better performance than the GA by achieving a mean ITAE of 2.41 as opposed to 30.79 for the GA; also the PSO results are more repeatable as is evident from its small standard deviation of 0.01. The closed-loop step response for the different tuning methods is illustrated in Figure 9.5. For ease of viewing, the step responses for each of the tuning methods are show separately. The response specifications and performance index for the pressure control loop are given in Table 9.3.

From Figure 9.5 and Table 9.3, the GA method yields a system with higher overshoot, longer settling and rise time in comparison to other methods. The closed-loop response for the ZN and CC methods are similar with the CC method yielding marginally higher overshoot and longer settling time. The PSO method delivers superior control performance with improved dynamic performance specifications over the other tuning methods.

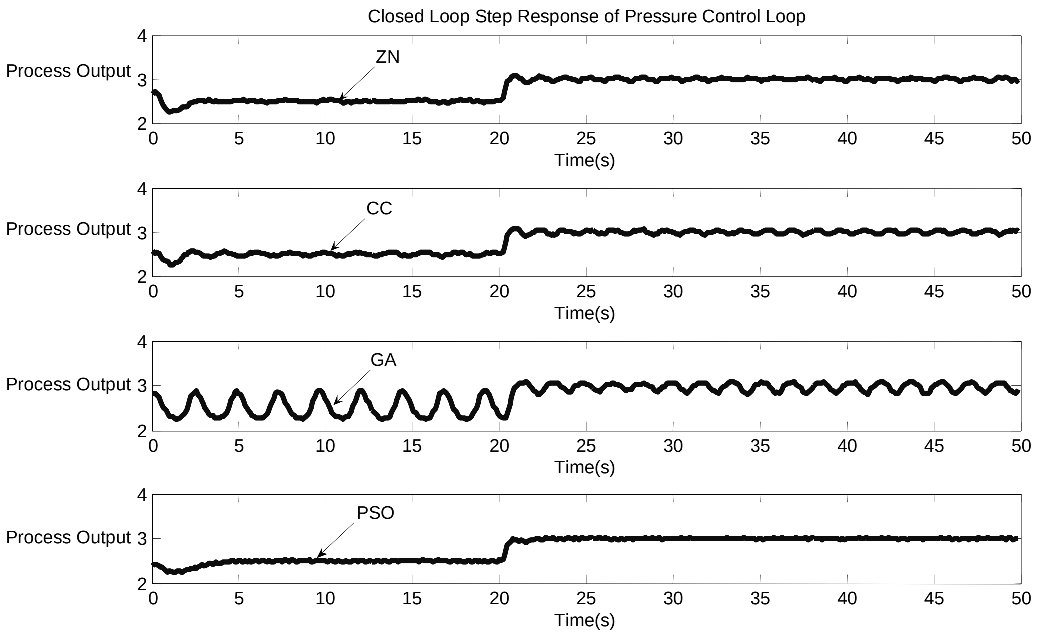


Figure 9.5: Closed-loop step responses of the pressure control loop using ZN, CC, GA and PSO tuning parameters $\left(G_{p_{pressure}}(s) = \dfrac{0.62\exp(-0.1s)}{(0.5s+1)} \right)$

Tuning Method	**Dynamic Performance Specifications**			**Performance Index**	
	t_r	t_s (2%)	M_p (%)	$ITAE$	$ITAE_{PSO} / ITAE_{ZN,CC,GA}$
ZN	0.5	48.8	1.6	293.9	0.63
CC	0.5	49.1	3.7	442	0.42
GA	0.7	49.8	5.8	1266.4	0.15
PSO	0.5	21.6	0.9	186.2	-

Table 9.3: Closed-loop performance of the pressure control loop using ZN, CC, GA and PSO tuning methods $\left(G_{p_{pressure}}(s) = \dfrac{0.62\exp(-0.1s)}{(0.5s+1)} \right)$

9.6 Flow Control

The controllability ratio for the flow process (9.2) is $\frac{L_p}{T_p} = 22.75$, making the system dead-time dominant. A PI controller was used in this experiment since derivative action is not recommended for dead-time dominant processes (Åström and Hägglund, 2004; Hägglund, 1992). Using (9.2), the PID tuning parameters are obtained by applying the ZN (closed-loop tuning), AH, GA and PSO tuning techniques. The ultimate gain (K_u) and ultimate period (P_u) of the process was heuristically determined. The closed-loop system under sustained oscillation is illustrated in Figure 9.6.

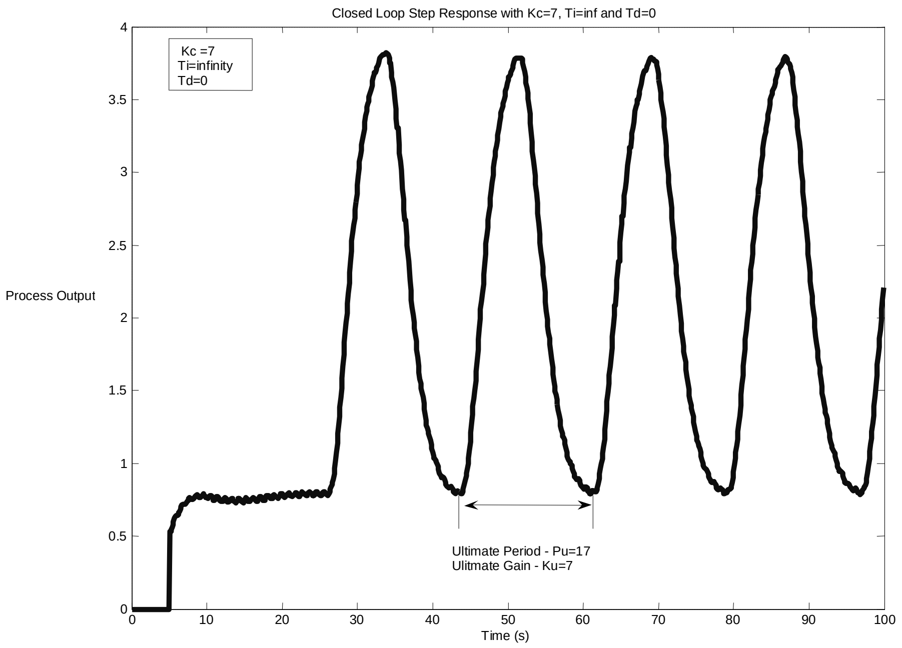


Figure 9.6: Closed-loop step response of the flow control loop with $K_c = 7, T_i = \infty$ and $T_d = 0$ $\left(G_{p_{flow}}(s) = \frac{0.5\exp(-6.5s)}{1.24s^2 + 3.5s + 1} \right)$

With regards to Figure 9.10, $K_u = 7$ and $P_u = 17$. A phase margin of $\phi_m = 60^o$ was used to determine the tuning parameters for the AH technique. The tuning parameters using the respective tuning methods are shown in Table 9.4. The statistical analysis of the ten trials for PSO and GA tuning is given in Table 9.5. The reasons for choosing only these two methods were mentioned in Chapter 8.

Tuning Method	**PI Parameters**	
	K_c	T_i
ZN	2.8	13.6
AH	3.5	38.99
GA	0.09	0.74
PSO	0.85	5.03

Table 9.4: Tuning parameters for the flow control loop $\left(G_{p_{flow}}(s) = \dfrac{0.5\exp(-6.5s)}{1.24s^2 + 3.5s + 1}\right)$

	GA						**PSO**					
	ITAE	K_c	T_i	T_d	Time(s)	Iter	*ITAE*	K_c	T_i	T_d	Time(s)	Iter
Mean($\bar{x}$) (After 10 trial runs)	2776	0.09	0.74	0.00	25.81	33.5	1013	0.85	5.03	0.00	24.74	36
Standard Deviation (σ) (After 10 trial runs)	194.48	0.04	0.23	0.00	17.49	17.69	0.01	0.00	0.01	0.00	3.93	5.99

Table 9.5: Statistical analysis of the 10 trial runs for PSO and GA for the flow control loop $\left(G_{p_{flow}}(s) = \dfrac{0.5\exp(-6.5s)}{1.24s^2 + 3.5s + 1}\right)$

9.6.1 Results and observations

From Table 9.7, the PSO tuned system displays a better performance than the GA by achieving a mean ITAE of 1013; this is more than doubled (2776) for the GA; also the PSO results are more repeatable as is evident from its small standard deviation of 0.01.

The closed-loop step responses of the PID controller to a step input under different tuning conditions are illustrated in Figure 9.7. The response specifications and performance index for the flow control loop is given in Table 9.8. From Figure 9.7 and Table 9.6, it is evident that both ZN and AH tuning results in an oscillatory response. This confirms that these methods are not suited for processes that are dead-time dominant. The GA tuned system is characterized by slow a rise time and a marginal improvement in overshoot. The delayed response and slow rise time is attributed to the weak proportional gain given by the GA method. Overall the PSO method delivers the best performance as is evident from its ITAE performance index.

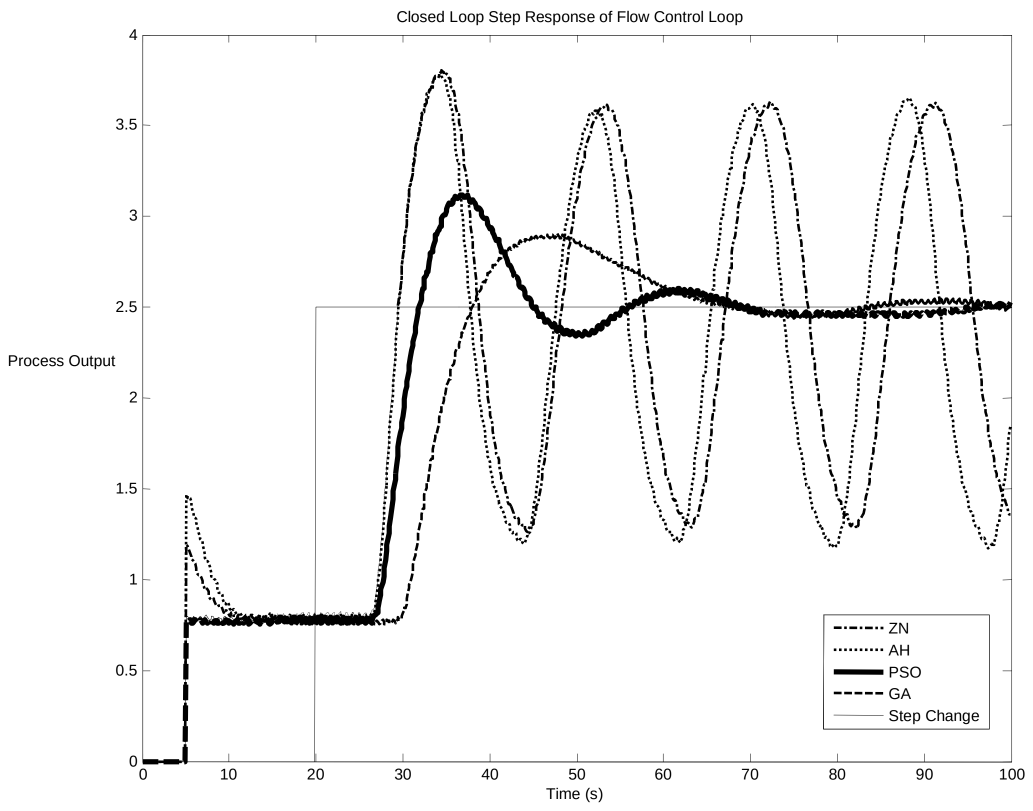


Figure 9.7: Closed-loop step responses of the flow control loop using ZN, AH, GA and PSO tuning parameters $\left(G_{p_{flow}}(s) = \dfrac{0.5\exp(-6.5s)}{1.24s^2 + 3.5s + 1} \right)$

Tuning Method	Dynamic Performance Specifications			Performance Index	
	t_r	t_s	M_p(%)	$ITAE$	$ITAE_{PSO} / ITAE_{ZN,CC,GA}$
ZN	22.5	-	109	39724.4	0.23
AH	23.2	-	105	42596.6	0.21
GA	31.2	93.3	15	11616	0.77
PSO	25.9	77.9	25	8950	-

Table 9.6: Closed-loop performance of the flow control loop using ZN, AH, GA and PSO tuning methods $\left(G_{p_{flow}}(s) = \dfrac{0.5\exp(-6.5s)}{1.24s^2 + 3.5s + 1} \right)$

9.7 Level Control

The controllability ratio for the level process model (9.3) is $\frac{L_p}{T_p} = 2$, hence the need for PI control for this dead-time dominant process. The PI tuning parameters for (9.3) are determined using P-P, GA and PSO tuning and are given in Table 9.7. The statistical analysis over ten trials for PSO and GA tuning is given in Table 9.8. The reasons for choosing only these two methods are the same as was mentioned in Chapter 8.

Tuning Method	**PI Parameters**	
	K_c	T_i
PP	7.03	17.29
GA	0.06	0.94
PSO	14.54	5.64

Table 9.7: Tuning parameters for the level control $\left(G_{p_{level}}(s) = \frac{0.02\exp(-3s)}{s(0.76s+1)} \right)$

	GA						**PSO**					
	ITAE	K_c	T_i	T_d	Time(s)	Iter	*ITAE*	K_c	T_i	T_d	Time(s)	Iter
Mean ($\bar{x}$) (After 10 trial runs)	7228.5	0.06	0.94	0.00	27.45	30	6137.5	14.54	5.64	0.00	454.5	361
Standard Deviation (σ) (After 10 trial runs)	16554.7	0.06	0.16	0.00	17.63	23	149.65	0.09	58.1	0.00	48.78	37.6

Table 9.8: Statistical analysis of the 10 trial runs for PSO and GA for the level control loop $\left(G_{p_{level}}(s) = \frac{0.02\exp(-3s)}{s(0.76s+1)} \right)$

9.7.1 Results and observations

From Table 9.10, the PSO tuned system displays a better performance than the GA by achieving a mean ITAE of 6137.5 as compared to 7228.5 for the GA; also the PSO results are more repeatable as is evident from its low small standard deviation of 149.65 *vs.* 16554.7 for the GA.

The closed-loop step responses of the PI controller tuned using the selected tuning methods are illustrated in Figure 9.18. The response specifications and performance index is given in Table 9.8. From Figure 9.18 and Table 9.9, the GA tuned response converges towards the stable region with unacceptable oscillation around the setpoint. The PP method produces a slower response with higher overshoot than the PSO tuned response. The PSO tuned system results in quicker settling time and smaller overshoot when compared to the PP and GA tuning methods.

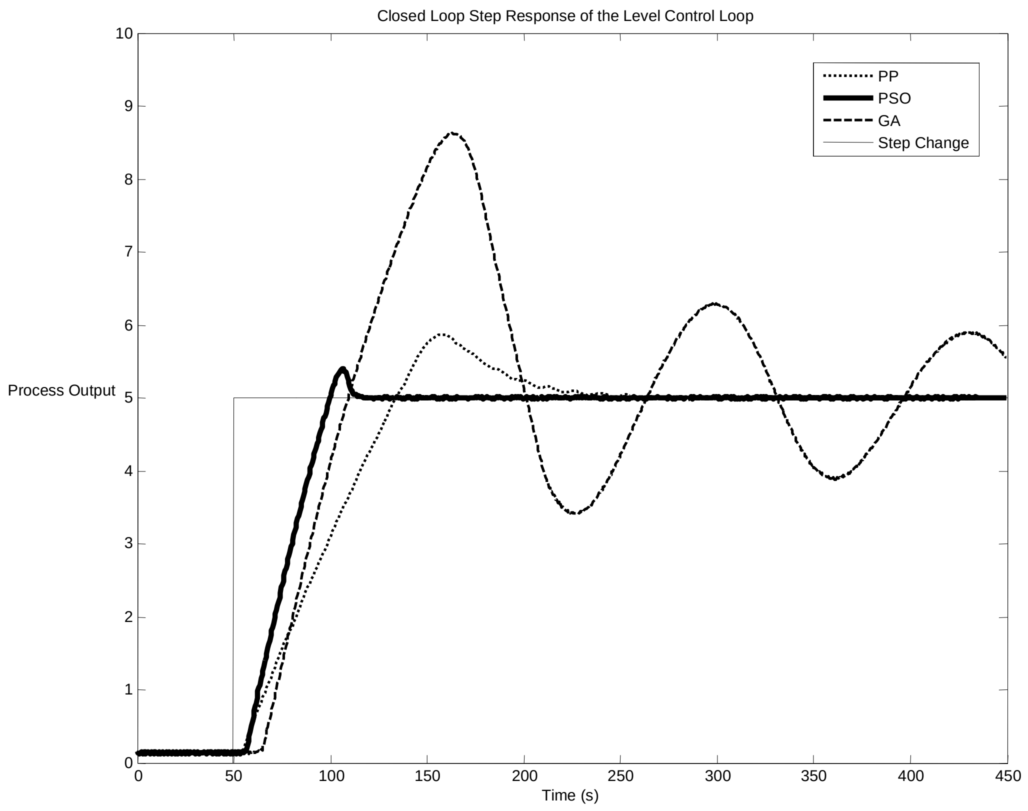


Figure 9.8: Closed-loop step responses of the level control loop using PP, GA and PSO tuning parameters $\left(G_{p_{level}}(s) = \dfrac{0.02\exp(-3s)}{s(0.76s+1)} \right)$

Tuning Method	Dynamic Performance Specifications			Performance Index	
	t_r	$t_s\,(2\%)$	$M_p\,(\%)$	$ITAE$	$ITAE_{PSO} / ITAE_{PP,GA}$
PP	53.6	260	17	28056	0.69
GA	41.5	-	55.3	210911	0.09
PSO	36	114	7.5	19527	-

Table 9.9: Closed-loop performance characteristics for level control loop using PP, GA and PSO tuning $\left(G_{p_{level}}(s) = \dfrac{0.02\exp(-3s)}{s(0.76s+1)} \right)$

9.8 Summary and conclusion

The ZN, CC, AH, PP, GA and PSO tuning methods have been implemented on pressure, flow and level control loops and a comparison of the control performance using these methods has been completed. An analysis of the performance characteristics for all the control loops shows that the PSO method outperforms all the tuning techniques under consideration in this study as is evident from the performance characteristics mentioned in the previous discussions. The next chapter will examine the performance of the PSO algorithm for real-time servo system control.

Chapter 10

Online Tuning for Real-Time Positional Control

10.1 Introduction

This chapter discusses the control performance of a servo positioning system tuned for optimal servo-tracking and regulatory control using the proposed PSO method. Two approaches of using the PSO tuning methodology were studied. The first approach made use of the system process model to determine the controller parameters *offline* using simulation. This is similar to the PSO tuning approach discussed in the previous chapter. In the second approach a model of the process is not required; rather the servo-mechanism is controlled in real-time using the different PID parameters given by the PSO technique. Following each iteration, the PSO selects the best PID parameters based on minimizing the ITAE performance index. This is called the *pre-tuning* phase. This phase is terminated once the PSO algorithm converges on an optimal PID tuning parameter. In both approaches the control performance of the PSO tuned system was compared to other selected tuning techniques.

10.2 Positioning servo-system

The system under study is based on the Modular Servo Positioning Control System (MS150 MKII) from FEEDBACK INSTRUMENTS LTD. The main requirement for the position control system is for the motor to rotate an output shaft to the same angle as the input shaft. In this case the error signal *e(t)* is created by mounting a potentiometer onto the reference shaft and the motor shaft. The potentiometers are connected to equal but opposite DC supplies and the voltages will cancel when each shaft is at the same position. Hence any misalignment between the two shafts will give an error signal proportional the angular displacement. The error signal is used in a closed-loop strategy with a PID algorithm for positional control. The control architecture is implemented with the MATLAB® Real-Time Workshop environment in conjunction with the Advantech I/O (PCI 1710) card. The system is illustrated in Figure 10.1.

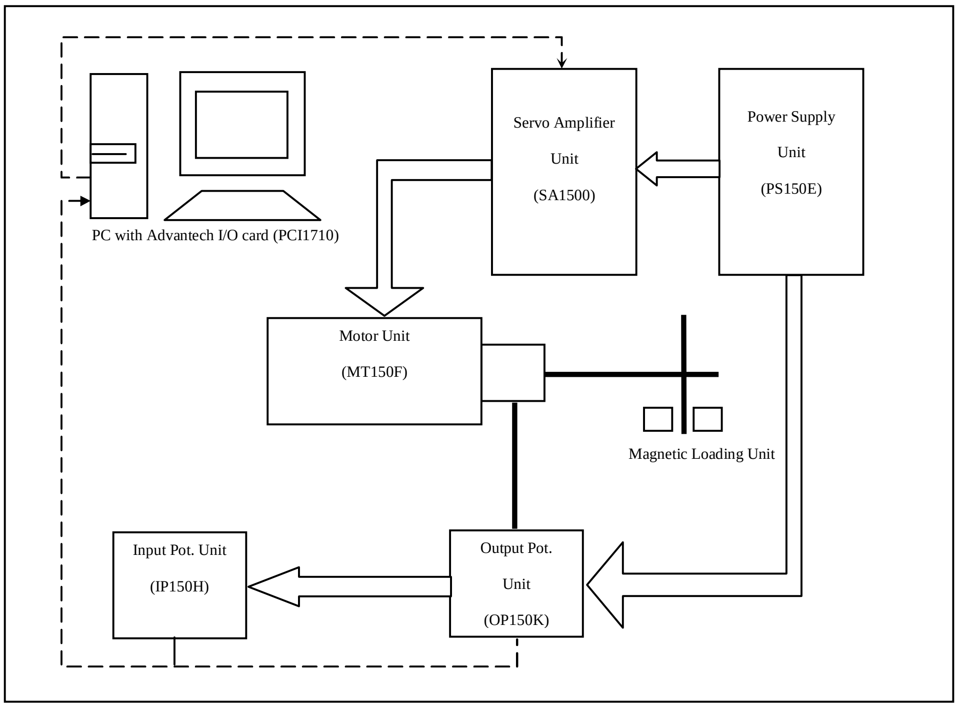


Figure 10.1: Schematic of Servo control system

10.2.1 PID control structure used for the positional servo-system

The objective of the control strategy is to ensure that the dynamical response of the system accurately tracks the setpoint and remains robust to disturbances. This is achieved by utilizing a one degree-of-freedom (1-DOF) PID controller implemented within MATLAB®. The algorithm for the PID controller used in this study is the standard non-interacting PID algorithm:

$$G_c(s) = K_c(1 + \frac{1}{T_i s} + T_d s) \qquad \text{Equation (10.1)}$$

10.2.2 Positioning servo-system control loop

The positioning system is actuated by means of an armature controlled DC motor with gear speed reduction. A schematic of the positioning servo-system is shown in Figure 10.2. The following characteristics applicable to the armature DC motor and the load will be considered for this study:

R_a	=	armature resistance,
L_a	=	armature inductance,
J	=	moment of inertia of motor and load,
f	=	viscous friction coefficient of the motor and load,
K	=	motor torque constant,
K_b	=	back emf constant,
$\theta(s)$	=	angular displacement,

$C(s)$ = angular displacement of motor shaft.

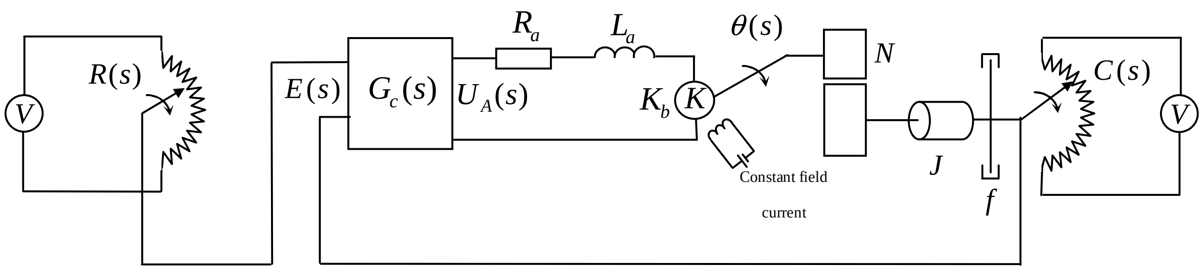


Figure 10.2: Schematic of the positional servo-mechanism

10.2.3 Model of the armature controlled DC motor and gear mechanism

The feedback control loop of the positioning system is shown in Figure 10.3. With regards to Figure 10.3, when considering $U_B(s)$ as input and $\theta(s)$ as output of the system $G_p(s)$, the transfer function of the armature controlled DC motor is:

$$\frac{\theta(s)}{U_B(s)} = \frac{K\exp(-Ls)}{s[LaJs^2 + (Laf + RaJ)s + Raf + KK_b]} \qquad \text{Equation (10.2)}$$

The motor gain and the time constants are given by Equations (10.3) and (10.4) respectively:

$$K_m = \frac{K}{(Raf + KK_B)} \qquad \text{Equation (10.3)}$$

$$T_m = \frac{R_a J}{(Raf + KK_B)}$$ Equation (10.4)

Substitution of Equation (10.3) into (10.4) yields the transfer function for the armature controller DC motor:

$$\frac{\theta(s)}{U_B(s)} = \frac{K_m \exp(-Ls)}{s(T_m s + 1)}$$ Equation (10.5)

The transfer function for the gear mechanism is:

$$\frac{C(s)}{\theta(s)} = N$$ Equation (10.6)

The lumped transfer function for the positional servo-mechanism is a second order integrating system with dead time:

$$\frac{C(s)}{U_B(s)} = \frac{NK_m \exp(-Ls)}{s(T_m s + 1)}$$ Equation (10.7)

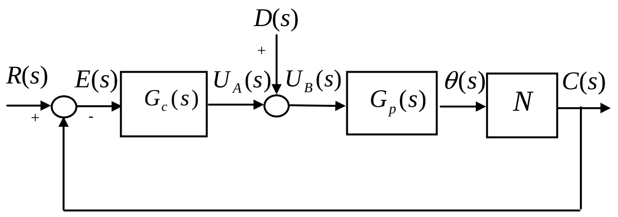


Figure 10.3: Feedback control loop for the positional servo-mechanism

Using the MATLAB® system identification toolbox the following transfer function of the servo positioning system was determined:

$$G_p(s) = \frac{9.65\exp(-0.1s)}{s(0.01s+1)} \qquad \text{Equation (10.8)}$$

10.3 Evaluating PSO Performance for Offline Tuning

ZN, PP, GA and PSO tuning parameters for the servo-system were obtained using the model (10.8) of the positioning control system. These parameters were then used for the *real-time* tests. The ZN and PP methods were chosen because of its widespread applicability to integrating process. Table 10.1 gives the tuning parameters given by the respective methods. The ultimate gain and ultimate period of the system was determined by trial and error.

10.3.1 Observations and analyses of results

10.3.1.1 *Controller tuned for setpoint tracking*

The performance specifications for the system are given in Table 10.2 and the response of the system to setpoint changes are illustrated in Figure 10.4.

Tuning Method	**PID Parameters**		
	K_c	T_i	T_d
ZN	0.98	0.105	0.026
PP	0.55	0.48	0.01
GA	0.98	0.37	0.06
PSO	0.81	5.38	0.05

Table 10.1: PID parameters of the positional servo-mechanism for setpoint tracking
$$\left(G_p(s) = \frac{9.65\exp(-0.1s)}{s(0.01s+1)}\right)$$

Tuning Method	**Dynamic Performance Specifications**			**Performance Index**	
	t_r	t_s	M_p (%)	$ITAE$	$ITAE_{PSO} / ITAE_{ZN,PP,GA}$
ZN	0.08	0.8	94	5791	0.54
PP	0.09	-	109	6620	0.47
GA	0.08	1.18	75.5	4654	0.67
PSO	0.15	0.47	5.6	3153	-

Table 10.2: Closed-loop response specifications for setpoint tracking
$$\left(G_p(s) = \frac{9.65\exp(-0.1s)}{s(0.01s+1)}\right)$$

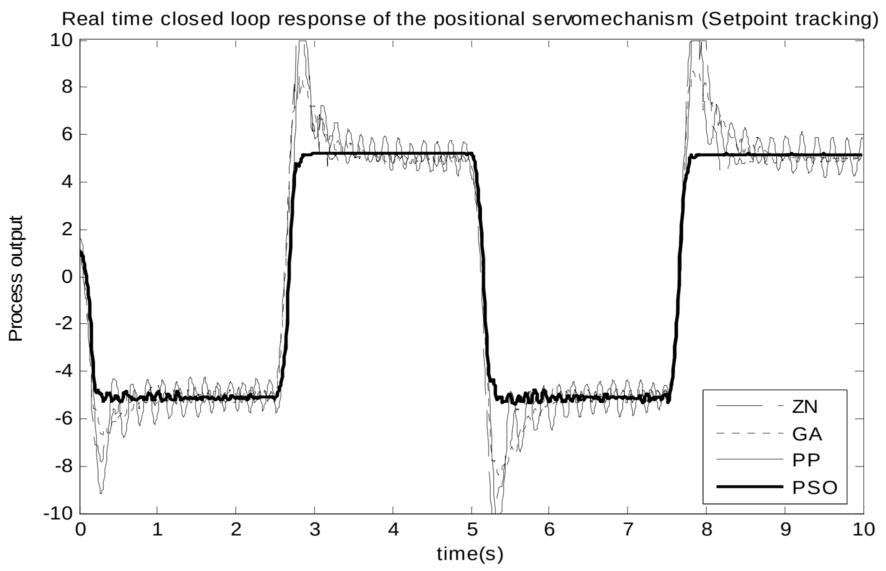


Figure 10.4: Closed-loop setpoint response of the positional servo-mechanism using off-line tuning $\left(G_p(s) = \dfrac{9.65\exp(-0.1s)}{s(0.01s+1)} \right)$

With regards to Table 10.2, it is evident that the PSO tuning method provides the best closed-loop performance in comparison to the other methods.

10.3.1.2 *System tuned for disturbance rejection*

Table 10.3 gives the PID parameters for the respective tuning methods. The results of the experiment are given in Table 10.4 and the response of the system to setpoint change and load disturbance of the process are illustrated in Figure 10.5.

Tuning Method	PID Parameters		
	K_c	T_i	T_d
ZN	0.98	0.105	0.026
PP	0.55	0.41	0.01
GA	0.98	0.25	0.05
PSO	1.24	0.21	0.05

Table 10.3: PID parameters of the positional servo-mechanism for disturbance rejection.

$$\left(G_p(s) = \frac{9.65\exp(-0.1s)}{s(0.01s+1)} \right)$$

Tuning Method	Dynamic Performance Specifications			Performance Index	
	t_r	$t_s\,(2\%)$	$M_p\,(\%)$	$ITAE$	$ITAE_{PSO} / ITAE_{ZN,PP,GA}$
ZN	0.09	0.86	72.5	1093	0.87
GA	0.08	0.85	72.4	1018	0.94
PSO	0.08	0.76	72.5	961	-

Table 10.4: Closed-loop response specifications for disturbance rejection

$$\left(G_p(s) = \frac{9.65\exp(-0.1s)}{s(0.01s+1)} \right)$$

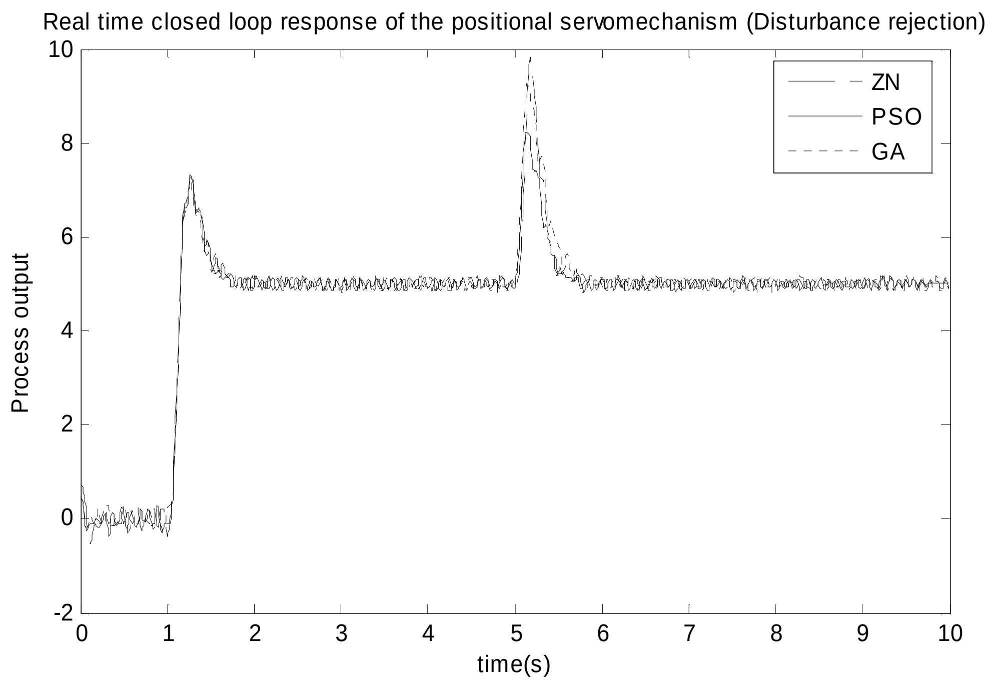


Figure 10.5: Closed-loop setpoint and disturbance response of the positional servo-mechanism using off-line tuning $\left(G_p(s) = \dfrac{9.65\exp(-0.1s)}{s(0.01s+1)} \right)$

With regards to Table 10.4, the PP method resulted in an unstable performance and therefore is deemed unsuitable for tuning disturbance rejection loops of this type. The PSO tuned loop results are similar to those of the ZN and GA, but overall the PSO yielded an improved control performance as is evident from its superior performance index.

10.4 Evaluating PSO performance using online PSO Tuning

The PID control and PSO tuning algorithm was implemented *online* for *real-time* control. PID parameters were obtained during the *pre-tuning* phase. The online PSO tuning method was compared to ZN, PP and GA tuning algorithms. Several tests were conducted in order to determine the minimum number of agents required for the PSO algorithm to be consistent upon convergence. The PSO parameters used in the tests are given Table 10.5. From initial tests it was observed that 1 agent resulted in poor performance of the PSO search and therefore does not form part of the evaluation.

Swarm Size	2,3,4,5
Maximum Velocity (V_{max})	1
Cognitive Acceleration (c_1)	2.05
Social Acceleration (c_2)	2.05
Upper Bound Of Initialization (ub)	1
Lower Bound Of Initialization (lb)	0
Stall Limit- Termination Criterion	10

Table 10.5: PSO parameters used in the test

The swarm size was adjusted within the range given in Table 10.5 and the test was repeated for 4 trial runs. The results of the tests for the setpoint tracking tuning and disturbance rejection tuning are shown in Table 10.6 and Table 10.7 respectively.

With regards to Table 10.6, the best ITAE performance index was found using 3 agents within 8.22 minutes. The standard deviation from the 4 trial runs indicate that the consistency of the PSO search improves with a higher number of agents. On the other hand, the time required for the ‘pre-tuning’ phase increases due to the increased computational burden. Table 10.7 displays a similar pattern to Table 10.6 – this indicates that for this application the best ITAE index was found using 5 agents and required 13.65 to 13.8 minutes to search.

Number of agents	Best result obtained				Standard Deviation from 4 trail runs				Average Time (Minutes)
	ITAE	K_c	T_i	T_d	ITAE	K_c	T_i	T_d	
2 agents	1483.47	2.06	3.41	0.02	1660.18	1.06	2.17	0.21	5.51
3 agents	1463.33	2.16	4.44	0.05	275.71	0.23	1.56	0.03	8.22
4 agents	1479.90	2.13	4.56	0.05	73.79	0.17	0.13	0.00	11.03
5 agents	1473.75	2.05	5.03	0.05	71.72	0.05	0.11	0.00	13.65

Table 10.6: Online PSO tuning for setpoint tracking tuning (4 trials)

Number of agents	Best result obtained				Standard Deviation from 4 trail runs				Average Time (Minutes)
	ITAE	K_c	T_i	T_d	ITAE	K_c	T_i	T_d	
2 agents	1302.78	3.25	0.65	1.70	762.28	1.06	0.36	0.67	5.51
3 agents	960.71	0.82	0.05	0.09	748.31	1.03	0.36	0.64	8.26
4 agents	659.95	0.60	0.10	0.06	510.34	0.42	0.15	0.49	11.02
5 agents	587.63	1.28	0.09	0.05	66.86	0.28	0.03	0.02	13.80

Table 10.7: Online PSO tuning for disturbance rejection tuning (4 trials)

10.4.1 Observations and analysis of results

10.4.1.1 System tuned for setpoint tracking

The PID parameters obtained for the setpoint tracking tuning is shown in Table 10.8. Figure 10.6 shows the closed-loop responses for the different tuning methods. The ZN closed-loop method was used and the PP method was tuned for setpoint tracking. With regards to Figure 10.6, the PP delivers a system with a large overshoot which oscillates around the setpoint. This is due to a lower proportional gain and a low value for the derivative time constant. The ZN and GA tuned responses also suffer from high overshoot; however the settling time is improved when compared to the PP method. The

PSO tuned system provides a stable response with little overshoot and very quick settling time. This is largely due to the high integral time constant that the PSO tuning method has specified.

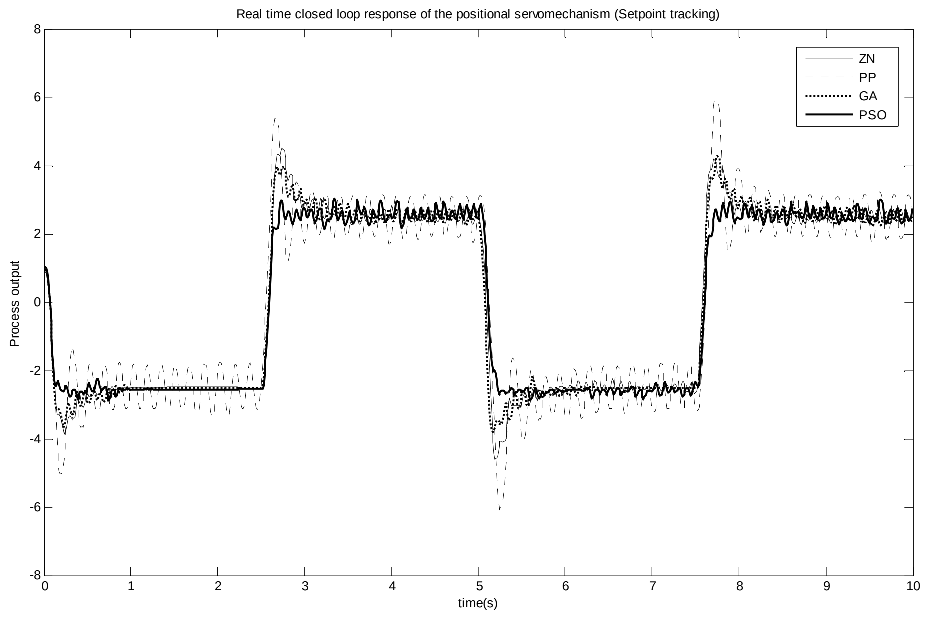


Figure 10.6: Closed-loop setpoint response of the positional servo-mechanism. (On-line tuning)

Tuning Method	**PID Parameters**			**Dynamic Performance Specifications**			**Performance Index**	
	K_c	T_i	T_d	t_r	t_s	$M_p(\%)$	$ITAE$	$ITAE_{PSO} / ITAE_{ZN,PP,AH}$
ZN	0.98	0.205	0.05	0.04	0.5	90.7	1721.6	0.86
PP	0.55	0.48	0.01	0.04	10	121.4	3679.1	0.40
GA	1.15	0.3	0.04	0.03	0.98	70.5	1653.6	0.89
PSO	2.16	4.44	0.05	0.03	0.2	10.9	1484.8	-

Table 10.8: PID settings and closed-loop response specifications for setpoint tracking only

10.4.1.2 System tuned for disturbance rejection

The PID parameters obtained for the disturbance rejection tuning is shown in Table 10.9. Figure 10.7 shows the closed-loop responses for the different tuning methods. The PP tuned system results in large overshoot thereby confirming the results from the previous section in that it may not be suitable for tuning disturbance rejection loops of this type. In contrast, the PSO tuned system delivered an improved response to load disturbance, with improved settling times and percentage overshoot.

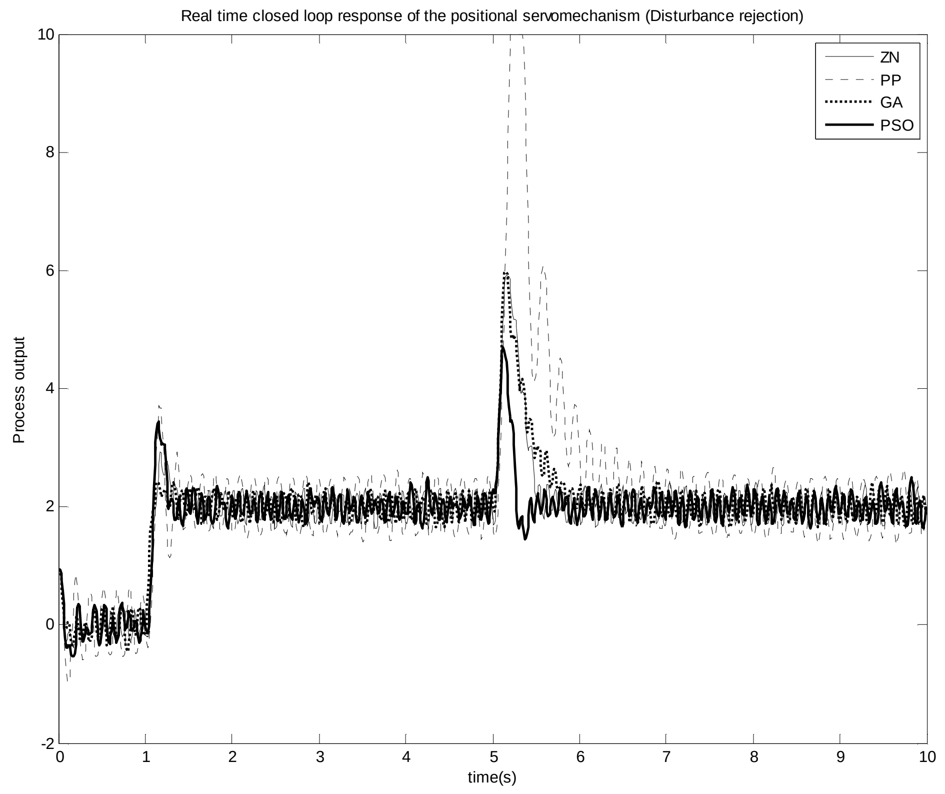


Figure 10.7: Closed-loop disturbance rejection response of the positional servo-mechanism. (On-line tuning)

Tuning Method	PID Parameters			Dynamic Performance Specifications			Performance Index	
	K_c	T_i	T_d	t_r	t_s	$M_p(\%)$	$ITAE$	$ITAE_{PSO} / ITAE_{ZN,PP,AH}$
ZN	0.98	0.205	0.05	0.03	0.87	191	1362.6	0.75
PP	0.55	0.41	0.01	0.03	10	387.4	3613.5	0.28
GA	0.97	0.38	0.06	0.04	1.21	225	1438.9	0.71
PSO	1.28	0.1	0.09	0.03	0.42	135.1	1017.1	-

Table 10.9: PID settings and closed-loop response specifications disturbance rejection and setpoint tracking

With regards to the results presented in this section, the PSO method yields systems with superior closed-loop response for servo control and disturbance rejection in comparison to conventional approaches.

10.5 Summary and conclusion

This chapter has presented the *real-time* responses of a positional servo system that was tuned using *off-line* and then *on-line tuning*. For off-line tuning a process model was obtained using the MATLAB® model identification toolbox. This model was then used in the simulation studies to determine the tuning parameters for the different tuning methods under consideration. For on-line tuning the parameters for the GA and PSO approach were obtained directly off the live plant. For both tests it was found that the PSO method outperformed all other tuning methods.

Chapter 11

Summary of Study, Recommendations and Conclusion

11.1 Introduction

The study has focused on the application of the PSO computational algorithm for PID control loops. The objective has been to improve the performance of systems that experience a poor control behavior when tuned using conventional tuning methodologies. Process control models commonly found in plant process control systems was selected for the study in order to test the efficacy of the PSO tuning methodology. Control behavior of selected plant models was measured through the system's transient response specifications to an input stimulus.

Tests were conducted using several existing conventional tuning methodologies (see ZN, 1942; CC; PP; AH; VC; DO) including the GA computational based algorithm. Each of these techniques was discussed in detail and their shortcomings in the chosen applications were also mentioned. For this study the ITAE performance index was chosen to evaluate the control performance of the process loops. The ITAE criterion penalizes large overshoots and minimizes long settling times – in our research it was heuristically found that the best control response is obtained by minimizing these two transient response criteria.

11.2 PSO Tuning

PSO tuning is implemented *offline* and then *online*. For *offline tuning*, the plant model (first order system for pressure, second order system for flow, second order integrating system for level and positional control) is determined using the MATLAB® system identification toolbox, and tuning was then performed under simulated conditions within the MATLAB® Simulink environment. O*ffline* tuning can be applied to tune a range of known process models.

For *online tuning*, all testing was conducted in real-time on a servo-positioning system. Using the PSO method for online tuning realizes the following advantages:

i) The presence of a process model is not required for determining the controller’s parameters,

ii) A minimal knowledge of the process under consideration is necessary when calculating the controller’s tuning parameters,

iii) No tuning formulae are used for determining the magnitudes of the controller’s adjustable parameters.

However, the shortcoming of online tuning is that the tuning method is not applicable to processes having long time constants - this is so because processes of this nature usually require a considerable time for the *pre-tuning* phase to execute. Pre-tuning could also lead to an unstable control action for processes experiencing the strong negative effects of nonlinearities within the control channel.

11.3 Advantages of the PSO

11.3.1 Improved Process Behaviour

Our study has also shown that processes tuned using the PSO methodology is characterized by an improved control behavior for setpoint tracking and disturbance rejection. This is evident from the improved ITAE performance index when compared to the control performance obtained with using the traditional tuning methods and the GA.

The reasons for this can be attributed to:

i) PSO relies on a memory based progression, in which the previous solutions are *remembered* and is continually improved upon until convergence is reached,

ii) GA's suffers from premature convergence since it relies on genetic operators that allow weak solutions to contribute to the composition of future candidate solutions,

iii) Traditional tuning methods require further fine tuning to improve control performance.

11.3.2 Attractive features of PSO Based PID Tuning

In this study the PSO algorithm was used as an alternative to finding suitable tuning parameters for a variety of processes. The PSO has several attractive features that make it an ideal candidate for the tuning of PID controllers, namely:

i) *Fast convergence*: The PSO is influenced by the simulation of social behaviour rather than the survival of the fittest as in the GA. From the tests discussed in Chapter 8, it was shown that each individual benefits from its history and its interactions with other agents within the population. This sharing of knowledge helps facilitates faster convergence to an optimal solution.

ii) S*imple operating algorithm:* The use of simple mathematical operators facilitates a faster computational time and makes the algorithm suitable for determining tuning parameters under high-speed dynamical conditions for processes that lend themselves to tuning of this nature, such as flow and pressure control.

iii) *Efficient operating algorithm*: From the tests that were conducted, it was shown that the PSO determined parameters provide the yielded the best control performance – this is evident from the low ITAE that was observed during the tests.

iv) *Repeatability:* The PSO was compared to the GA evolutionary algorithm. Tuning parameters obtained with the PSO are consistent over a number of tuning sessions. This does not apply to the GA based tuning method.

11.3.3 Fixed PSO Operating Parameters for Improved Repeatability

The study has also presented experiments to analyze the effects of variations in PSO parameters for different process models. Observations of the results revealed that a fixed set of PSO parameters, namely a constriction factor of $\chi = 0.73$, cognitive acceleration

c_1=2.05, social acceleration c_2=2.05, velocity v_{max} = 1 and a swarm size of 20 agents produces repeatable results for all the processes considered for this study. Large magnitudes of c_1 and c_2 made the constriction factor $\chi < 0.73$ and damped particle movement within the search space (see Clerc, 1999).

11.4 Recommendations for further research

The study can be extended by using the ZN tuning in conjunction with the PSO tuning methodology for refining the ZN tuning parameters. In this approach the PSO tuning utilizes initial tuning parameters given by the ZN tuning method as a starting point to begin search. The effects of using hybrid PSO-GA optimization strategies may be considered. It may be advantageous to *kill* poor performing PSO particles in order to improve efficiency and search capability.

11.5 Summary and conclusion

Research was conducted to study the effects of using the PSO algorithm as a tool for PID tuning. From the results presented in the study it was shown that the PSO tuning yielded improved responses and can be applied to different process models encountered in the process control industry.

References

[1] Anandanatarajan R., Chidambaram M. and Jayasingh T., “Limitations of a PI controller for a first-order nonlinear process with dead time”, *ISA Transactions,* Vol. 45, pp. 185-199, 2006

[2] Åström K.J., “Automatic Tuning of PID Controller”, Instrument Society of America, Research Triangle Park, 1995

[3] Åström K.J. and Hägglund T., “Automatic tuning of simple regulators with specification on phase and amplitude margins”, *Automatica*, Vol. 20, pp. 645-651, 1984

[4] Åström K., and Hägglund T., “PID controllers: Theory, Design and Tuning”, ISA, Research Triangle Park, NC, 1995

[5] Åström K., and Hägglund T., “Revisiting the Ziegler-Nichols Step Response method for PID control ”, *Journal of Process Control,* Vol. 14, pp. 635-650, 2004

[6] Bonabeau E., Dorigo M. and Theraulaz T., “From Natural to Artificial Swarm Intelligence”, Oxford University Press, 1999

[7] Boyd R. and Recharson P., “Culture and the Evolutionary Process”, University of Chicago Press, 1985

[8] Brown M., “The state of PID control in South Africa”, *S.A Instrumentation and Control*, pp. 131-136, 1994

[9] Carlisle A. and Dozier G., “An Off-The-Shelf PSO”, *In Proceedings of the PSO Workshop, pp 1-6*, 2001

[10] Cohen G.H. and Coon G.A., “Theoretical consideration of retarded control”, *Trans. ASME,* Vol. 75, pp. 827-834, 1953

[11] Cooper J. D., “Practical Process Control Using Control Station 3.7”, *Control Station LLC*, 2004

[12] Clerc M., “The Swarm and the queen: towards a deterministic and adaptive particle swarm optimization”, *Proceedings of the Conference on Evolutionary Computation*, pp. 1951-1957, 1999

[13] Dorigo M., Gambardella L.M., “Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem”, *IEEE Transactions on Evolutionary Computation*, pp.53-66, 1997

[14] Eberhart R.C. and Shi Y., “Comparing Inertia Weights and Constriction Factors in Particle Swarm Optimization”, *Proceedings of the Congress on Evolutionary Computation*, pp 84-88, 2000

[15] Eberhart R.C. and Shi Y., “Comparison between genetic algorithms and particle swarm optimization”, *IEEE Int. Conf. Evol. Comput., Anchorage,* pp 611-616, 1998

[16] Engelbrecht A.P., “Computational Intelligence”, John Wiley and Sons, 2002

[17] Fukuyama Y., Naka S., Genji T. and Yura T., “A Particle Swarm Optimization for Reactive Power and Voltage Control Considering Voltage Security Assessment”, *IEEE Transactions on Power Systems*, Vol. 15, No. 4, 2000

[18] Gaing Z.L., “A Particle Swarm Optimization approach for Optimum Design of PID controller in AVR System”, *IEEE Transactions On Energy Conversion*, Vol. 19, No.2, pp. 284-291, 2004

[19] Goldberg D.E., “GA in search, optimization and machine learning”, Addison-Wesley, 1989

[20] Govender P., “Design of a non-linear analog PID controller”, Masters Thesis – Department of Electrical Engineering (Light Current) at Technikon Natal, 1997

[21] Govender P., “Nonlinear Predictors in PID control: Controlling Processes having Dead-Time”, *PhD Thesis – Department of Electrical Engineering (Light Current) at Durban Institute of Technology*, 2003

[22] Hassan R., Cohanim B., de Weck O. and Venter G., “A Comparison of Particle Swarm Optimization and the Genetic Algorithm”, *Structural Dynamics & Materials Conference, American Institute of Aeronautics and Astronautics*, pp. 1-13, 2005

[23] Hägglund T., “A Predictive PI Controller for Processes with Long Dead Times”, *IEEE Control Systems*, pp. 57-60, 1992

[24] Hang C.C., Åström K. and Ho W.K., “Refinements of the Ziegler-Nichols tuning formula”, *IEE Proceedings-D*, Vol. 138, No. 2, pp. 111-118, 1991

[25] Haung H.P. and Chen C.C., “Control-system synthesis for open-loop unstable process with time delay”, *IEE Proc.-Control Theory Appl.*, Vol. 144, No. 4, pp. 334-345, 1996

[26] Kennedy J. and Eberhart R., “Particle swarm optimization”, *Proc. IEEE Int. Conf. Neural Networks*, Vol. 4, Perth, Australia, pp. 1942-1948, 1995

[27] Kennedy J., Russell R.C. and Shi Y., “Swarm Intelligence”, The Morgan Kaufmann Series in Evolutionary Computation, 2001

[28] Krohling R. A. and Rey J. P., “Design of optimal disturbance rejection PID controllers using genetic algorithm”, *IEEE Trans. Evol. Computation*, Vol. 5, pp. 78-82, 2001

[29] Lee Y., Lee J. and Park S., “PID controller tuning for integrating and unstable processes with time delay”, *Chemical Engineering Science,* Vol. 55, pp. 3481-3493, 2000

[30] Lipták, “Process Control”, Chilton Book Company, 1995

[31] Liu G.P. and Daley S., “Optimal-tuning PID control for industrial systems”, *Control Engineering Practice,* Vol. 9, pp. 1185-1194, 2001

[32] Majhi S. and Atherton D.P., “Modified Smith Predictor and controller for processes with time delay”, *IEE Proc.-Control Theory Appl.*, Vol. 146, pp. 359-366, 1999

[33] Ogata K., “Modern Control Engineering”, Englewood Cliffs, N.J., Prentice-Hall, 1970

[34] Omran M.G.M., “Particle swarm optimization methods for pattern recognition and image processing”, *PhD Thesis – Department of Computer Science at University of Pretoria,* 2005

[35] Panda C.P., Yu C.C. and Huang H.P., “PID tuning rules for SOPDT systems: Review and some new results”, *ISA Transactions,* Vol. 43, pp. 283-295, 2004

[36] Paor D.E. and O’Malley M., “Controllers of Ziegler-Nichols type for unstable process with time delay”, *Int. J. Control*, Vol. 49, pp. 1273-1284, 1989

[37] Pomerleau A. and Poulin E., "Manipulated variables based PI tuning and detection of poor settings: An industrial experience", *ISA Transactions,* Vol. 43, pp. 445-457, 2004

[38] Pomerleau A., Poulin E., Desbiens A. and Hodouin D., "Development and Evaluation of an Auto-tuning and Adaptive PID Controller", *Automatica,* Vol. 32, pp. 71-82, 1996

[39] Poulin E. and Pomerleau A., "PID tuning for integrating and unstable processes", *IEE Proc.-Control Theory Appl.*, Vol. 143, pp. 429-435, 1996

[40] Pillay N. and Govender P., "A Particle Swarm Optimization Approach for Model Independent Tuning of PID Control Loop", *IEEE Africon 2007, IEEE Catalog*: 04CH37590C, *ISBN*: 0-7803-8606-X, 2007

[41] Reynolds C., "Flocks, Herds and Schools: A Distributed Behavioral Model, *Computer Graphics,* Vol.21, No. 4, pp. 25-34, 1987

[42] Srinivas D., "Autotuning of PID controllers", Masters Thesis – IDP in Systems and Control Engineering at the India Institute of Technology, Mumbai, 2006

[43] Salerno J., "Using the Particle Swarm Optimization Technique to Train a Recurrent Neural Model", *Proc. Of the 9th International Conference on Tools with Artificial Intelligence (ICTAI'97),* 1997

[44] Shi Y. and Eberhart R.C., “A modified particle swarm optimizer”, *Proceedings of the IEEE International Conference on Evolutionary Computation,* pp 69-73, 1998

[45] Shinskey F.G., “Process Control – Where have we been, where are we going? ”, *Elektron*, pp. 7-10, 1994

[46] Van Der Bergh F., “An analysis of Particle Swarm Optimizers”, PhD Thesis – Faculty of Natural and Agricultural Science at the University of Pretoria, 2001

[47] Veeramachaneni K., Peram T., Mohan C. and Osadciw L., “Optimization Using Particle Swarm with Near Neighbor Interactions”, *Lecture Notes Computer Science,* Springer Verlag, 2003

[48] Venkatashankar V. and Chidambaram M., “Design of P and PI controllers for unstable process with time delay”, *Int. J. Control*, Vol. 60, pp. 1367-144, 1994

[49] Walgama K.S., Ronnback S. and Sternby J., “Generalizations of conditioning technique for anti-windup compensators”, *IEE Proceedings,* Vol. 135, pp. 109-117, 1992

[50] Youla D.C. and Bongiorno J.J.J. and Jabr H.A., “Modern Wiener-Hoppf design of optimal controller-part I: The single-input-single output case”, *IEEE Transactions,* Vol. 21, pp. 3-13, 1976

[51] Zhuang M. and Atherton D.P., “Automatic tuning of optimum PID controllers”, *IEE Proceedings-D*, Vol. 140, No.3, pp. 216-224, 1993

[52] Ziegler J.G. and Nichols N.B., “Optimum settings for automatic controllers”, *Trans. ASME*, Vol. 65, pp. 433-444, 1942

Appendix A

A1. PSO source code (MATLAB® m-file).

Appendix B

B1. Trial runs for PSO and GA tuning for Chapter 8.

B1.1. Experiment 8.1: Tuning for setpoint tracking of FOPDT process model:

$$\left(G_p(s) = \frac{\exp\,(-0.2s)}{(s+1)} \right)$$

PSO tuning method						
Trial	**ITAE**	**Kc**	**Ti**	**Td**	**Time(s)**	**Iterations**
1	3.66	3.63	0.97	0.07	20.14	32
2	3.65	3.63	0.97	0.07	19.80	31
3	3.72	3.41	0.97	0.06	20.60	33
4	3.65	3.63	0.97	0.07	24.74	43
5	3.65	3.63	0.97	0.07	21.10	37
6	3.65	3.63	0.97	0.07	20.24	35
7	3.66	3.61	0.97	0.07	17.25	30
8	3.66	3.62	0.97	0.07	18.18	31
9	3.65	3.63	0.97	0.07	20.75	36
10	3.66	3.63	0.96	0.07	18.25	31
$\bar{x}$	3.66	3.63	0.97	0.07	20.19	32.50
σ	0.02	0.07	0.00	0.00	2.07	3.98

GA tuning method						
Trial	**ITAE**	**Kc**	**Ti**	**Td**	**Time(s)**	**Iterations**
1	18.69	0.98	0.60	0.28	40.64	67
2	16.89	0.95	0.67	0.07	59.10	101
3	17.01	0.98	0.72	0.12	55.90	95
4	18.38	0.98	0.57	0.23	21.56	36
5	16.58	0.93	0.70	0.01	58.60	101
6	18.34	0.88	0.67	0.10	50.00	86
7	17.96	0.86	0.70	0.03	58.40	101
8	19.60	0.85	0.54	0.13	43.30	74
9	19.36	0.94	0.58	0.29	26.55	44
10	16.16	0.97	0.77	0.04	22.79	38
$\bar{x}$	18.15	0.94	0.67	0.11	46.65	80.00
σ	1.19	0.05	0.08	0.10	15.26	26.79

B1.2. Experiment 8.2: Tuning for setpoint tracking of SOPDT process model:

$$\left(G_p(s) = \frac{\exp\,(-0.5s)}{(s^2 + 2s + 1)}\right)$$

PSO tuning method						
Trial	**ITAE**	**Kc**	**Ti**	**Td**	**Time(s)**	**Iterations**
1	17.79	2.06	1.98	0.54	29.84	32
2	17.79	2.05	1.99	0.54	16.82	28
3	17.79	2.06	1.98	0.54	15.50	26
4	17.79	2.06	1.98	0.54	20.01	31
5	17.79	2.07	1.98	0.54	14.80	25
6	17.79	2.07	1.99	0.54	13.72	23
7	17.79	2.06	1.98	0.54	14.07	24
8	17.79	2.07	2.00	0.54	15.12	24
9	17.79	2.07	1.99	0.53	20.48	35
10	17.79	2.07	1.99	0.54	18.29	31
$\bar{x}$	17.79	2.07	1.99	0.54	16.16	27
σ	0.00	0.01	0.01	0.00	4.83	4.12

GA tuning method						
Trial	**ITAE**	**Kc**	**Ti**	**Td**	**Time(s)**	**Iterations**
1	88.31	0.86	0.91	0.89	26.34	40
2	80.06	0.94	0.95	0.92	31.80	48
3	85.11	0.83	0.97	0.90	56.90	87
4	88.69	0.79	0.95	0.93	65.40	101
5	86.99	0.87	0.92	0.92	37.86	58
6	76.71	0.97	0.98	0.83	65.84	101
7	100.90	0.52	0.97	0.43	40.00	60
8	89.66	0.88	0.89	0.93	34.53	53
9	81.60	0.87	0.98	0.95	36.42	56
10	86.50	0.93	0.92	0.73	31.02	44
$\bar{x}$	86.75	0.87	0.95	0.91	37.14	57
σ	6.57	0.13	0.03	0.16	14.58	22.92

B1.3. Experiment 8.3: Tuning for setpoint tracking of SOIPDT process model:

$$\left(G_p(s) = \frac{\exp\,(-0.2s)}{s(s+1)} \right)$$

PSO tuning method						
Trial	**ITAE**	**Kc**	**Ti**	**Td**	**Time(s)**	**Iterations**
1	12.72	5.15	2.17	0.56	22.39	40
2	12.72	5.15	2.17	0.55	16.53	29
3	12.72	5.14	2.18	0.56	21.18	37
4	12.73	5.16	2.09	0.55	14.96	26
5	12.72	5.17	2.17	0.55	17.87	31
6	12.72	5.13	2.18	0.56	19.90	35
7	12.72	5.19	2.16	0.55	17.11	30
8	12.72	5.16	2.18	0.55	15.58	27
9	12.73	5.14	2.13	0.56	22.91	40
10	12.72	5.17	2.15	0.55	15.40	27
$\bar{x}$	12.72	5.16	2.17	0.55	17.49	30.50
σ	0.00	0.02	0.03	0.01	2.99	5.39

GA tuning method						
Trial	**ITAE**	**Kc**	**Ti**	**Td**	**Time(s)**	**Iterations**
1	452.60	0.94	0.97	0.81	34.20	48
2	414.20	0.94	0.97	0.85	62.50	99
3	560.60	0.93	0.92	0.77	21.43	34
4	403.10	0.87	0.99	0.91	47.92	75
5	543.00	0.83	0.96	0.80	30.00	48
6	1142.00	0.23	0.91	0.94	23.06	37
7	578.20	0.92	0.82	0.92	63.37	103
8	396.80	0.95	0.97	0.86	62.05	101
9	875.90	0.94	0.26	0.75	21.37	34
10	1153	0.84	0.88	0.88	35.80	58
$\bar{x}$	497.80	0.90	0.94	0.86	35.00	53
σ	303.74	0.34	0.22	0.07	17.42	28.50

B1.4. Experiment 8.4: Tuning for setpoint tracking of FODUP process model:

$$\left(G_p(s) = \frac{\exp\left(-0.2s\right)}{(s-1)}\right)$$

PSO tuning method						
Trial	**ITAE**	**Kc**	**Ti**	**Td**	**Time(s)**	**Iterations**
1	16.93	3.83	1.36	0.00	19.90	32
2	16.93	3.82	1.36	0.00	14.98	24
3	16.93	3.83	1.36	0.00	13.70	22
4	16.93	3.83	1.36	0.00	16.57	27
5	16.93	3.84	1.36	0.00	13.46	22
6	16.93	3.82	1.35	0.00	24.09	40
7	16.93	3.83	1.35	0.00	16.31	26
8	16.93	3.83	1.37	0.00	13.81	22
9	16.93	3.82	1.36	0.00	13.44	22
10	16.93	3.83	1.36	0.00	14.10	23
$\bar{x}$	16.93	3.83	1.36	0.00	14.54	23.50
σ	0.00	0.01	0.01	0.00	3.48	5.87

GA tuning method						
Trail	**ITAE**	**Kc**	**Ti**	**Td**	**Time(s)**	**Iterations**
1	12190.00	0.98	0.85	0.00	27.83	30
2	11690.00	0.98	0.86	0.00	39.93	58
3	9744.00	1.02	0.98	0.00	21.79	30
4	21940.00	0.92	0.84	0.00	25.75	36
5	11370.00	0.98	0.87	0.00	23.04	33
6	24550.00	0.91	0.80	0.00	45.47	66
7	10830.00	0.99	0.85	0.00	56.97	83
8	23260.00	0.95	0.55	0.00	21.83	30
9	21210.00	0.95	0.59	0.00	22.40	31
10	12470.00	0.97	0.97	0.00	31.11	45
$\bar{x}$	12330.00	0.97	0.85	0.00	26.79	34.50
σ	5972.52	0.03	0.14	0.00	12.04	18.69

B1.5. Experiment 8.5 – Tuning for setpoint tracking and disturbance rejection of FOPDT process model: $\left(G_p(s) = \frac{\exp(-0.2s)}{(s+1)}\right)$

PSO tuning method						
Trial	**ITAE**	**Kc**	**Ti**	**Td**	**Time(s)**	**Iterations**
1	14.98	4.92	0.40	0.07	15.98	34
2	14.98	4.93	0.40	0.07	17.30	37
3	14.98	4.93	0.40	0.07	19.96	43
4	15.00	4.90	0.39	0.07	18.52	40
5	15.00	4.90	0.39	0.07	18.43	40
6	14.99	4.91	0.40	0.07	14.45	31
7	15.02	4.90	0.38	0.08	9.59	20
8	14.98	4.93	0.40	0.07	23.22	50
9	14.99	4.92	0.40	0.07	16.73	36
10	14.98	4.93	0.40	0.07	19.44	42
$\bar{x}$	14.99	4.92	0.40	0.07	17.36	37
σ	0.01	0.01	0.01	0.00	3.64	8

GA tuning method						
Trial	**ITAE**	**Kc**	**Ti**	**Td**	**Time(s)**	**Iterations**
1	83.83	1.23	0.31	0.04	63.59	110
2	88.55	0.92	0.335	0.08	52.7	101
3	85.77	0.96	0.3	0.2	38.107	75
4	83.66	1.28	0.31	0.01	52.38	104
5	90.49	0.9	0.31	0.02	31.92	63
6	89.47	0.92	0.29	0.1	33.9	66
7	82.76	1.35	0.35	0.01	51.96	101
8	93.61	0.85	0.34	0.04	43.05	84
9	82.83	1.31	0.37	0.06	34.32	68
10	89.41	0.93	0.31	0.18	44.37	88
$\bar{x}$	87.04	1.07	0.32	0.07	44.63	86
σ	83.83	1.23	0.31	0.04	63.59	110

B1.6. Experiment 8.6: Tuning for setpoint tracking and disturbance rejection of SOPDT process model: $\left(G_p(s) = \frac{\exp(-0.5s)}{(s^2+2s+1)}\right)$

PSO tuning method						
Trial	**ITAE**	**Kc**	**Ti**	**Td**	**Time(s)**	**Iterations**
1	87.03	3.39	1.26	0.56	11.37	17
2	86.51	3.33	1.31	0.54	21.68	34
3	86.48	3.30	1.33	0.53	21.97	34
4	86.49	3.31	1.33	0.53	17.85	28
5	86.47	3.30	1.33	0.53	27.20	43
6	86.47	3.29	1.34	0.53	26.58	42
7	86.47	3.29	1.34	0.53	22.16	35
8	86.47	3.29	1.34	0.53	26.84	42
9	86.49	3.29	1.34	0.52	15.14	23
10	86.47	3.29	1.34	0.53	26.30	41
$\bar{x}$	86.54	3.31	1.33	0.53	21.71	33.90
σ	0.17	0.03	0.03	0.01	5.44	8.85

GA tuning method						
Trial	**ITAE**	**Kc**	**Ti**	**Td**	**Time(s)**	**Iterations**
1	228	0.85	0.73	0.88	42.96	70
2	199	1.25	0.3	0.59	34.1	59
3	220	0.9	0.77	0.85	60.4	102
4	223	0.85	0.69	0.91	50.45	88
5	272.4	1.36	0.78	0.88	34.87	54
6	275	1.42	0.87	0.94	66.9	107
7	278	1.45	0.91	0.92	59.1	101
8	324	0.91	0.78	0.63	40.5	69
9	266.2	1.23	0.95	0.84	59.24	101
10	312	0.75	0.87	0.85	21.53	36
$\bar{x}$	259.76	1.1	0.77	0.83	47.01	78.7
σ	41.17	0.27	0.18	0.12	14.55	24.53

B1.7. Experiment 8.7: Tuning for setpoint tracking and disturbance rejection of SOIPDT process model: $\left(G_p(s) = \frac{\exp(-0.2s)}{s(s+1)}\right)$

PSO tuning method						
Trial	**ITAE**	**Kc**	**Ti**	**Td**	**Time(s)**	**Iterations**
1	52.19	7.64	0.87	0.46	24.86	45
2	52.19	7.62	0.86	0.46	28.36	51
3	52.21	7.62	0.86	0.46	16.80	30
4	52.30	7.70	0.82	0.48	24.98	45
5	52.19	7.63	0.86	0.46	21.62	39
6	52.20	7.63	0.87	0.46	17.73	32
7	52.30	7.71	0.81	0.48	19.54	35
8	52.19	7.63	0.86	0.46	25.48	46
9	52.19	7.63	0.86	0.46	28.09	51
10	52.19	7.64	0.86	0.46	19.43	35
$\bar{x}$	52.22	7.64	0.85	0.47	22.69	41
σ	0.05	0.03	0.02	0.01	4.22	8

GA tuning method						
Trial	**ITAE**	**Kc**	**Ti**	**Td**	**Time(s)**	**Iterations**
1	3073	0.29	0.86	0.41	29.38	53
2	1112	0.88	0.79	0.76	40.35	73
3	1868	0.84	0.82	0.62	25.17	47
4	3121	0.28	0.75	0.82	23.43	42
5	3121	0.28	0.75	0.82	23.42	42
6	1320	0.81	0.87	0.59	36.9	66
7	1437	1.23	0.81	0.88	21.93	39
8	1569	1.10	0.8	0.72	39.2	38
9	2347	0.81	0.76	0.72	25.7	46
10	3133	0.71	1.32	0.94	55.69	101
$\bar{x}$	2210.10	0.72	0.85	0.73	32.10	55
σ	843.17	0.34	0.17	0.16	10.78	20

B1.8. Experiment 8.8: Tuning for setpoint tracking and disturbance rejection of FODUP process model: $\left(G_p(s) = \frac{\exp(-0.2s)}{(s-1)}\right)$

PSO tuning method						
Trial	**ITAE**	**Kc**	**Ti**	**Td**	**Time(s)**	**Iterations**
1	80.92	4.37	1.24	0	15.70	27
2	80.92	4.41	1.24	0	10.90	18
3	80.91	4.41	1.24	0	14.63	25
4	80.89	4.42	1.23	0	20.61	36
5	80.89	4.42	1.23	0	20.31	35
6	80.93	4.35	1.23	0	10.58	18
7	80.90	4.41	1.23	0	15.63	27
8	80.90	4.42	1.23	0	17.17	30
9	80.89	4.42	1.23	0	17.91	31
10	80.90	4.42	1.23	0	12.43	21
$\bar{x}$	80.91	4.41	1.23	0.00	15.59	27
σ	0.01	0.03	0.00	0.00	3.55	6

GA tuning method						
Trial	**ITAE**	**Kc**	**Ti**	**Td**	**Time(s)**	**Iterations**
1	16205	0.86	0.8	0	23.31	37
2	16670	1.2	1.03	0	55.49	101
3	11264	0.82	0.78	0	41	77
4	15227	0.92	0.93	0	55.11	101
5	16051	0.94	0.87	0	38.63	71
6	10243	1.34	0.95	0	41.27	76
7	16367	0.93	0.89	0	30.4	60
8	17275	0.87	0.64	0	22.67	41
9	11149	0.96	0.9	0	60.6	112
10	15332	0.91	0.73	0	23.5	43
$\bar{x}$	14578.30	0.98	0.85	0.00	39.20	72
σ	2628.69	0.16	0.12	0.00	14.27	27

Appendix C

C1. Process Control Rig Loop Schematics.

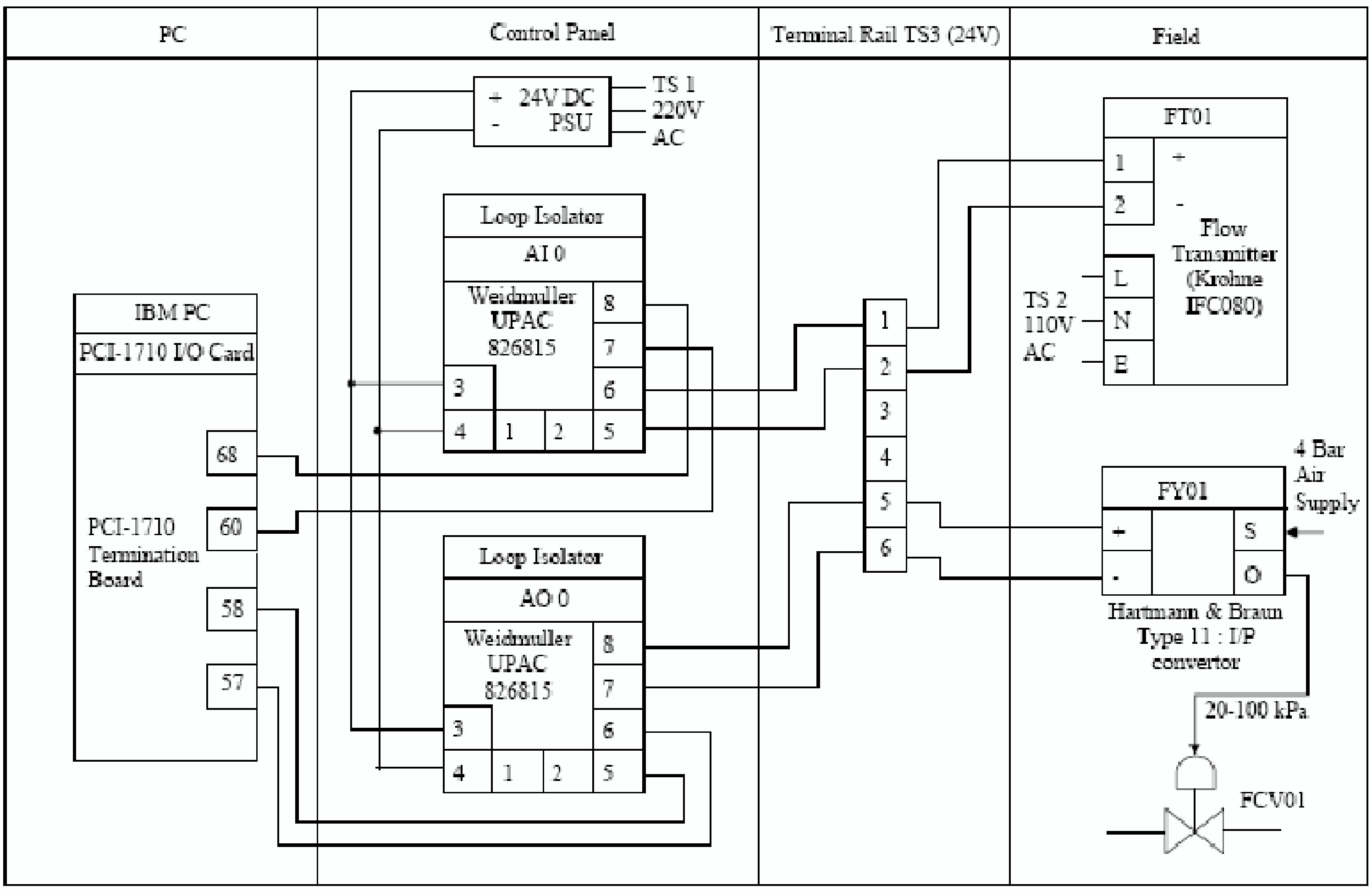


Figure C1.1: Wiring diagram of the flow control loop (FT01)

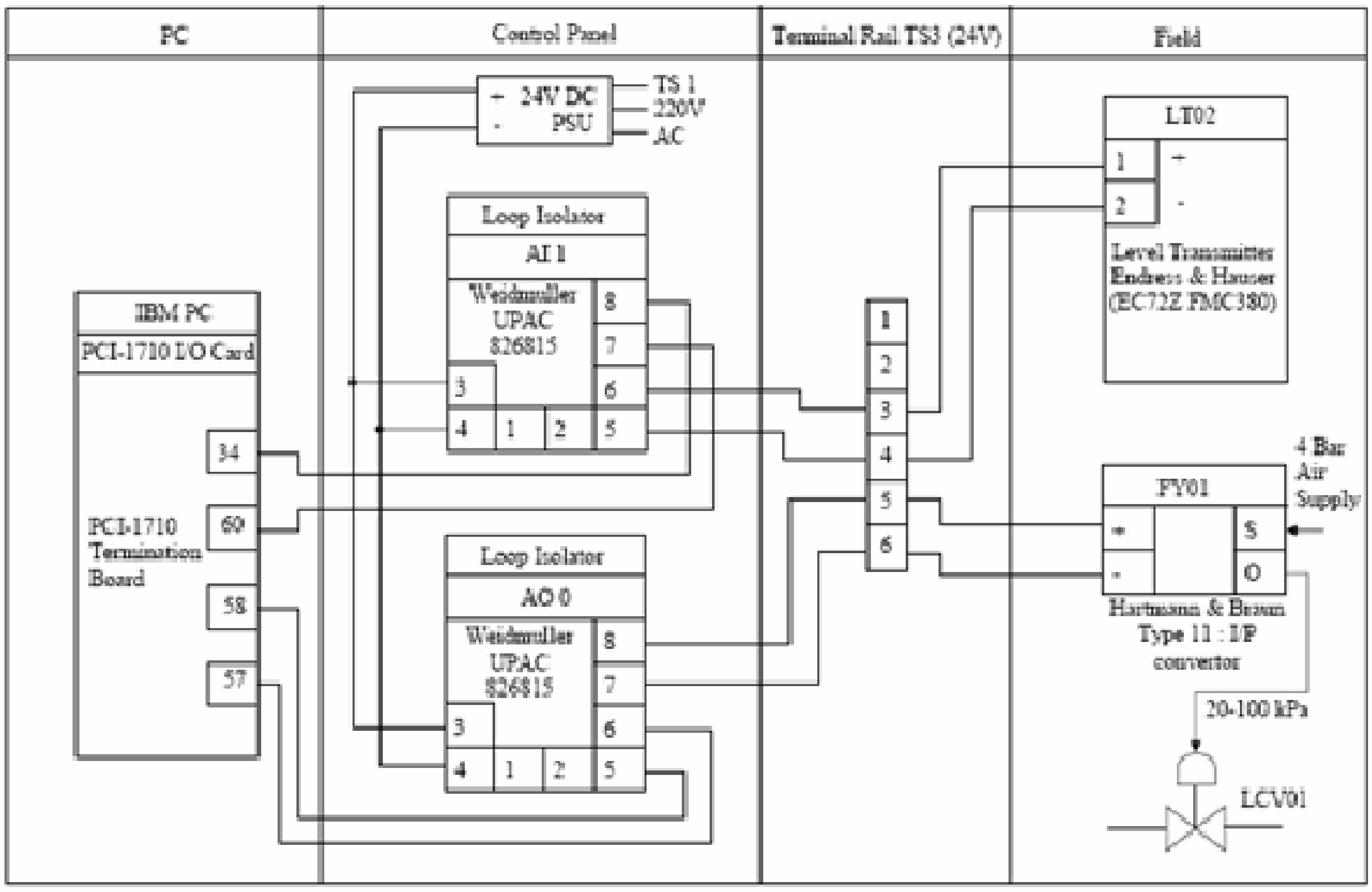


Figure C1.2: Wiring diagram of the level control loop (LT02)

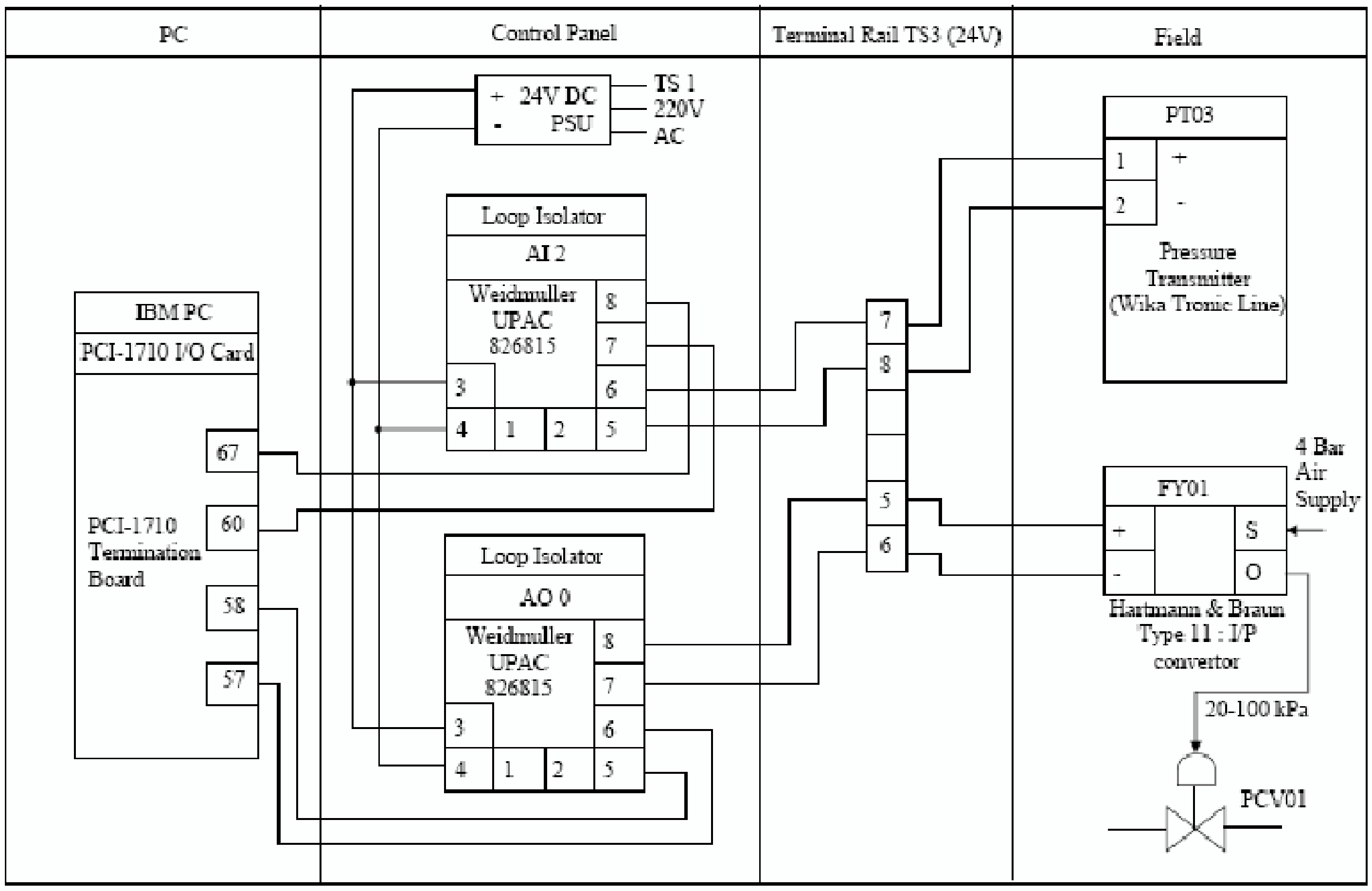


Figure C1.3: Wiring diagram of the pressure control loop (PT03)

C2. Trial runs for PSO and GA tuning for Chapter 9

C2.1 Pressure control loop: $\left(G_p(s) = \frac{0.62\exp(-0.1s)}{(0.5s+1)}\right)$

GA tuning method						
Trial	**ITAE**	**Kc**	**Ti**	**Td**	**Time(s)**	**Iterations**
1	33.00	0.93	0.02	0.00	51.27	40
2	27.97	0.93	0.02	0.00	38.79	62
3	22.07	0.81	0.01	0.00	43.65	70
4	34.30	0.94	0.07	0.00	21.58	34
5	64.40	0.77	0.19	0.00	21.60	34
6	22.18	0.67	0.01	0.00	41.60	67
7	58.40	0.77	0.06	0.00	21.80	35
8	25.37	0.94	0.02	0.00	34.37	55
9	54.30	0.88	0.06	0.00	30.10	48
10	28.57	0.84	0.02	0.00	36.14	58
$\bar{x}$	30.79	0.86	0.02	0.00	35.26	52
σ	15.85	0.09	0.05	0.00	10.27	14

PSO tuning method						
Trial	**ITAE**	**Kc**	**Ti**	**Td**	**Time(s)**	**Iterations**
1	2.41	4.53	0.42	0.00	8.47	20
2	2.41	4.59	0.43	0.00	8.46	20
3	2.41	4.61	0.43	0.00	8.07	19
4	2.4	4.49	0.42	0.00	9.32	22
5	2.4	4.52	0.43	0.00	11.35	27
6	2.4	4.50	0.42	0.00	10.49	25
7	2.41	4.57	0.43	0.00	7.74	18
8	2.4	4.49	0.42	0.00	7.30	17
9	2.4	4.51	0.43	0.00	7.34	17
10	2.41	4.61	0.44	0.00	7.72	18
$\bar{x}$	2.41	4.53	0.43	0.00	8.27	20
σ	0.01	0.05	0.01	0.00	1.36	3

C.2.2 Flow control loop: $\left(G_p(s) = \frac{0.5\exp(-6.5s)}{(1.24s^2+3.5s+1)}\right)$

GA tuning method						
Trial	**ITAE**	**Kc**	**Ti**	**Td**	**Time(s)**	**Iterations**
1	3015.00	0.04	0.30	0.00	63.53	52.00
2	2764.00	0.08	0.72	0.00	28.28	34.00
3	2763.00	0.08	0.73	0.00	26.25	34.00
4	2788.00	0.08	0.72	0.00	23.90	30.00
5	2744.00	0.09	0.75	0.00	25.36	33.00
6	3112.00	0.15	0.95	0.00	21.63	28.00
7	3042.00	0.04	0.36	0.00	21.70	28.00
8	3214.00	0.16	0.94	0.00	64.00	85.00
9	2746.00	0.10	0.82	0.00	21.60	28.00
10	2638.00	0.11	0.96	0.00	50.00	38.08
$\bar{x}$	2776.00	0.09	0.74	0.00	25.81	33.50
σ	194.48	0.04	0.23	0.00	17.49	17.69

PSO tuning method						
Trial	**ITAE**	**Kc**	**Ti**	**Td**	**Time(s)**	**Iterations**
1	1013.00	0.85	5.04	0.00	24.90	36.00
2	1013.00	0.85	5.03	0.00	23.93	35.00
3	1013.00	0.85	5.02	0.00	22.61	33.00
4	1013.00	0.85	5.03	0.00	24.57	36.00
5	1013.00	0.85	5.03	0.00	25.43	37.00
6	1013.02	0.86	5.04	0.00	18.53	27.00
7	1013.00	0.85	5.03	0.00	21.12	31.00
8	1013.00	0.85	5.03	0.00	29.01	43.00
9	1013.00	0.85	5.02	0.00	32.37	48.00
10	1013.00	0.85	5.02	0.00	27.18	40.00
$\bar{x}$	1013.00	0.85	5.03	0.00	24.74	36.00
σ	0.01	0.00	0.01	0.00	3.93	5.99

C.2.3 Level control loop $\left(G_p(s) = \frac{0.02\exp(-3s)}{s(0.76s+1)}\right)$

GA tuning method						
Trial	**ITAE**	**Kc**	**Ti**	**Td**	**Time(s)**	**Iterations**
1	7825.00	0.11	0.76	0.00	30.40	25.00
2	29100.00	0.18	0.95	0.00	22.90	29.00
3	2228.00	0.01	0.96	0.00	70.06	90.00
4	1176.00	0.06	0.50	0.00	21.40	27.00
5	45680.00	0.17	0.82	0.00	21.50	27.00
6	39470.00	0.13	0.68	0.00	21.40	27.00
7	5069.00	0.03	0.93	0.00	62.00	80.00
8	7735.00	0.05	0.99	0.00	32.88	42.00
9	6722.00	0.04	0.98	0.00	37.58	48.00
10	4898.00	0.03	0.98	0.00	24.49	31.00
$\bar{x}$	7228.50	0.06	0.94	0.00	27.45	30.00
σ	16554.76	0.06	0.16	0.00	17.63	23.65

PSO tuning method						
Trial	**ITAE**	**Kc**	**Ti**	**Td**	**Time(s)**	**Iterations**
1	6137.50	14.54	5.65	0.00	455.50	361.00
2	6137.48	14.54	5.64	0.00	454.50	361.00
3	6138.32	14.34	5.62	0.00	354.50	283.00
4	6137.49	14.54	5.64	0.00	454.50	361.00
5	6478.20	14.41	5.65	0.00	399.39	314.00
6	6136.29	14.54	5.64	0.00	453.50	362.00
7	6138.22	14.34	5.63	0.00	352.50	282.00
8	6136.49	14.55	5.64	0.00	454.50	361.00
9	6137.49	14.54	5.64	0.00	456.50	363.00
10	6505.79	14.41	5.71	0.00	502.18	393.00
$\bar{x}$	6137.50	14.54	5.64	0.00	454.50	361.00
σ	149.65	0.09	58.09	0.00	48.78	37.57

Appendix D

D1. Derived publications.

D1.1 “A Particle Swarm Optimization Approach for Model Independent Tuning of PID Control Loop,” *IEEE Africon 2007, IEEE Catalog: 04CH37590C, ISBN: 0-7803-8606-X*, 2007

D1.2 “Particle Swarm Optimization of PID Control for Servo-System Positioning,” *Proceedings of the Tenth IASTED International Conference on Control and Applications,* pp.148-153, 2008

D1.3 *Draft Journal Paper*