II.1.Introduction

A control system is a device or set of devices used to manage, command, direct or regulate the behavior of other devices to provide desired system response. PID controllers are the most popular controllers because of their effectiveness, simplicity of implementation and broad applicability. However, PID controller tuning is considered as an obstacle towards having an efficient and stable control system, where most of the PID controllers in practice are tuned by traditional techniques or by manual tuning which are difficult and time consuming.

In This chapter presents a brief theoretical introduction to the PID controllers. Furthermore an overview of the most common model for the glucose insulin dynamic are presented, focusing on the control-relevant models which are the type used for the controller implementation. and study for the methodology and application of Swarm intelligence for the tuning of the PID controllers for compared to two other methods, first one is the traditional tuning method , and second one is the random search method. Four case studies are included to emphasize the effectiveness of tuning using swarm. Simulation results showed that the PID controller, tuned by PSO method, provides accurately the desired closed loop dynamics (overshoot, rise time, settling time, and steady state error).

II.2.The PID controller

Proportional Integral Derivative (PID) controllers are widely used in industrial practice over 60 years ago, Today PID is used in more than 90% of practical control system, ranging from consumer electronics such as cameras to industrial such as chemical process. The (PID controller) is a genetic control loop feedback mechanism widely, and attempts to correct the error between a measured process variable and a desired set point by calculating and then outputting a corrective action that can adjust the process accordingly, the PID controller helps get our output (velocity, temperature, position) in a short time, with minimal overshoot, and while little error[7].

The PID controller calculation (algorithm) involves three separate parameters: the Proportional, the Integral and Derivative values. Proportional value determines the reaction to the current error the Integral value determines the reaction based on the sum of recent errors and the Derivative value determines the reaction to the rate at which the error has been changing.

II.2.1. Proportional Control

The proportional controller output uses a 'proportion' of the system error to control the system However, this introduces an offset error into the system[8]

$$P_{out} = K_p e(t) \text{.....EqII.1.}$$

II.2.2.Integral Control

The integral controller output is proportional to the amount of time there is an error present in the system. The integral action removes the offset introduced by the proportional control but introduces a phase lag into the system[8].

$$I_{out} = K_i \int_0^t e(t) d\tau \text{.....EqII.2.}$$

II.2.3. Derivative Control

$$D_{out} = K_d \frac{d}{dt} e(t) \text{.....EqII.3.}$$

The derivative controller output is proportional to the rate of change of the error. Derivative control is used to reduce, eliminate overshoot and introduces a phase lead action that removes the phase lag introduced by the integral action [8].

II.2.4.Continuous PID Controller

The three types of control are combined together to form a PID controller with the transfer function[9].

$$K(s) = K_p + \frac{K_i}{s} + K_d s \text{.....EqII.4.}$$

II.2.5.Discrete PID Controller

The PID controller will be discretised using the Trapezoidal Difference method.

At first we need a complete model of the digital controller. We start our work by continues model and then the discrete model could be achieved with some manipulations.

As we know PID controller consists of proportional, derivative and integral coefficients. The controller could be shown in time domain by **EqII.5**[8].

$$u(t) = K_p e(t) + K_i \int e(t).dt + K_d \frac{de(t)}{dt} \text{.....EqII.5.}$$

k_p is the proportional coefficient and e(t) shows controller input signal. T_i and T_d are integral and derivative coefficients respectively. In Laplace domain we can write **Eq.II.5** as follows:

$$H(s) = K_p \left[1 + \frac{1}{s * T_i} + s * T_d \right] \text{.....EqII.6.}$$

Now in order to have discrete form we must transfer **Eq.II.4** to Z domain. We can determine a digital implementation of this controller by using a discrete approximation for the derivative and integration .Impulse invariance or The trapezoidal difference method or bilinear methods could be used to achieve discrete form. Here we use The trapezoidal difference method[8].

The substitution $s = \frac{2}{T} \frac{(Z-1)}{(Z+1)}$ is used to produce a mapping as shown in **Figure II.1.**

The trapezoidal method is implemented in Matlab using the 'tustin' operator in conjunction with the C2D (Continuous to Discrete) command:

```
Discrete_PID = c2d (Continuous_PID, 0.01,'tustin');[8].
```

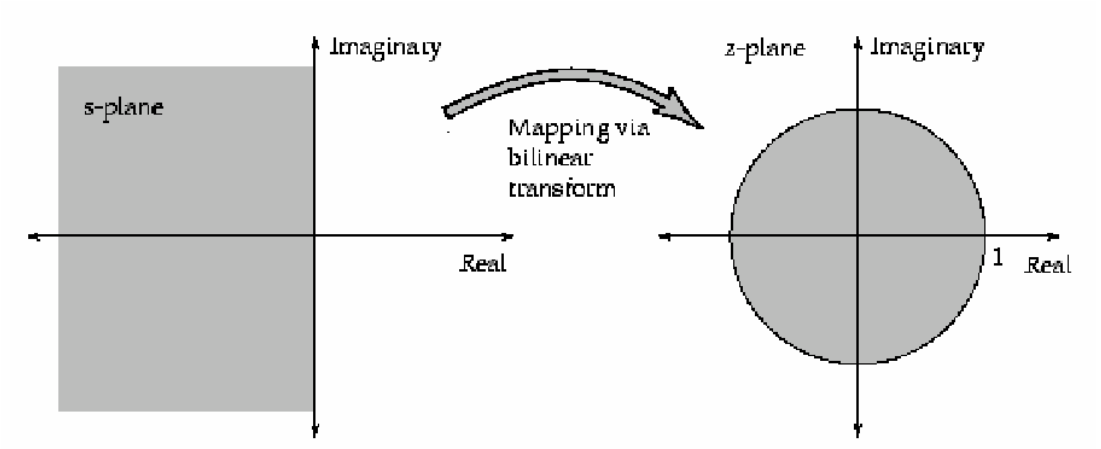


Figure II.1: Mapping using Trapezoidal Difference Method.

II.3.PID Controller Tuning Methods

The goal of tuning the PID controller is to determine parameters that meet closed loop system performance specifications, and to improve the robust performance of the control loop over a wide range of operating conditions. 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 set point change. A number of tuning techniques that take into consideration the nature of the dynamics present within a process control loop have been proposed, All these methods are based upon the dynamical behavior of the system under either open-loop or closed-loop conditions.

II.4.Ziegler-Nichols Rules for Tuning PID Controllers.

Ziegler and Nichols proposed rules for determining values of the proportional gain K_p , integral time Ti and derivative time T_d based on the transient response characteristics of a given plant. Such determination of the parameters of PID controllers or tuning of PID controllers can be made by engineers on-site by experiments on the plant. (Numerous tuning rules for PID controllers have been proposed since the Ziegler-Nichols proposal. They are available in the literature and from the manufacturers of such controllers.)[10].

II.5.Tuning of PID Controller using Ziegler Nichols Method

The first method of Z-N tuning is based on the open-loop step response of the system. The open loop system's S shaped response is characterized by the parameters, namely the process time constant T and L. These parameters are used to determine the controller's tuning parameters. The second method of Z-N tuning is closed-loop tuning method that 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 (Astrom and Hagglund, 1995). This can be achieved by adjusting the controller gain (Ku) till the system undergoes sustained oscillations (at the ultimate gain or critical gain), whilst maintaining the integral time constant (Ti) at infinity and the derivative time constant (Td) at zero. In This work uses the second method as shown in **Table II.1.[**10].

Table II.1: Ziegler-Nichols open-loop tuning rule

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

II.6.Problem Formulation

A variety of PID control strategies have been developed for diabetes control Most of the related evaluations have been based on simulation studies but experimental applications to dogs or humans have also been reported.

The PID controller calculates the insulin infusion rate that is released by the pump into subcutaneous tissues. PID controller consists of Proportional, Integral and Derivative gains. The feedback control system is illustrated in **Figure.II.2.**

II.7.A Case Study

The value of the parameters for a diabetic patient is shown below:

$P_1 = 0, P_2 = 0.81/100$,$P_3 = 4.01/1000000$, $I_0 = 192$, $G_0 = 337$, $G_b = 99$ $I_b = 8\gamma = 2.4/1000$ h=93, n= 0.23, a= 2 [12].

II.8.ControllerDesign

We start with a linearized state space model for our systems.

The general form of the state space is defined in the following equation:

$$\dot{x} = Ax + Bu$$

$$y = Cx + Du$$

The proposed mathematical model at the equilibrium point (x_0,u_0) can be written in the state space form as shown below:

$$\dot{x} = \begin{bmatrix} -p_1 - x_{20} & -x_{10} & 0 & 0 \\ 0 & -p_2 & p_3 & 0 \\ \gamma t & 0 & -n & 1 \\ 0 & 0 & 0 & -\frac{1}{a} \end{bmatrix} x + \begin{bmatrix} 0 \\ 0 \\ 0 \\ \frac{1}{a} \end{bmatrix} u \quad \text{.....EqII.7.}$$

$$y = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix} x \quad \text{.....EqII.8.}$$

To calculate the parameters using the following code Matlab

```
p1=0;
p2=0.81/100;
p3=4.01e-6;
i0=192;
G0=337;
gamma=2.4e-3;
h=93;
n=0.23;
alpha=2;
Gb=99;
Ib=8;
%%%%%%%%%%%%%%%%%%%%%%%%
t=20;
```

```
a=-p3*gamma*t/(p2*n);
b=-p1+(p3*gamma*t*h)/(p2*n)+p3*Ib/p2;
c=p1*Gb;
x10=(-b-sqrt(b.^2-4*a*c))/(2*a);
x20=p3*gamma*t*x10/(p2*n)-p3*gamma*t*h/(p2*n)-p3*Ib/p2;
 A=[-p1-x20 -x10 0 0;0 -p2 p3 0;gamma*t 0 -n 1;0 0 0 -1/alpha];
B=[0 0 0 1/alpha]';
%B=[0 0 0 1/a]';
 C=[1 0 0 0];
D=0;
```

II.9.Design of PID Controllers for Diabetic Patient

When designing a controller, the designer must define the specifications that need to be achieved by the controller. Normally the maximum overshoot (Mp) of the system step response should be small. Commonly arrange between 10% and 20% is acceptable. Also the settling time (ts), is an important factor. The objective here is to design a PID controller, so that the closed-loop system has the following specifications: small steady-state error for a step input, less than 10% overshoot, settling time less than 60 minutes.

Figure II.2 is shown PID controller for diabetic patient.

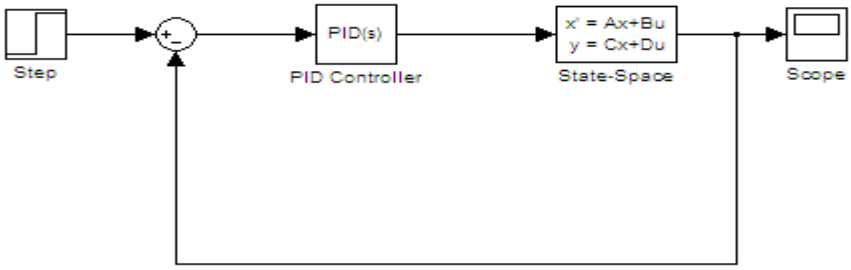


Figure II.2:PIDcontrollerfor diabetic patient

The PID controllers designed based on the models at t=20 minutes.

II.10.Optimization

Optimization, in mathematics deals with searching for the optimal solution(s) for a given problem. The optimal solution(s) must be chosen from a set of possible solutions, the so-called search space. In general, this set of possible solutions is very large. In this chapter we will focus on a particular sub set of optimization problems. We will only consider optimization problems with exactly one objective function. This is called single-objective optimization as opposed to multi-objective optimization. Furthermore, we will not consider any constraints on either input or output parameters[13]. We assume that the uncontrollable parameters are modeled within the system and are therefore not explicitly present. The search space is confined to real-valued parameters which are often limited by box-constraints (lower and upper bound on parameter values) [13].

II.10.1 Particle Swarm Optimizer

A Particle Swarm Optimizer (PSO) is a nature-inspired swarm intelligence algorithm. Swarm intelligence is a problem-solving technique that relies on interactions of simple processing units (also known in artificial intelligence as agents). The notion of a swarm implies multiple units that are capable of interacting with each other resulting in a complex behavior. The notion of intelligence suggests that this approach is successful. In the following section,[14]we will describe the general characteristics of PSOs as well as the canonical form of the algorithm and a common variant: the Fully Informed Particle Swarm.

II.10.2. History

Particle Swarm Optimization (PSO) technique based on the ability of a flock of birds or a school of fish to capitalize on their collective knowledge in finding food or avoiding predators . PSO was originally inspired by the study of bird flocking behavior by biologist Frank Heppner in the 1970s . PSO technique was invented by James Kennedy and Russell Eberhart in the mid1990s while attempting to simulate the choreographed, graceful motion of swarms of birds as part of a study investigating the notion of collective intelligence in biological populations[14].

II.10.3.The main advantages

The main advantages of PSO are as follows:

(a) -The objective function’s gradient is not required. Hence, PSO can easily deal with highly nonlinear and non-differentiable objective functions.

(b)- PSO is a population-based search algorithm. This property ensures that PSO usually does not get stuck into the so-called local optima.

(c) -PSO uses probabilistic transition rules. In this sense, PSO is a type of stochastic optimization that can support search through a complicated and highly dimensional search space. This makes PSO more flexible and robust in comparison with traditional optimization methods.

(d) -Unlike GA and other heuristic algorithms, PSO has the flexibility to maintain the balance between the global and local exploration of the search space. This exceptional attribute of PSO prevents the premature convergence problem and enhances the global search ability [15].

(e)- Unlike the traditional approaches, PSO is not sensitive to starting point such that starting anywhere in the search space, the algorithm ensures the convergence to the optimal solution.

II.10.4.Particle Swarm Optimization (PSO)-Algorithm

It uses a number of agents (particles) that constitute a swarm moving around in the search space looking for the best solution.

Each particle is treated as a point in a N-dimensional space which adjusts its "flying" according to its own flying experience as well as the flying experience of other particles.

Each particle keeps track of its coordinates in the solution space which are associated with the best solution (fitness) that has achieved so far by that particle. This value is called personal best ,pbest.

Another best value that is tracked by the PSO is the best value obtained so far by any particle in the neighborhood of that particle. This value is called gbest[16].

The basic concept of PSO lies in accelerating each particle toward its pbest and the gbest locations, with a random weighted acceleration at each time step as shown in **Figure.II.4.**

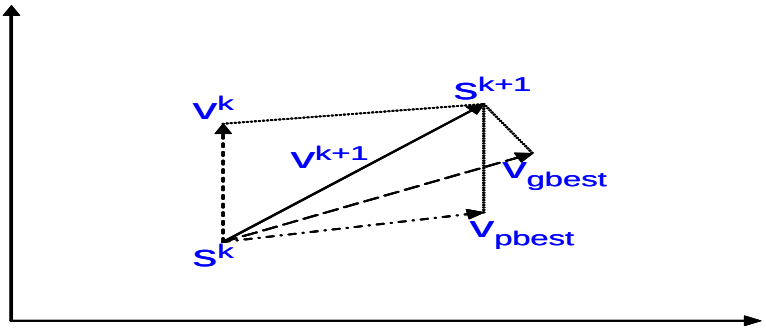


Figure.II.3:Concept of modification of a searching point by PSO

s^k: current searching point. s^{k+1}: modified searching point.

v^k: current velocity. v^{k+1}: modified velocity.

v_{pbest} : velocity based on pbest. v_{gbest} : velocity based on gbest

Each particle tries to modify its position using the following information:

the current positions, the current velocities, the distance between the current position and pbest,

the distance between the current position and the gbest.

The modification of the particle's position can be mathematically modeled according the following equation :

$$V_i^{k+1} = WV_i^k + C_1 rand_1(...)X\ (P_{best_i} - S_i^k) + C_2 rand_2(....)X\ (g_{best} - S_i^k) ...\text{EqII.9.}$$

Where:

v_i^k : velocity of agent i at iteration k.

w: weighting function.

c_j : weighting factor.

rand : uniformly distributed random number between 0 and 1

s_i^k : current position of agent i at iteration k.

$pbest_i$:pbest of agent i.gbest: gbest of the group.

The following weighting function is usually utilized in **EqII.9**:

$$W = W_{Max} - [(W_{Max} - W_{Min})Xiter]/Maxiter .....\text{EqII.10.}$$

where:

W_{Max} = initial weight.

W_{Min} = final weight.

maxiter = maximum iteration number.

iter = current iteration number.

$$S_i^{k+1} = S_i^k + V_i^{k+1} .....\text{EqII.11.}$$

II.10.5.Comments on the Inertial weight factor

A large inertia weight (w) facilitates a global search while a small inertia weight facilitates a local search. By linearly decreasing the inertia weight from a relatively large value to a small value through the course of the PSO run gives the best PSO performance compared with fixed inertia weight settings.

Larger w ---------- greater global search ability

Smaller w ---------- greater local search ability[16].

II.10.6.General basic algorithm flowchart

The PSO algorithm consists of just three steps, which are repeated until some stopping condition is met

1. Evaluate the fitness of each particle.

2. Update individual and global best fitnesses and positions.

3. Update velocity and position of each particle**.**

The first two steps are fairly trivial. Fitness evaluation is conducted by supplying the candidate solution to the objective function. Individual and global best fitnesses and positions are updated by comparing the newly evaluated fitnesses against the previous individual and global best fitnesses, and replacing the best fitnesses and positions as necessary.
The velocity and position update step is responsible for the optimization ability of the PSO algorithm. The velocity of each particle in the swarm is updated using the **EqII.8.**
The general basic algorithm flow chart is shown in **FigureII.4**[17].

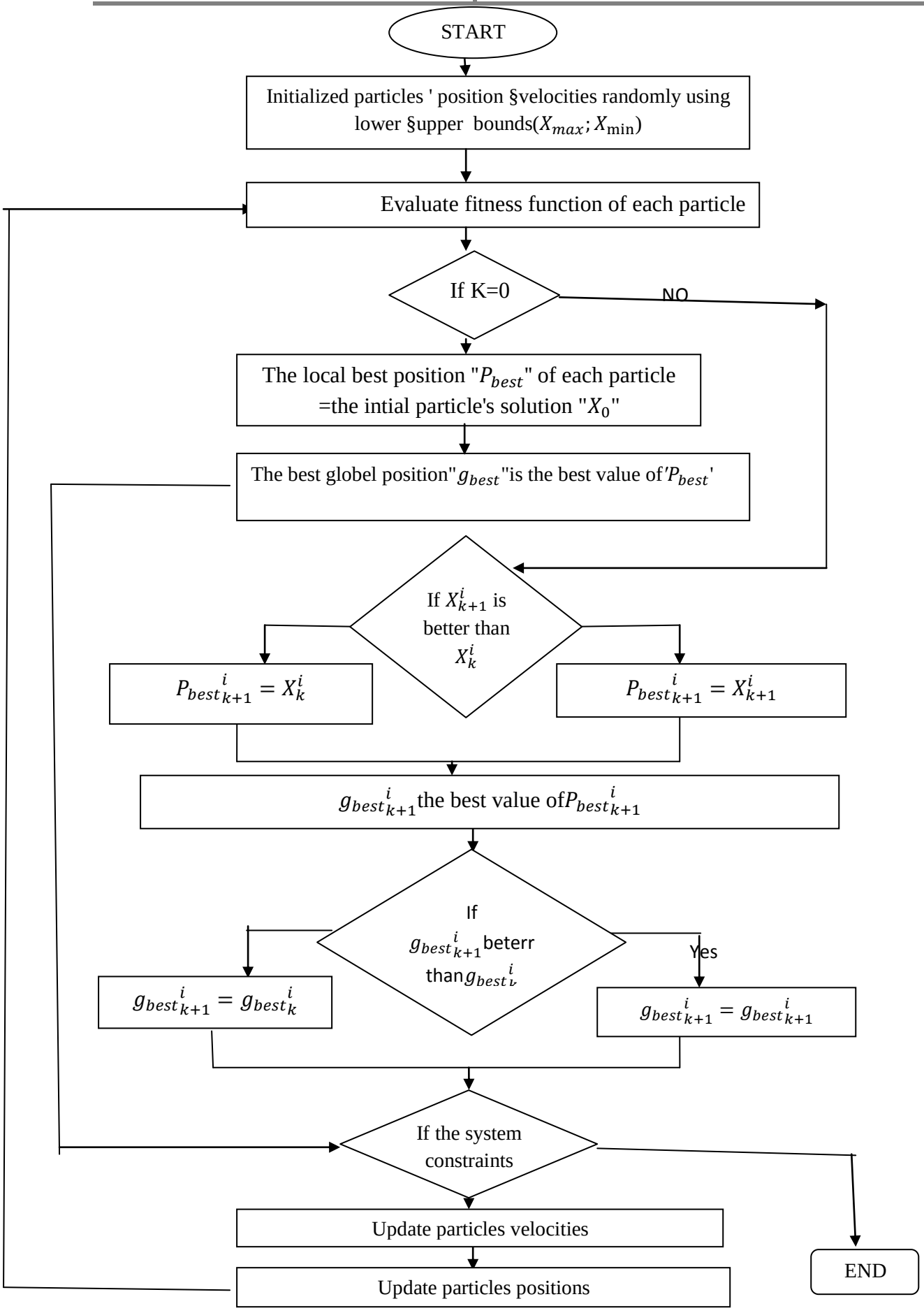


Figure II.4 The general basic algorithm flow chart of PSO

II.10.7. Design of PSO-PID controller

PID controller is optimized to achieve the optimal behavior of the plant. The optimizer is used to search for the optimal solution of the PID control gains.

II.10.8. Proposed Algorithm

In Matlab, the particle swarm optimization (PSO) generates the set of swarm particles including the proportional gain K_p, integral gain K_i , and differential $K_d$of PID controller algorithm. The PID controller gains are sent to the Simulink environment where the Bergman's model is located. The model is simulated with the PID gains sent to Simulink and the results are sent back to Matlab workspace. The three performance measures including settling time t_{stl}, rise time t_{rise} and overshoot d_{st} are evaluated from the simulation results. The fitness function value is calculated from the equation as follows:

$$f = \frac{t_{stl}}{t_{stl0}} + \frac{t_{rise}}{t_{rise0}} + \frac{d_{st}}{d_{st0}}$$

The optimization terminates if the convergence criteria is satisfied otherwise the fitness function values are sent back to SPO and sets of swarm particle are created for next loop of iteration.
In PID controller design methods the most common performance criteria are integrated absolute error (IAE), the integrated of time weight square error (ITSE), Integrated of Squared Error (ISE) and Mean Square Error (MSE).

II.10.9.PSO-PID controller

The structure diagram of PID control system based on PSO is shown in Figure.4.2.

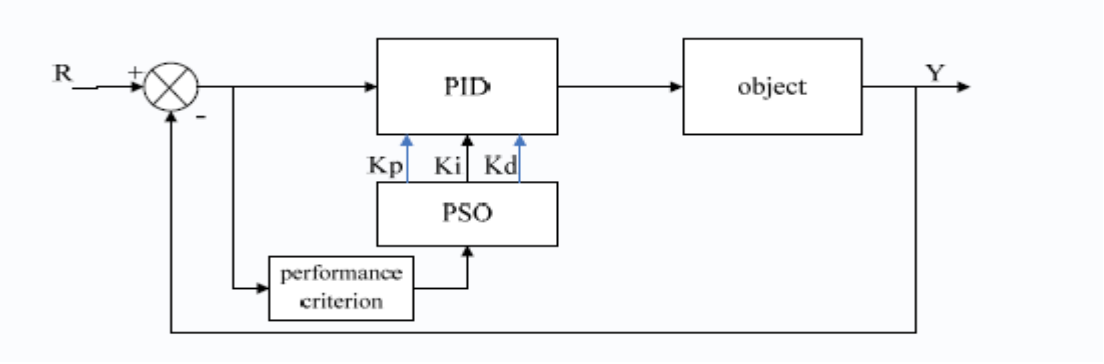
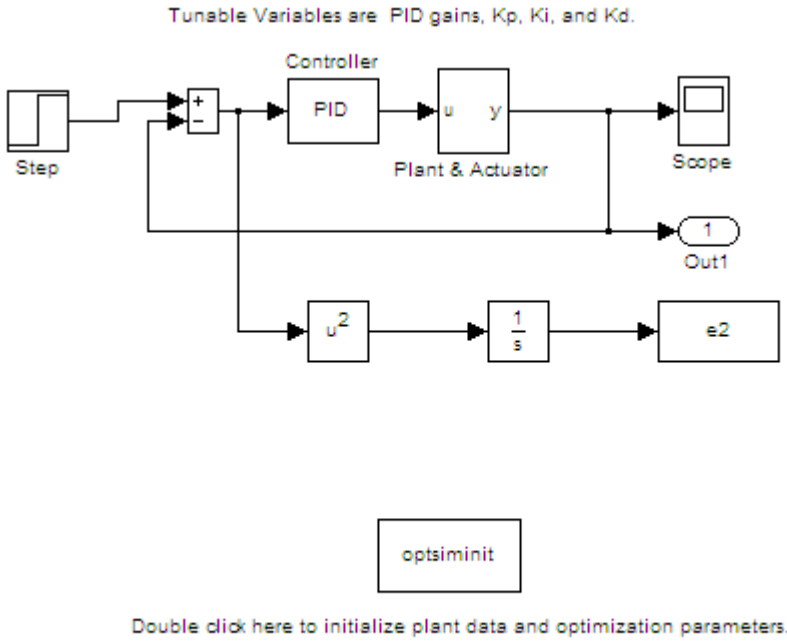


Figure II.5: Structure diagram of PID control system based on PSO

This controller is designed mainly for the following two components:

the PID Controller for the object and the module of the PSO algorithm. According to the operating state of the system, the module of PSO can optimize the parameters of the PID controller to meet the performance requirements, and the output of this module will provide the optimized parameter of PID controller.



FigureII.6:Structure diagram of PID control system based on PSO with Matlab

II.11.Conclusion

PSO is one of the new artificial intelligence branches that have many applications in several fields. Automatic control field is one of the fields that could employ the PSO. This chapter proposes a new process between the artificial intelligence which is presented in PSO and automatic control which is presented in PID controller.

So, it could be considered as an efficient method for auto-tuning process of the PID controller to solve the problems of the former tuning methods.

It could be considered as a breakthrough in the automatic control field where the features of artificial intelligence and automatic control are allied to form an optimum solution of auto-tuning of PID controller.

Sommaire