A COMPARATIVE STUDY OF ECONOMIC LOAD DISPATCH PROBLEM USING DIFFERENT INERTIA WEIGHTS OF PSO

DISSERTATION
SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE AWARD OF THE DEGREE
OF
MASTER OF TECHNOLOGY
IN
POWER SYSTEMS
(Electrical Engineering)

Submitted by:
NITIN ARORA

(2K12/PSY/13)

Under the supervision of

Prof. N K Jain & Prof. Uma Nangia

DEPARTMENT OF ELECTRICAL ENGINEERING

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

Bawana Road, Delhi-110042

2014

DEPARTMENT OF ELECTRICAL ENGINEERING

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

Bawana Road, Delhi-110042

CERTIFICATE

I, NITIN ARORA, Roll No. 2K12/PSY/13 student of M. Tech. (Power System), hereby declare that the dissertation titled — "A COMPARATIVE STUDY OF ECONOMIC LOAD DISPATCH PROBLEM USING DIFFERENT INERTIA WEIGHTS OF PSO" under the supervision of Prof. N.K. Jain & Prof. Uma Nangia of Electrical Engineering Department, Delhi Technological University in partial fulfillment of the requirement for the award of the degree of Master of Technology has not been submitted elsewhere for the award of any Degree.

Place: Delhi
Date: 29.07.2014

NITIN ARORA
2K12/PSY/13
M.Tech (PS)

Prof. N.K. Jain
Professor
EE Dept., DTU

Prof. Uma Nangia
Professor
EE Dept., DTU

ACKNOWLEDGEMENT

Any project is a team work, successful completion of which is possible only through cooperation and sincere efforts of the team mates. This is of great pleasure for me to avail the opportunity of expressing my gratefulness to all those who were of immense help to me in completing this Major Project.

Firstly I want to show my sincere and profound thanks to **Prof.N K JAIN & Prof.UMA NANGIA (project guide).** She has been a constant source of inspiration throughout the development of this Major Project and provided me with all the requisite guidance and information from time to time .I also thank Almighty for the entire blessings he has bestowed upon me. Last but not least I express my sincere thanks to my family members, colleagues, friends, college staff and all those who have contributed directly and indirectly through suggestions , thoughts, support, and presence for the completion of the Major Project ,they have remained a source of encouragement and inspiration for me.

NITIN ARORA
2K12/PSY/13
M.TECH. (PSY)

ABSTRACT

In this thesis, comparison of different inertia weights is taken into account to analyse the performance of PARTICLE SWARM OPTIMIZATION on Economic load dispatch considering the cost of the generation. Equality constraints of the problem have been considered with the inclusion of a new parameter 'k'. Comparative analysis suggests that the use of simulated annealing procedure for the variation in inertia weight in the algorithm of PSO significantly improves the performance with lesser number of iterations. The data sets being used have been generated for IEEE 5 , 14 and 30 bus system using Particle swarm optimization method. An attempt has been made to reach the target point in lesser number of iterations and with minimum cost of generation.

A MATLAB program has been developed for Particle Swarm Optimization (PSO) method to solve economic load dispatch problem considering cost of generation. All different inertia weight functions have been implemented on ECONOMIC LOAD DISPATCH problem to get the optimum value of cost of generation with lesser number of iterations.

List of Symbols & abbreviations

Cp	Acceleration coefficient for Cognitive component
C_g	Acceleration coefficient for Social component
χ	Constriction Factor
w	Inertia Weight
It_{max}	Maximum no. of iteration
nop	Number of particles
K	Penalty coefficient
ε	Tolerance Limit
r_1 & r_2	Uniformly distributed Random Numbers (0,1)
CF	Constriction Factor
ELD	Economic Load Dispatch
PSO	Particle Swarm Optimization

CHAPTER-1

INTRODUCTION

1.1 OVERVIEW

The size of our electrical power system is increasing energy demands. To accomplish this, a number of power plants are connected in parallel to supply the system load by interconnection of systems. In grid system, it is very essential to operate plant units most economically.

The economic scheduling aims to guarantee at all times the optimum combination of generators connected to the system to supply the load demand.

Economic Load Dispatch (ELD) is a very important function in the planning and operation of power system. It involves two separate steps namely the unit commitment and on-line economic dispatch considering all constraints (equality constraints and inequality constraints). The complexity of ELD depends on the factors like size of the system, generator characteristics and system constraints.

By ELD we mean to find the generations made by different generators or plants so that the total cost of the fuel is minimum. The generation in ELD is not fixed but they lie under certain limits so as to meet a particular load demand with minimum consumption of fuel and hence we can say ELD is basically the solution to a large number of load flow problems.

In this work the cost of generation is taken as the objective which is needed to be minimised. For this IEEE 5 bus, 14 bus and 30 bus systems have been considered.

Our objective is accomplished in the order as given below:

- Exploring PSO and coding the programs in MATLAB 2012a.
- Application of PSO to various benchmark functions.
- Application of PSO to Economic load dispatch problem considering cost of generation for IEEE 5 , 14 and 30 bus systems using.

1.2 AIM AND APPROACH

Our main aim in this thesis is to solve economic load dispatch (ELD) problem with minimum number of iterations considering the cost of generation for which IEEE 5, 14, & 30 bus system have been considered.

The work has been carried out in the following manner:

a. Knowing about particle Swarm Optimization and coding its algorithm in MATLAB R2012a.
b. Solution of different mathematical benchmark functions using PSO.
c. Formulation of Economic Load Dispatch (ELD) considering the cost of generation for IEEE 5, 14, & 30 Bus System.
d. Generation of non-inferior sets of IEEE 5, 14 and 30 bus systems.

Achievement of solution for IEEE 5, 14 and 30 bus system with lesser no of iterations considering the cost of generation.

1.3 LITERATURE REVIEW

We do not have a single optimization method available to solve all the optimization problems. Various optimization methods have been developed to solve many types of optimization problems recently. Latest methods of optimization (few times referred to non-traditional methods of optimization) are popular and powerful methods for solving many complicated engineering problems. The methods consists of particle swarm optimization algorithm, artificial immune systems, genetic algorithms, neural networks, ant colony optimization and fuzzy optimization.

The consumption level of the electric power is increasing exponentially in the modern world. Since the electric power network is the most complex as well as huge in size, which needs proper control and decision making algorithms to get economical operation from the available resources and their best utilization by generation, transmission and distribution utilities.

In power systems the continuous unpredictable change in load demand leads to the necessity in adjusting the power generation outputs. The scheduling of the generator

output is taken care by Economic Load Dispatch (ELD) problem. Economic load dispatch is one of the major optimization issue in power system. Its objective is to allocate the demand among committed generators in the most economical manner, while all physical & operational constraints are satisfied. Many conventional & nonconventional optimization techniques available in literature are applied to solve such problems. Conventional methods have simple mathematical model and high search speed but they are failed to solve such problem because they have the drawbacks of multiple local minimum points in the cost function. Algorithms require the characteristics to be approximated; however, such approximations are not desirable as they may lead to suboptimal operation and hence huge revenue loss over time, restrictions on the shape of the fuel-cost curves. Other methods based on artificial intelligence have been proposed to solve the economic dispatch problem, these are genetic algorithm, Tabu search, particle swarm optimization etc [1].

A concept for the optimization of nonlinear functions using particle swarm methodology is introduced first time by James Kennedy and Russell Eberhart.The evolution of several paradigms is outlined, and an implementation of one of the paradigms is discussed. Benchmark testing of the paradigm is described, and applications, including nonlinear function optimization and neural network training, are proposed. The relationships between particle swarm optimization and both artificial life and genetic algorithms are described in [2].

The optimization of nonlinear functions using particle swarm methodology is described. Implementations of two paradigms are discussed and compared [3].

In paper Wei-Bing Liu and Xian-JiaWangr presented a new particle swarm optimizer based on evolutionary game (EGPSO).We map particles' finding optimal solution in PSO algorithm to players' pursuing maximum utility by choosing strategies in evolutionary games, using replicator dynamics to model the behavior of particles. And in order to overcome premature convergence a multi-start technique was introduced. Experimental results show that EGPSO can overcome premature convergence and has great performance of convergence property over traditional PSO[4].

Tao Gong and Andrew L. Tuson_presented the working mechanism of PSO in a principled manner with formal analysis and investigates the applicability of pso on the quadratic assignment problem (qap). Particularly, the derived pso operator for qap is empirically studied against a genetic algorithm (ga)[5].

In this paper, Ying-Ping Chen and Wen-Chih Peng, gave suggestion to improve the performance of the particle swarm optimizer by incorporating the linkage concept, which is an essential mechanism in genetic algorithms,and design a new linkage identification technique called dynamic linkage discovery to address the linkage problem in real-parameter optimization problems[6].

Keisuke KAMEYAMA investigsted the dynamics of PSO research and numerous variants for improvement of performance of PSO[7].

Hardiansyah et al. suggested application of PSO for Economic load dispatch problem .The results have been demonstrated for 3 and 6 generator systems with and without consideration of losses[9].

Jaya Sharma et al. presented review of PSO application in ELD problems[10].

The intensified research on environmental safety guided to create public awareness about the emission. The passage of the clean air amendments in 1990 has forced the utilities to reduce their SO_2,CO_2,NO_X emission by 40% from 1980 levels.Therefore apart from cost, emission objective must also be taken into account .The multi-objective Environmental/ economic dispatch is having two conflicting objectives, as the minimisation of cost maximizes the pollution, leads to the necessity of trade –off analysis to define admissible dispatch policies for any demand level.There has been much research pertaining to MOEED problem. M.A. Abido has compared three multiobjective evolutionary algorithms and the same has been successfully applied to environmental/economic power dispatch problem. Strength Pareto Evolutionary Algorithm (SPEA) has better diversity characteristics and is more efficient when compared to other Muti Objective Evolutionary Algorithms (MOEAs)[11].

J. C. Bansal et al proposed a large number of variations of Inertia Weight strategy[12]. This paper studies 15 relatively recent and popular Inertia Weight strategies and

compares their performance on 05 optimization test problems.Russell Eberhart et al proposed two Paradigms of PSO namely globally oriented (GBEST), and locally oriented (LBEST) and compared for the extremely nonlinear Schaffer f6 function.The authour propose both the paradigm for training of neural network & learning of robot[13]. Ajit Abraham et al implemented Particle Swarm Optimization (PSO) and Ant Colony Optimization (ACO) algorithms on some mathematical function (Griewank function, Schwefel function, Quadric function), real world applications as travelling sales men problem & data mining[14].

Singh and Dhillon converted a multiobjetcive economic emission dispatch problem into a scalar problem. This scalar set optimization problem is then solved for many types of different set of weights pattern to generate non- inferior solution along with trade-off functions. Between conflicting objectives. The optimal solution is calculated by considering real and reactive power losses, which are calculated by performing fast decoupled load flow analysis[31].

Airashidi and El- Harwary presented a PSO algorithm as an effective tool for solving constrained multiobjetcive optimization problems like Economic Dispatch(EED). Results showed that PSO was successfully capable of capturing the shape of Pareto solution sets[32].

1.4 PLAN OF THESIS

This thesis has been arranged in six chapters. The contents of the chapters are briefly outlined as indicated below:

Chapter 1: Introduction to economic load dispatch problem and research aim of the thesis. Literature survey for the covered topics has also been shown.

Chapter 2: Introduces the Particle Swarm Optimization

Chapter 3: Discusses about the applications of PSO in various fields.

Chapter 4: Explores the concepts of Particle Swarm Optimization algorithm in MATLAB R2011b and its application on various mathematical benchmark functions. Analysis of various parameters in PSO algorithm has also been carried out.

Chapter 5: Discusses the solution of Economic Load Dispatch for IEEE 5, 14 and 30 bus systems.

Chapter 6: Conclusion and the future work directions have been discussed.

Appendix and references are at the end of the thesis.

CHAPTER-2

PARTICLE SWARM OPTIMIZATION

2.1 INTRODUCTION

The Particle Swarm Optimization (abbreviated as PSO) algorithm is a stochastic search algorithm based on population and an alternate solution to complicated optimization problem which are non-linear in nature. PSO algorithm was firstly introduced by Dr. Eberhart and Dr. Kennedy in 1995 and the basic idea of PSO was initially inspired by simulation of the social behaviour of animals such as bird flocking, fish schooling and so on. PSO can be easily implemented and is computationally inexpensive, since its memory and CPU speed requirements are low (Eberhart et al.,1996). PSO is based on the natural technique of the group communication to share individual knowledge when a group of birds or insects search food or migrate and so forth in a searching space, although all birds or insects do not know where the best position is and from the nature of the social behaviour, if any member can find out a desirable path to go, the rest of the members will follow quickly.

The particle swarm optimization (PSO) is a parallel evolutionary computation technique developed by Kennedy and Eberhart based on the social behaviour metaphor. The PSO algorithm is initialized with a population of random candidate solutions, conceptualized as particles. Each particle is assigned a randomized velocity and is iteratively moved through the problem space. It is attracted towards the location of the best fitness achieved so far by the particle itself and by the location of the best fitness achieved so far across the whole population (global version of the algorithm).

The PSO algorithm includes some tuning parameters that greatly influence the algorithm performance, often stated as the exploration–exploitation tradeoff: Exploration is the ability to test various regions in the problem space in order to locate a good optimum, hopefully the global one. Exploitation is the ability to concentrate the search around a promising candidate solution in order to locate the

optimum precisely. Despite recent research efforts, the selection of the algorithm parameters remains empirical to a large extent. A complete theoretical analysis of the algorithm has been given by Clerc and Kennedy. Based on this analysis, the authors derived a reasonable set of tuning parameters, as confirmed by. The reference contains a good deal of mathematical complexity, however, and deriving from it simple user-oriented guidelines for the parameter selection in a specific problem is not straightforward.

The present work gives some additional insight into the PSO parameter selection topic. It is established that some of the parameters add no flexibility to the algorithm and can be discarded without loss of generality. Results from the dynamic system theory are used for a relatively simple theoretical analysis of the algorithm which results in graphical guidelines for parameter selection. The user can thus take well-informed decisions according to the desired exploration–exploitation trade off: either favour exploration by a thorough sampling of the solution space for a robust location of the global optimum at the expense of a large number of objective function evaluations or, on the contrary, favour exploitation resulting in a quick convergence but to a possibly non-optimal solution. Non-surprisingly , the best choice appears to depend on the form of the objective function. The newly established parameter selection guidelines are applied to standard benchmark functions.

2.2 THE BASIC MODEL OF PSO

Particle swarm optimization (PSO) is an optimization approach in which a swarm of candidate solutions are used which are referred to particles. Particles are made to ‘‘fly’’ into a search space, with each particle getting attracted towards the global best solution found by the particle’s neighbourhood and the personal best solution found by the particle. The velocity v_i is used to modify the position x_i, of the i_{th} particle and this velocity depends on the distance of the particle from its personal best solution and from the global best solution. For the original PSO,

$$v_{ij}(t+1) = w * v_{ij}(t) + c_p * r_p * (y_{ij}(t) - x_{ij}(t)) + c_g * r_g * (\hat{y}_{ij}(t) - x_{ij}(t)) \quad (2.1)$$

$$x_{ij}(t+1) = x_{ij}(t) + v_{ij}(t+1) \quad (2.2)$$

for i = 1, . . . ,nop and j = 1, . . . ,n,

where,

$\varphi_p = c_p * r_p$ and $\varphi_g = c_g * r_g$, (2.3)

nop represents the total number of particles in the swarm,
n represents the dimensions in the given problem, i.e. the total number of parameters of the function getting optimized

c_p and c_g are acceleration coefficients,

r_p, r_g ~ U(0, 1),

$x_i(t)$ represents the position of particle i_{th} at time t,

$v_i(t)$ represents the velocity of particle i_{th} at time t,

$y_i(t)$ represents the personal best solution of particle i at time t,

$y\hat{}_{ij}(t)$ represents the best position found by the neighbourhood of particle i at time t.
From Eq. (2.1), the velocity of a particle is obtained by three factors:

$v_i(t)$, which behaves as a momentum term to avoid excessive oscillations in the direction of search.

$y_i(t)$, each particle remembers its own coordinates in the solution space which are linked with the best solution (fitness) which has been obtained so far by that particle. This value is known as personal best (**p_{best}**). In the p_{best} swarm, only a limited number of particles (neighbour count) can modify the velocity of a given particle. The swarm will converge taking more time but can locate the global optimum with a greater chance.

$y\hat{}_{ij}(t)$, another best value that is obtained by the PSO is the best value achieved so far by any of the particle in the neighbourhood of that particle. This value is known as global best (**g_{best}**).All the particles in the g_{best} swarm, are neighbours of one other; thus, the best particle's position in the swarm has been used in the social term in the velocity update equation. It is supposed that g_{best} swarms converge in lesser time, as all the particles are attracted simultaneously to the best part of the search space. However,

if the global optimum is not so near as compared to the best particle, it might be impossible for the swarm to explore different areas; this means that the swarm could be trapped in the local optima.

cp $*$ $\mathbf{r_1}$ *** (**$\mathbf{y_i(t)}$ **-** $\mathbf{x_i(t))}$, known as the cognitive component. This component shows that the distance a particle is from the best solution, $y_i(t)$, found by itself. The cognitive component shows the natural tendency of individuals to go back to environments where they obtained their best performance.

cg * $\mathbf{r_2}$ *** (** $\mathbf{y^{\wedge}_{ij}(t) - x_{ij}\ (t))}$ represents the social component. This component shows the distance of a particle from the best position found by its neighbourhood. It shows the behaviour of the individuals to follow the success of the other individuals.

In the social component, $\mathbf{y^{\wedge}_{ij}(t)}$ shows the best solution found by the neighbourhood of particle i_{th}. Neighbourhood topologies are used to constrict the information exchanged between particles.

Particles modify their positions according to the ``Psychosocial compromise'' with which an individual is comfortable and what the society reckons.

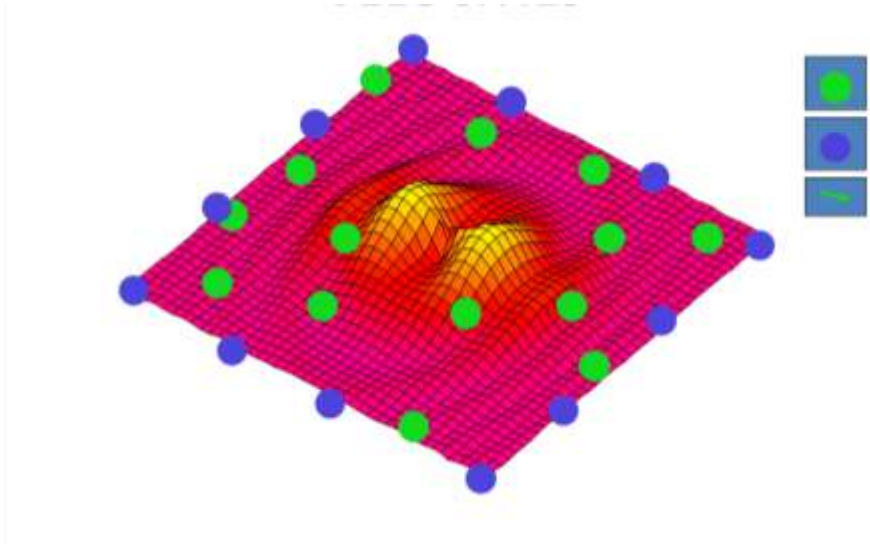


Fig 2.1 Initialization of the positions of all particles of x1 and x2

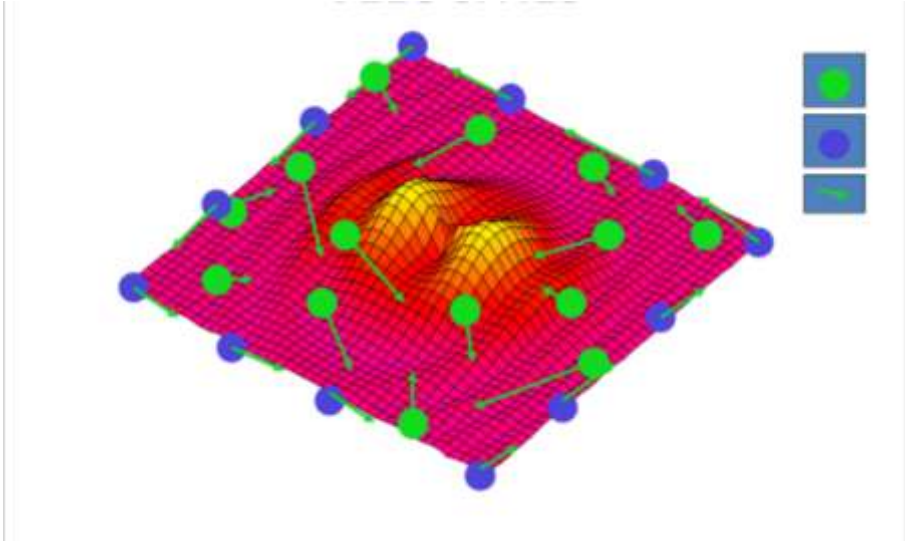


Fig 2.2 Initialization of velocities in random directions of al particles of x1 and x2

Fig 2.1 shows the random distribution of particles for two variables lets say for x1 and x2 in a given limit or boundary.whereas Fig 2.2 tells us about the random distribution of velocities for all the particles of x1 and x2.

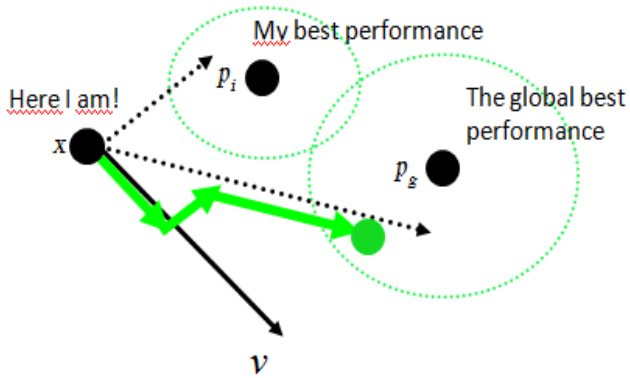


Fig 2.3 Movement of a particle on the basis of pbest and gbest

Fig2.3 shows the movement of a particle is influenced by two factors i.e. one factor is the personal best position and the second factor is related to the global best position. Both these factors contributes in deciding the new position of the different particles.

2.3 PSO ALGORITHM PARAMETERS

There are some parameters in PSO algorithm that may affect its performance. For any given optimization problem, some of these parameter's values and choices have large

impact on the efficiency of the PSO method, and other parameters have small or no effect. The different PSO parameters are number of particles or swarm size, velocity components, acceleration coefficients and number of iterations illustrated below.

2.3.1 Swarm size

Population size or swarm size is the number of particles 's' in the swarm. A large number of swarm size generates larger parts of the search space to be covered per iteration. A large number of particles may reduce the number of iterations need to obtain a better optimization result. In contrast, large amounts of particles improve the computational complexity per iteration, and more time consuming. From a large number of empirical studies, **it has been shown that most of the implementations in PSO use an interval of s ε [10,60] for the size of swarm.**

2.3.2 Velocity clamping

Initial PSO studies used cp = cg = 2.0. Although good results have been achieved, it was seen that velocities fastly exploded to large values, especially for particles at a large distance from their global best (^y) and personal best (y_i) positions. Consequently, particles have large no. of position updates, with particles leaving behind the boundaries of their search space. Velocities are clamped to control the increase in velocity.

$$v_{ij}(t+1) = v'_{ij}(t+1) \quad \text{if} \quad v'_{ij}(t+1) < V_{max} \qquad (2.4)$$

$$V_{max} \quad \text{if} \quad v'_{ij}(t+1) >= V_{max}$$

Velocity clamping does not avoid a particle from leaving the boundaries of its search space, it limits the particle step sizes, thereby divergent behaviour is restricted.

2.3.3 Iteration numbers

The number of iterations to obtain a good result depends on the problem. A very low number of iterations may halt the search process prematurely, while a very large no. of iterations have the consequence of unnecessary added computational complexity and make the convergence slow.

2.3.4 Acceleration coefficients

The acceleration coefficients $\mathbf{c_p}$ and $\mathbf{c_g}$, combined with the random values $\mathbf{r_p}$ and $\mathbf{r_g}$, maintain the stochastic influence on the cognitive and social components of the particle's velocity respectively. The constant $\mathbf{c_p}$ shows how much confidence a particle has on itself, while $\mathbf{c_g}$ expresses how much confidence a particle has on its neighbors. There are some properties of $\mathbf{c_p}$ and $\mathbf{c_g}$:

$c_p = c_g = 0$ represents all particles continue flying at their present speed until they hit the boundary of the search space. Therefore, the update equation for velocity is calculated as

$$v_{ij}(t+1) = v_{ij}(t)$$

$c_p > 0$ and $c_g = 0$ represents that all particles are independent. The velocity update equation for this condition will be

$$v_{ij}(t+1) = w * v_{ij}(t) + c_p * r_p (y_{ij}(t) - x_{ij}(t)) \quad (2.5)$$

On the contrary, $c_p > 0$ and $c_g = 0$ represents that all particles are attracted towards a single point in the entire swarm and the update in the velocity will be as under:

$$v_{ij}(t+1) = v_{ij}(t) + c_g * r_g * (\hat{y}_{ij}(t) - x_{ij}(t)) \quad (2.6)$$

$c_p = c_g$ represents that all the particles are attracted towards the average of P_{best} and G_{best} .

$c_p >> c_g$ represents that each particle is strongly influenced by its own best position, resulting in an increased wandering. In contrast, when $c_g >> c_p$ then all of the particles are much more influenced by the global best position, which causes all particles to converge prematurely to the optima.

Normally, c_p and c_g are static, with empirically finding the optimized values. Wrong initialization of c_p and c_g may result in cyclic or divergent behaviour. From the different empirical researches, this has been proposed that the two acceleration constants must be $c_p = c_g = 2$.

2.3.5 Inertia weight

Shi and Eberhart introduced the inertia weight to eliminate the need for velocity clamping and to still restrict the divergent behaviour. The momentum of the particles is controlled by the inertia weight (w) by weighing the contribution of the previous velocities–basically it is used to control how much memory of the last flight direction will affect the new velocity. The velocity equation modified to

$$v_{ij}(t+1) = w * v_{ij}(t) + cp * r_1 * (y_{ij}(t) - x_{ij}(t)) + cg * r_2 * (\hat{y}_{ij}(t) - x_{ij}(t)) \qquad (2.7)$$

Shi and Eberhart introduced the concept of inertia weight in 1999 to reduce the velocities over time (or iterations), to control the exploitation and exploration abilities of the swarm and to converge the swarm more efficiently and accurately.

If $w \geq 1$ then the velocities increase with time and particles can hardly divert their directions to return towards optimum, and the swarm diverges. If $\mathbf{w} \leq 1$ then little momentum is only saved from the initial step and quick changes to directions are set in the process. If $\mathbf{w} = 0$ particles velocity disappears and all particles move without knowledge of the last velocity in each step.

The inertia weight might be implemented either as dynamically changing values or a fixed value. Initial implementations of used a fixed value for the whole process for all particles, but now dynamically changing inertia values is used because this parameter controls the exploration and exploitation of the search space. Various strategies were suggested time to time were studied in detail.

The inertia value is usually high initially, which allows all particles to freely move in the search space in the initial steps and decreases with time. Therefore, the process shifts from the exploratory mode to the exploitative mode. This decrease in inertia weight has produced good results in most of optimization problems. To control the balance between local and global exploration and to obtain quick convergence and to reach an optimum, the inertia weight whose value decreases linearly with the increase in iteration number is set accordingly by the following equation

$$w(t+1) = w_{max} - (((w_{max} - w_{min}) * i_t) / (i_{tmax})) , \quad wmax > wmin \qquad (2.8)$$

where, w_{min} and w_{max} are the final and initial values of the inertia weight respectively, i_{tmax} is the maximum iteration number, 'i_t' is the current iteration number. Commonly, the inertia weight decreases linearly from 0.9 to 0.4 over the full run. Trelea have defined a condition that (w < ((cp+cg) / 2)-1) guarantees the convergence. Cyclic or Divergent behavior can occur in the process if this condition is not satisfied.

The technique of inertia weight is quite useful to ensure convergence. However there is a disadvantage of the inertia weight method that once the inertia weight is decreased, it cannot increase even if the swarm needs to search new areas. This method is not able to recover its exploration mode.

2.3.6 Constriction Coefficient

When the algorithm of particle swarm is allowed run without restraining the velocity in some way, the system simply explodes after some iterations. In the initial stage, researchers used V max but the reason for this was not understood fully. Kennedy (Kennedy, 1998) noted that the trajectories of one-dimensional, non-stochastic particles contained interesting regularities when sum of the acceleration constants φ1 and φ2 were in the range of 0.0 and 4.0.

The re- attempt to analyze the trajectories was conducted by Ozcan and Mohan (Ozcan and Mohan, 1999).They reported that the particles were "surfing the waves" of underlying sinusoidal curves. However, Clerc's analysis of the iterative system demonstrated that the behavior discovered by Ozcan and Mohan was in fact the signature of a five-dimensional attractor (Clerc and Kennedy, 2002).

The simplest constriction described by Clerc (Clerc and Kennedy, 2002) as type 1″ constriction is the simplifed system:

$$v_{ij}(t+1) = \chi \, [\, (v_{ij}(t) + U[0, \varphi_1] * (\, (y_{ij}(t) - x_{ij}(t)) + U[0, \varphi_2] * (\, y\hat{}_{ij}(t) - x_{ij}(t)) \,] \qquad (2.9)$$

$$x_{ij}(t+1) = x_{ij}(t) + v_{ij}(t+1) \qquad (2.10)$$

The following formula is used to compute the constriction coeffcient:

$$\chi = \frac{2k}{\left| 2 - \varphi - \sqrt{\varphi^2 - 4\varphi} \right|} \quad \text{where } k \in [0,1],\ \varphi = \varphi_1 + \varphi_2,\ \varphi > 4 \qquad (2.11)$$

Most researchers using the constriction method use ' set to 4.1 (thus having $\varphi_1 = \varphi_2 = 2.05$) and $k = 1$ which determines that $\chi \sim\sim 0.729$.

This is algebraically equivalent to using the inertial model $\chi \sim\sim 0.729$.

And $\varphi_1 = \varphi_2 \sim\sim 1.49445$.

The constriction method results in convergence over time; the amplitude of the trajectory's oscillations decreases over time. When $k = 1$ convergence is slow enough to allow thorough exploration before the search converges.

The advantage of using constriction is that there is no need to use V_{max} nor to guess the values for any parameters governing convergence and preventing explosion. Subsequent experiments (Eberhart and Shi, 2000) concluded that it was prudent to set V_{max} to X_{max}, the dynamic range of each variable in each dimension. The result is a particle swarm algorithm with no problem-specific parameters, considered the canonical particle swarm algorithm.

2.3.7 Neighbourhood Topologies

A neighbourhood must be defined for each particle. This neighbourhood determines the extent of social interaction within the swarm and influences a particular particle's movement. Less interaction occurs when the neighbourhoods in the swarm are small. For small neighbourhood, the convergence will be slower but it may improve the quality of solutions. For larger neighbourhood, the convergence will be faster but the risk that sometimes convergence occurs earlier. To solve this problem, the search process starts with small neighbourhoods size and then the small neighbourhoods size is increased over time. This technique ensures an initially high diversity with faster convergence as the particles move towards a promising search region.

The PSO algorithm is social interaction among the particles in the entire swarm. Particles communicate with one another by exchanging information about the success of each particle in the swarm. When a particle in the whole swarm finds a better position, all particles move towards this particle. This performance of the particles is determined by the particles' neighborhood. Researchers have worked on developing this performance by designing different types of neighborhood structures. Some neighborhood structures or topologies are discussed below:

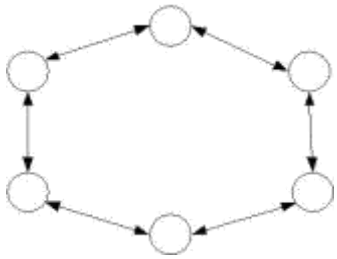


Fig 2.4 Star or gbest

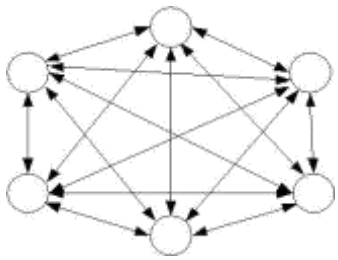


Fig 2.5 Ring or lbest

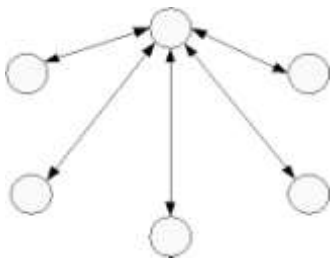


Fig 2.6 Wheel

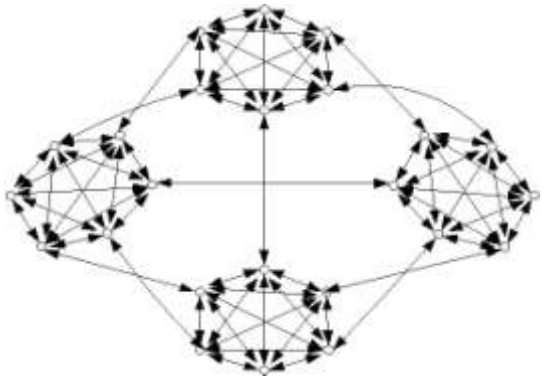


Fig 2.7 Four Clusters

Figure 2.4 explains the star topology, where each particle is connected with every other particle. This topology basically leads to convergence at a faster rate than other topologies, but there is a chance to be trapped in local minima. Because all particles are aware about each other, this topology is known as the g_{best} PSO.

Figure 2.5 represents the ring topology, where each particle is associated to its immediate neighbours. In this particular process, when better result is found by one particle then this particle gives it to its immediate neighbours and these immediate neighbours gives it to their individual immediate neighbours, until it gained by the last particle. Thus the best result found is spreaded very slowly in a ring made by all particles. Convergence is slow, but a great part of the search space is covered than with the star topology. It is known as the l_{best} PSO.

Figure 2.6 shows the wheel topology, in this only one of the particle (a focal particle) associates to the others, and all informations are communicated through this particular particle. This focal particle compares the best performance of all the particles in the swarm and adjusts its own position towards the best performance particle analyse by itself and finally the new position of the focal particle is shared with all the particles.

Figure 2.7 represents a four clusters topology, where four cliques (or clusters) are connected with one edge between opposite clusters and two edges between neighboring clusters. There are more different topologies or neighborhood structures (for instance, Von Neumann topology, the pyramid topology and so on), but there is no single best topology still known to find the required optimum for all varities of optimization problems.

2.4 APPLICATIONS OF PSO

The different application areas of PSO are discussed in this chapter. Kennedy and Eberhart in 1995 made the first practical application of Particle Swarm Optimization. They worked in the field of neural network training and finally reported the algorithm jointly. PSO has been successfully used over a great range of applications, for example, system control, combinatorial optimization , data mining, telecommunications power systems design, , network training, signal processing and many other areas. In present days, PSO algorithms have also been introduced to solve the multi-objective optimization problems , constrained problems, problems related to dynamically changing landscapes, and to find multiple solutions, while the initial PSO algorithm was mainly used to solve single-objective optimization, unconstrained problems. Different areas where PSO is nowadays applied are listed in Table2.1.

Table 2.1 Application areas of Particle Swarm Optimization

Antenna Design		The design of phased arrays and optimal control, reflector antennas, design of Yagi-Uda arrays, broadband antenna design and modeling, synthesis of antenna arrays, adaptive array antennas, optimization of a reflect array antenna, far-field radiation pattern reconstruction, antenna modeling, array failure correction, design of planar antennas, conformal antenna array design, , design of a periodic antenna arrays, near-field antenna measurements, design of patch antennas optimization of profiled corrugated horn antennas, design of implantable antennas.
Signal Processing		Design of IIR filters, Pattern recognition of flatness signal, speech coding, 2D IIR filters ,analogue filter tuning, nonlinear adaptive

		filters, Costas arrays, particle filter optimization wavelets, blind detection, blind source separation, localization of acoustic sources, distributed odour source localization, and so on.
Networking		Bluetooth networks Radar networks, auto tuning for universal mobile telecommunication system networks, TCP network control, optimal equipment placement in mobile communication, routing, peer-to-peer networks , wavelength division-multiplexed network, WDM telecommunication networks, wireless networks, grouped and delayed broadcasting, bandwidth and channel allocation, bandwidth reservation, voltage regulation, transmission network planning, network reconfiguration and expansion, economic dispatch problem, distributed generation, microgrids, cellular neural networks, design of radial basis function networks, feed forward neural network training, product unit networks, congestion management, neural gas networks, design of recurrent neural networks, neuron controllers, wireless sensor network design, wavelet neural networks estimation of target position in wireless sensor networks, wireless video sensor

	networks optimization.
Biomedical	Human tremor analysis for the diagnosis of Parkinson's disease, inference of gene regulatory networks, human movement biomechanics optimization, RNA secondary structure determination, phylogenetic tree reconstruction, cancer classification, and survival prediction, DNA motif detection, biomarker selection, protein structure prediction and docking, drug design, radiotherapy planning, analysis of brain magneto encephalography data, electroencephalogram analysis, biometrics and so on.
Electronics and electromagnetic	On-chip inductors, configuration of FPGAs and parallel processor arrays, fuel cells, circuit synthesis, FPGA-based temperature control, AC transmission system control electromagnetic shape design, microwave filters, generic electromagnetic design and optimization applications, CMOS RF wideband amplifier design, linear array antenna synthesis, conductors, RF IC design and optimization, semiconductor optimization, high-speed CMOS, frequency selective surface and absorber design, voltage flicker measurement, shielding, digital circuit design.
Robotics	Control of robotic manipulators and arms, Motion planning and control, odour source

		localization, soccer playing, robot running, robot vision, collective robotic search, transport robots, voice control of robots, unsupervised probotic learning, path planning, obstacle avoidance, swarm robotics, unmanned vehicle navigation, environment mapping and so forth.
Design and Modelling		Conceptual design, electromagnetics case, induction heating cooker design, VLSI design, power systems, RF circuit synthesis, worst case electronic design, motor design, filter design, antenna design, CMOS wideband amplifier design, logic circuits design, transmission lines, mechanical design, library search, inversion of underwater acoustic models, modeling MIDI music, customer satisfaction models, thermal process system identification, friction models, model selection, ultrawideband channel modeling, identifying ARMAX models, power plants and systems, chaotic time series modeling, model order reduction.
Image and Graphics		Image segmentation, autocropping for digital photographs, synthetic aperture radar imaging, locating treatment planning landmarks in orthodontic x-ray images, image classification, inversion of ocean

		color reflectance measurements, image fusion, photo time-stamp recognition, traffic stop-sign detection, defect detection, image registration, microwave imaging, pixel classification detection of objects, pedestrian detection and tracking texture synthesis, scene matching, contrast enhancement, 3D recovery with structured beam matrix, character recognition, image noise cancellation.
Power generation and Controlling		Automatic generation control, power transformer protection, power loss minimization, load forecasting, STATCOM power system, fault-tolerant control of compensators, hybrid power generation systems, optimal power dispatch, power system performance optimization, secondary voltage control, power control and optimization design of power system stabilizers, operational planning for cogeneration systems, control of photovoltaic systems, large-scale power plant control, analysis of power quality signals, generation planning and restructuring, optimal strategies for electricity production, production costing operation planning.
Fuzzy systems, Clustering, data mining		Design of neurofuzzy networks, fuzzy rule extraction, fuzzy control, membership functions optimization, fuzzy modeling, fuzzy classification, design of hierarchical

		fuzzy systems, fuzzy queue management, clustering, clustering in large spatial databases, document and information clustering, dynamic clustering, cascading classifiers classification of hierarchical biological data, dimensionalit prediction, genetic-programming-based classification, fuzzy clustering, classification threshold optimization, electrical wader sort classification, data mining, feature selection.
Optimization		Electrical motors optimization, optimization of internal combustion engines, optimization of nuclear electric propulsion systems, floor planning, travelling-sales man problems, n-queens problem, packing and knapsack, minimum spanning trees, satisfiability, knights cover problem, layout optimization, path optimization, urban planning, FPGA placement and routing.
Prediction and forecasting		Water quality prediction and classification, prediction of chaotic systems, streamflow forecast, ecological models meteorological predictions, prediction of the floe stress in steel, time series prediction, electric load forecasting, battery pack state of charge estimation, predictions of elephant migrations, prediction of surface roughness in end milling, urban traffic flow forecasting, and so on.

2.4 ADVANTAGES AND DISADVANTAGES OF PSO

PSO algorithm is said to be one of the most useful and powerful methods for solving the non-smooth optimization problems but there are few disadvantages associated to the PSO algorithm. The advantages and disadvantages of the PSO technique are discussed below:

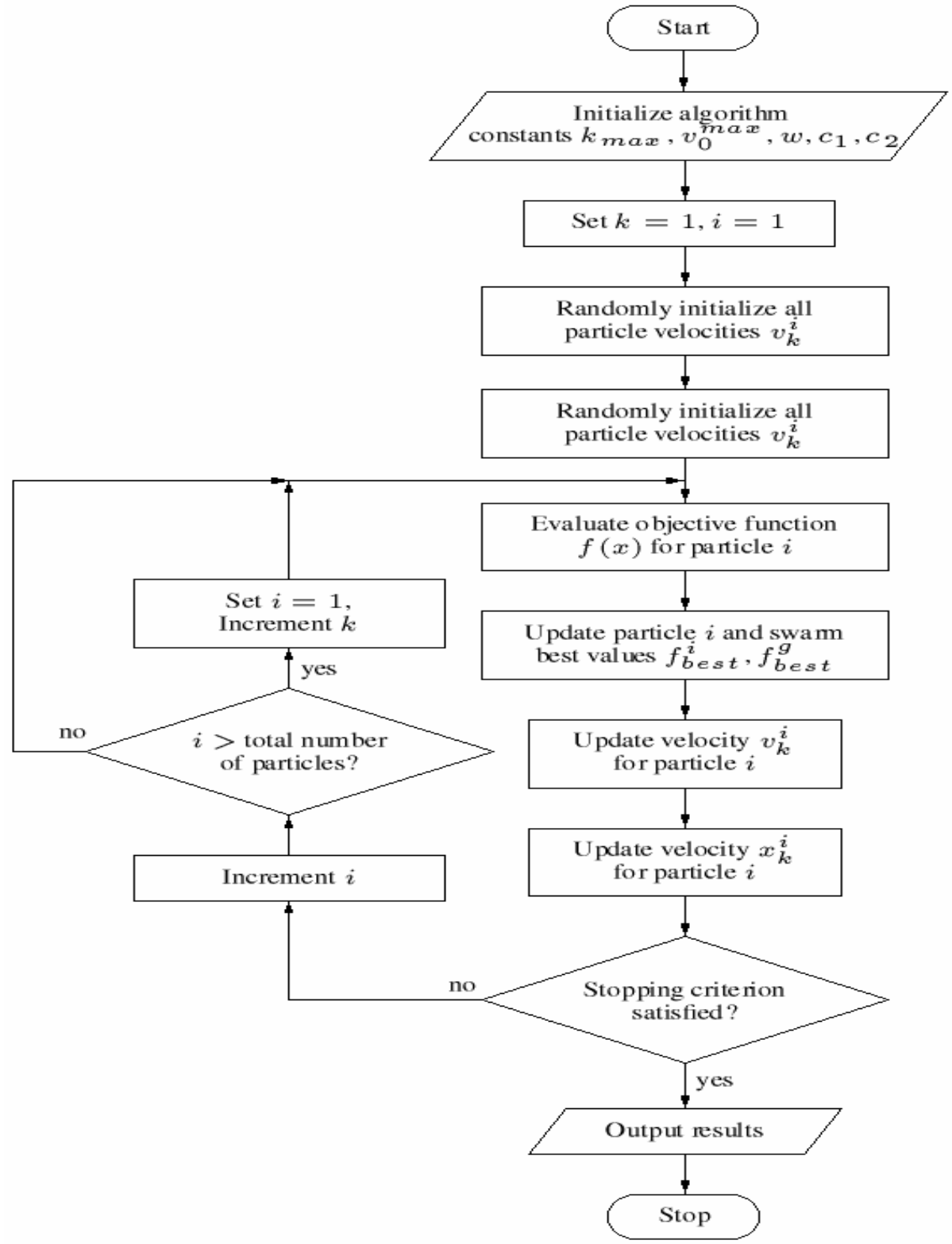
2.4.1 Advantages of the PSO algorithm:

1 PSO technique includes a derivative-free algorithm.
2 It is easily implemented, so it can be applied both in engineering problems and scientific research.
3 Number of parameters is limited and these parameters accounts for a lesser impact to the solutions as compared to other techniques of optimization.
4 Its algorithm includes a very simple calculation algorithm.
5 Few modifications ensures the rapid convergence i.e. the optimum value of the problem gets calculated easily within a short time.
6 PSO is less dependent on a set of initial points as compared to other optimization techniques.
7 Conceptually PSO is very easy.

2.4.2 Disadvantages of the PSO algorithm:

1 PSO algorithm suffers from the partial optimism, which degrades the regulation of its speed and direction.
2 Problems with non-coordinate system (for instance, in the energy field) exists.

2.5 FLOW CHART



2.5.1Steps involved in particle swarm optimization in MATLAB

1 Using the zero command of MATLAB initialize all the variable matrices.

2 Set the values of the random no.s ‘r1’ & ‘r2’ assigned to the personal and global best expressions respectively.

3 Set the values of acceleration constants ‘cp’ & ‘cg’ assigned to the personal and global best expressions respectively.

4 Set the tolerance value.

5 Generate the random position values of particles for all the variables (eg. x1,x2) and also generate the random velocities values of particles (eg. v1,v2) for all the variable.

6 Calculate the fitness for the assumed values of the positions of the particles.

7 Using the above fitness personal best and global best values for all variables are deduced.

8 Using the previous iteration values of personal best, global best and velocity vectors new velocities are generated in the current iteration using the equation:

$$v_{ij}(t+1) = w * v_{ij}(t) + cp * r_1 * (y_{ij}(t) - x_{ij}(t)) + cg * r_2 * (\hat{y}_{ij}(t) - x_{ij}(t))$$

9 Using the new velocity vector and the old position vector , a new position vector is generated for all the variables.

10 Calculate fitness using the new positions in the current iteration.

11 Using the new fitness values the personal and global best values are updated.

12 The difference between the previous and the current fitness is calculated and check against the tolerance value, if within the tolerance iteration stops and global best value is the solution else iteration flow goes back to previous step for further updation.

CHAPTER-3

APPLICATION OF PSO TO MATHEMATICAL BENCHMARK TEST FUNCTIONS

3.1 BENCHMARK FUNCTIONS

Artificial landscapes the second name given to the **Test functions,** are very useful to evaluate characteristics of optimization algorithms. In this case of application of Particle Swarm Optimization to the mathematical benchmark functions, the PSO algorithm can be applied directly to the particular mathematical function, i.e. without any modification. As the mathematical functions are single objective functions and no equality criteria on the fitness functions values, no further formulation for objective function is required and the inequality constraints on the variables, if present, are taken care of in the PSO algorithm itself.

Using particle swarm optimization the basic steps for solving the optimization problem is same as discussed before but if some modifications are provided then we can use it for any type of objective function. Here, we have used PSO for the optimization of some mathematical benchmark functions, which are as follows:

- De jong's function

$$f(x) = \sum_{i=1}^{n} x_i^2 \qquad (3.1)$$

- Booth's function

$$f(x1,x2) = (x1 + 2 * x2 - 7)^2 + (2 * x1 + x2 - 5)^2 \qquad (3.2)$$

- Beale function

$$f(x1,x2) =(1.5-x1+x1*x2)^2+(2.25-x1+x1*x2^2)^2+(2.625-x1+x1*x2^3)^2 \qquad (3.3)$$

- Axis parallel hyper-ellipsoid function

$$f(x) = \sum_{i=1}^{n} (i * x_i^2) \qquad (3.4)$$

- Rosenbrock function(2D)

$$f(x_1,x_2) = 100(x_2-x_1)^2 +(x_1-1)^2 \qquad (3.5)$$

- Rosenbrock function(3D)

f(x1,x2)=100*((x2-x1^2)^2)+ (x1-1)^2 + 100*((x3-x2).^2).^2) + (x2-1).^2;

- Hyper-ellipsoid function

f(x1,x2)= x1^2 + 2 * x2^2 ; (3.7)

- Rastrigin's function

f(x1,x2) = 10 + (x1^2) - (10 * cos(2*pi*x1)); (3.8)

3.2 THE DIFFERENT PARAMETERS USED IN PARTICLE SWARM OPTIMIZATION

The various parameters of particle swarm optimization are as follows:

1. No. of particles in the swarm, **p**.
2. Max. no. of iteration, **it**.
3. Random no. for personal and global factors $\mathbf{r_p}$ and $\mathbf{r_g}$.
4. Acceleration constant for the personal and global factors, $\mathbf{c_p}$ and $\mathbf{c_g}$.
5. Tolerance value, **T**.

The values of these parameters for optimizing various mathematical benchmark functions were chosen as:

1. P= 30
2. it= 1000
3. r_p=0.5 and r_g=0.6
4. c_p=2 and c_g=2

5. T= 10^(-7)

3.3 MATHEMATICAL RESULTS AND DISCUSSION

Various benchmark functions and results obtained after application of PSO has been discussed as follows:

3.3.1 Axis parallel HYPER-Ellipsoid function

$$f(x) = \sum_{i=1}^{n} (i * x_i^2)$$

Minimum value and range for the function are as follows:

- Min. Value: f(x1,x2)=0
- Range : -5.12 =< x1,x2 <= 5.12

TABLE 3.1 Application of different inertia weights to Axis parallel HYPER-Ellipsoid Function

	No. of particles	No. of iterations	Function value	X1	X2	X3
Linearly decreasing	30	185	1.052977e-13	-2.90284e-07	-9.719746e-08	-2.009151e
Simulated annealing	30	37	7.542848e-12	-5.24452e-07	-1.847722e-06	-3.82684e-
Constant IW						
0.4	30	42	8.61517e-13	-7.1316e-08	2.6742e-07	-2.4103e
0.6	30	98	1.19253e-18	-4.54225e-10	-5.3095e-10	4.81666e
0.8	30	162	3.41920e-14	-1.3987e-07	6.07023e-08	-8.5200e

Table 3.1 shows the application of PSO on the Axis parallel HYPER-Ellipsoid function with different inertia weights. Simulated annealing method and constant inertia weight of 0.4 both provides the desired results in less time and with more accuracy as compared to linearly decreasing inertia weight and constant inertia weight with values

of 0.6 and 0.8 while constant inertia weight of 0.4 is more accurate as compared to simulated annealing.

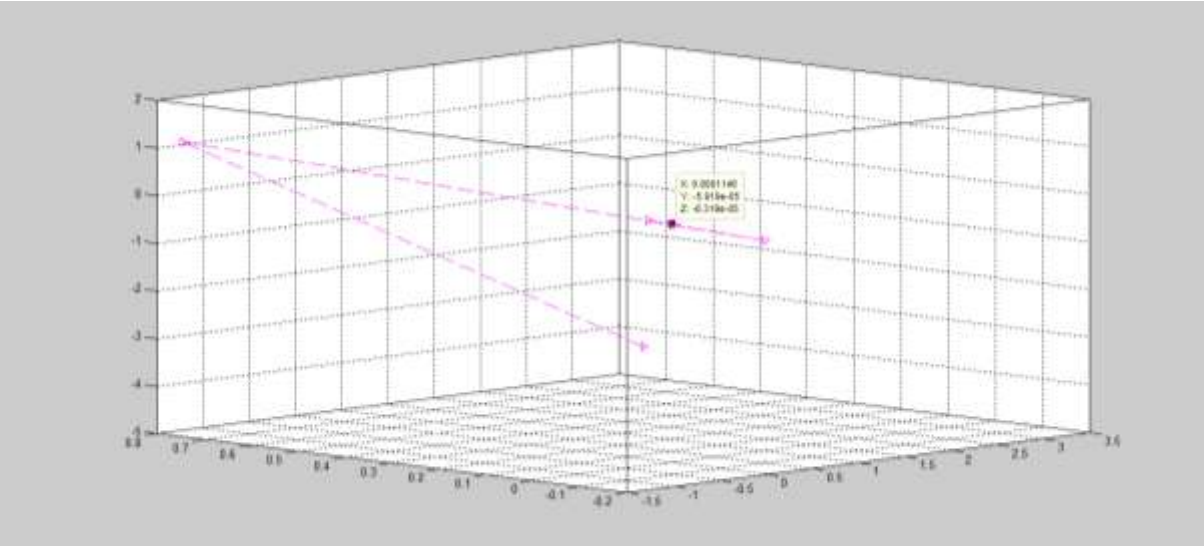


Fig 3.1 Movement of particle 1 of x1,x2 and x3

Fig 3.1 shows the 3D view of change in the position values of paricle 1 of x1, x2 and x3 with the increase in number of iterations. It shows that as the iterations increases particle 1 tries to attain co-ordinates (0,0,0) i.e. x1=0, x2=0 and x3=0, for which function approaches it minimum value.

3.3.2 Booth's Function

y(x1,x2) = (x1 + 2*x2 – 7)^ 2 + (2*x1 + x2 – 5)^ 2 ;

Minimum value and range for the function are as follows:

- Min. Value: y(x1,x2)=0
- Range : -10 =< x1,x2 <= 10

Table 3.2 Application of different inertia weights to Booth's Function

	No. of particles	No. of iterations	Function value	X1	X2
Linearly decreasing	30	134	5.341514e-10	1.000000e+00	3.000000e+0

Simulated annealing	30	52	3.408260e-10	1.000002e+00	2.999998e+0
Constant IW **0.4** **0.6** **0.8**	30 30 30	30 66 119	1.57313e-10 3.465076e-15 1.224909e-11	9.999918e-01 1.000000e+00 1.000003e+00	3.000009e+ 3.000000e 3.000000e

Table 3.2 shows the application of PSO on the BOOTH'S function with different inertia weights. Simulated annealing method and constant inertia weight of 0.4 both provides the desired results in less time and with more accuracy as compared to linearly decreasing inertia weight and constant inertia weight with values of 0.6 and 0.8 while constant inertia weight of 0.4 is more accurate as compared to simulated annealing.

Fig 3.2 Shows the variation of the function value with the increase in iterations and it can be seen that the function approaches its minimum value that is zero in this case as the iterations count increases.

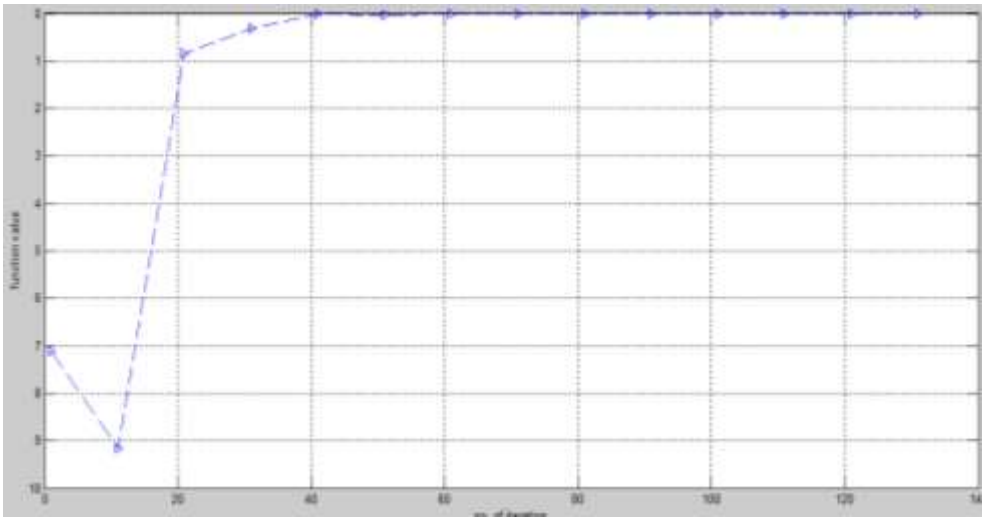


Fig 3.2 variation of function values with iterations

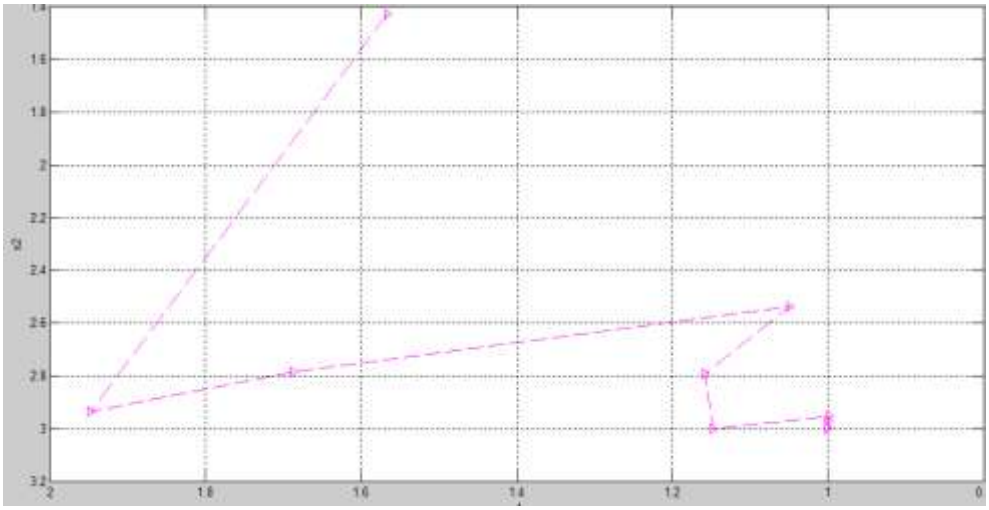


Fig 3.3 Movement of particle 1 of x1 and x2

Fig 3.3 shows the change in the position values of paricle 1 of x1 and x2. It shows that as the iterations increases particle 1 tries to attain co-ordinates (1,3) i.e. x1=1 and x2=3, for which function approaches it minimum value.

3.3.3 Beale Function

f(x1,x2) = (1.5-x1+ x1*x2)^2+(2.25 -x1+x1*x2^2)^2+(2.625 - x1+x1*x2^3)^2;

Minimum value and range for the function are as follows:

- Min. Value: f(x1,x2)=0
- Range : -4.5 =< x1,x2 <= 4.5

Table 3.3 Application of different inertia weights to Beale's Function

	No. of particles	No. of iterations	Function value	X1	X2
Linearly decreasing	30	132	4.044835e-10	-7.305251e-06	2.031228e-0
Simulated annealing	30	22	4.42063e-12	-1.257050e-06	1.585567e-0

Constant IW					
0.4	30	31	1.847590e-10	-1.311315e-05	-2.530239e
0.6	30	76	4.059354e-17	3.925190e-09	-2.802216e
0.8	30	119	1.904880e-11	3.950240e-06	1.233997e

Table 3.3 shows the application of PSO on the Beale's function with different inertia weights. Simulated annealing method and constant inertia weight of 0.4 both provides the desired results in less time and with more accuracy as compared to linearly decreasing inertia weight and constant inertia weight with values of 0.6 and 0.8 while simulated annealing is more accurate and takes lesser computational time as compared to constant inertia weight of 0.4.

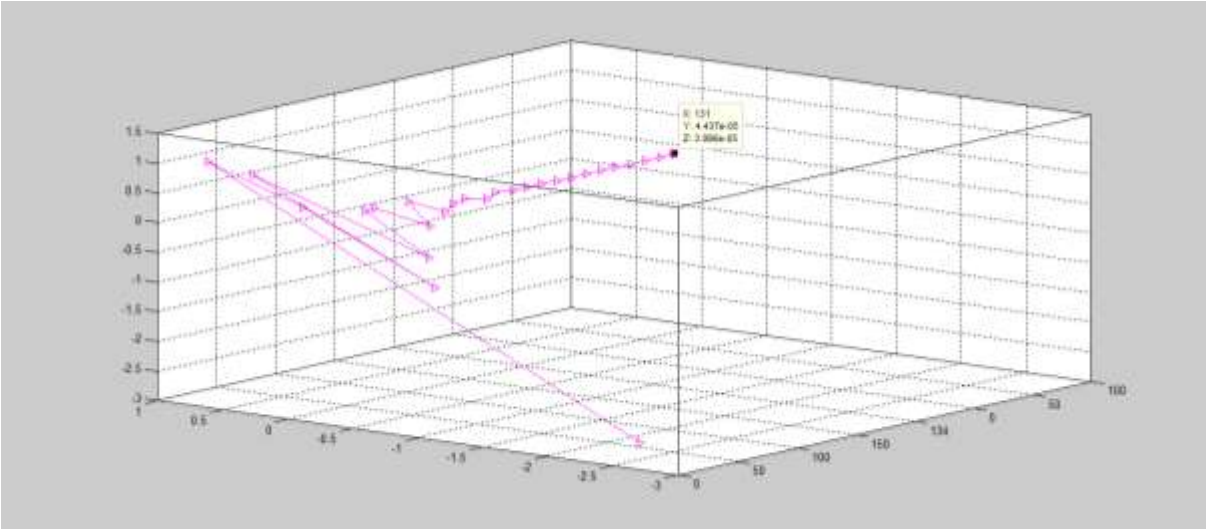


Fig 3.4 Movement of particle 1 for x1 and x2 with iterations (3D view)

Fig 3.4 shows the 3D view of change in the position values of paricle 1 of x1 and x2 with the increase in number of iterations. It shows that as the iterations increases particle 1 tries to attain co-ordinates (0,0) i.e. x1=1 and x2=3, for which function approaches it minimum value.

3.3.4 Sphere's Function

$$f(x) = \sum_{i=1}^{n} x_i^2$$

Minimum value and range for the function are as follows:

- Min. Value: f(x1,x2)=0
- Range : -5.12 =< x1,x2 <= 5.12

TABLE 3.4 Application of different inertia weights to Sphere's Function

	No. of particles	No. of iterations	Function value	x1	X2	X3
Linearly decreasing	30	204	6.297194e-04	1.210595e-02	-1.267901e-02	-1.795573e-
Simulated annealing	30	59	2.161785e-04	3.343558e-03	7.272390e-03	1.233335e-(
Constant IW						
0.4	30	45	4.179781e-04	1.484689e-02	5.736923e-03	-1.28310 02
0.6	30	68	2.503011e-03	3.182450e-02	-3.464515e-02	-1.70271 02
0.8	30	149	1.790865e-03	=2.765392e-02	1.995319e-02	-2.50598 02

Table 3.4 shows the application of PSO on the SPHERE'S function with different inertia weights. Simulated annealing method and constant inertia weight of 0.4 both provides the desired results in less time and with more accuracy as compared to linearly decreasing inertia weight and constant inertia weight with values of 0.6 and 0.8 while simulated annealing is more accurate as compared to constant inertia weight of 0.4.

3.3.5Rastrigin's Function

$$f(x) = \sum_{i=1}^{n} (x_i^2 - 10 * \cos(2 * pi * x_i))$$

Minimum value and range for the function are as follows:

- Min. Value: f(x1,x2)=0
- Range : -5.12 =< x1,x2 <= 5.12

Table 3.5 shows the application of PSO on the Rastrigin function with different inertia weights. Simulated annealing method and constant inertia weight of 0.4 both provides the desired results in less time and with more accuracy as compared to linearly decreasing inertia weight and constant inertia weight with values of 0.6 and 0.8 while constant inertia weight of 0.4 is more accurate as compared to simulated annealing.

Table 3.5 Application of different inertia weights to Rastrigin's Function

	No. of particles	**No. of iterations**	**Function value**	**X1**
Linearly decreasing	30	159	9.84107e-13	-3.109719e-09
Simulated annealing	30	70	0	-1.640973e-09
Constant IW				
0.4	30	52	0	2.480308e-09
0.6	30	89	0	-1.092749e-09
0.8	30	185	0	3.838570e-10

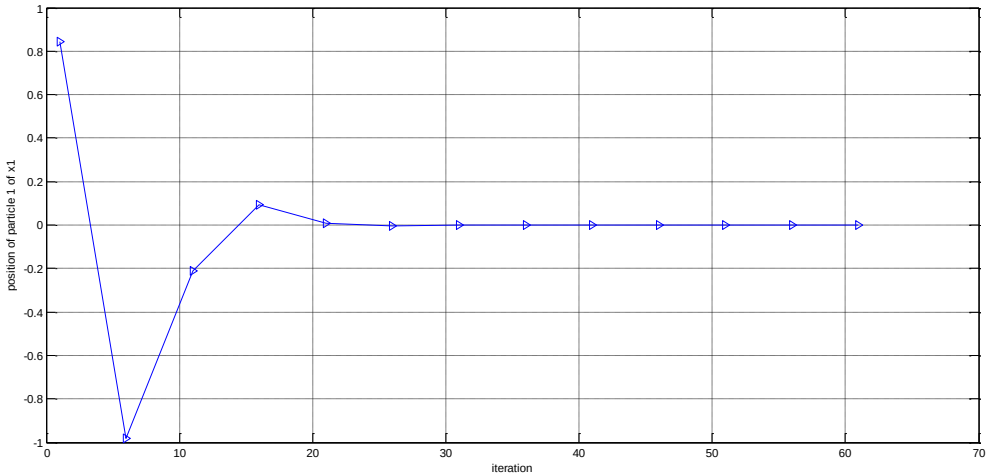


Fig 3.5 Movement of position of particle 1 of x1 with iterations

Fig 3.5 shows the variation of the position of particle 1 of x1 and it gives us the desired result of obtaining the zero value as the number of iterations increases.

3.3.6 Dejong's Function

f(x1,x2)= x1(j,1)^2 + x2(j,1)^2 + x3(j,1)^2;

Minimum value and range for the function are as follows:

- Min. Value: f(x1,x2)=0
- Range : -1 =< x1,x2 <= 1

Table 3.6 Application of different inertia weights to Dejong's Function

	No. of particles	No. of iterations	Function value	X1	X2	X3
Linearly decreasing	30	206	1.118761e-04	-6.825449e-03	5.561369e-03	5.861784e-0
Simulated annealing	30	60	8.930197e-05	-6.624732e-03	6.090060e-03	-2.885491e-
Constant IW						
0.4	30	43	7.9667e-04	1.545415e-02	2.148686e-02	-9.80591e
0.6	30	70	2.1869e-03	1.276701e-02	4.051782e-02	-1.95503e
0.8	30	153	8.8508e-04	-1.47405e-02	-1.65747e-02	1.982660

Table 3.6 shows the application of PSO on the Rastrigin function with different inertia weights. Simulated annealing method and constant inertia weight of 0.4 both provides the desired results in less time and with more accuracy as compared to linearly decreasing inertia weight and constant inertia weight with values of 0.6 and 0.8 while constant inertia weight of 0.4 is more accurate as compared to simulated annealing.

3.3.7 Rosenbrock Function

$$f(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^{\,2})^2 + (1 - x_i)^2]$$

where , $-2.048 < x_i < 2.048$, i = 1,2,3....,n;

3.3.7.1 Rosenbrock Function (2d)

f(x1,x2)=100*((x2-(x1.^2)).^2)+(x1(1,j)-1).^2;

Minimum value and range for the function are as follows:

- Min. Value: f(x1,x2)=0
- Range : -2.048 =< x1,x2 <= 2.048

Table 3.7 Application of different inertia weights to Rosenbrock's Function

	No. of particles	No. of iterations	Function value	X1	X2
Linearly decreasing	30	188	2.967168e-22	1.000000e+00	1.000000e+00
Simulated annealing	30	44	7.344731e-15	1.000000e+00	1.000000e+00
Constant IW					
0.4	30	45	2.35023e-13	9.999997e-01	999993e-01
0.6	30	53	2.434551e-0	1.048215e+00	1.099803e+00
0.8	30	55	4.85839e-04	1.022022e+00	1.044437e+00

Table 3.7 shows the application of PSO on the Rosenbrock function with different inertia weights. Simulated annealing method and constant inertia weight of 0.4 both provides the desired results in less time and with more accuracy as compared to linearly decreasing inertia weight and constant inertia weight with values of 0.6 and 0.8 while constant inertia weight of 0.4 is more accurate as compared to simulated annealing.

3.3.7.2 Rosenbrock Function (3d)

f(x1,x2,x3)=100*((x2-(x1.^2).^2)+(x1-1).^2+100*((x3-(x2).^2).^2)+(x2-1).^2;

Minimum value and range for the function are as follows:

- Min. Value: f(x1,x2,x3)=0
- Range : -2.048 =< x1,x2 <= 2.048

Table3.8 Application of different inertia weights to Rosenbrock's Function

	No. of particles	**No. of iterations**	**Function value**	**X1**	**X2**	**X3**
Linearly decreasing	30	188	2.967168e-22	1.000000e+00	1.000000e+00	1.000000e
Simulated annealing	30	44	7.344731e-15	1.000000e+00	1.000000e+00	1.000000e
Constant IW						
0.4	30	45	2.3502e-13	9.999997e-01	9.99993e-01	9999959
0.6	30	53	2.43451e-0	1.048215e+00	1.099803e+00	1.05960
0.8	30	55	4.8589e-04	1.022022e+00	1.044437e+00	1.03483

Table 3.8 shows the application of PSO on the Rosenbrock function with different inertia weights. Simulated annealing method and constant inertia weight of 0.4 both provides the desired results in less time and with more accuracy as compared to linearly decreasing inertia weight and constant inertia weight with values of 0.6 and 0.8 while constant inertia weight of 0.4 is more accurate as compared to simulated annealing.

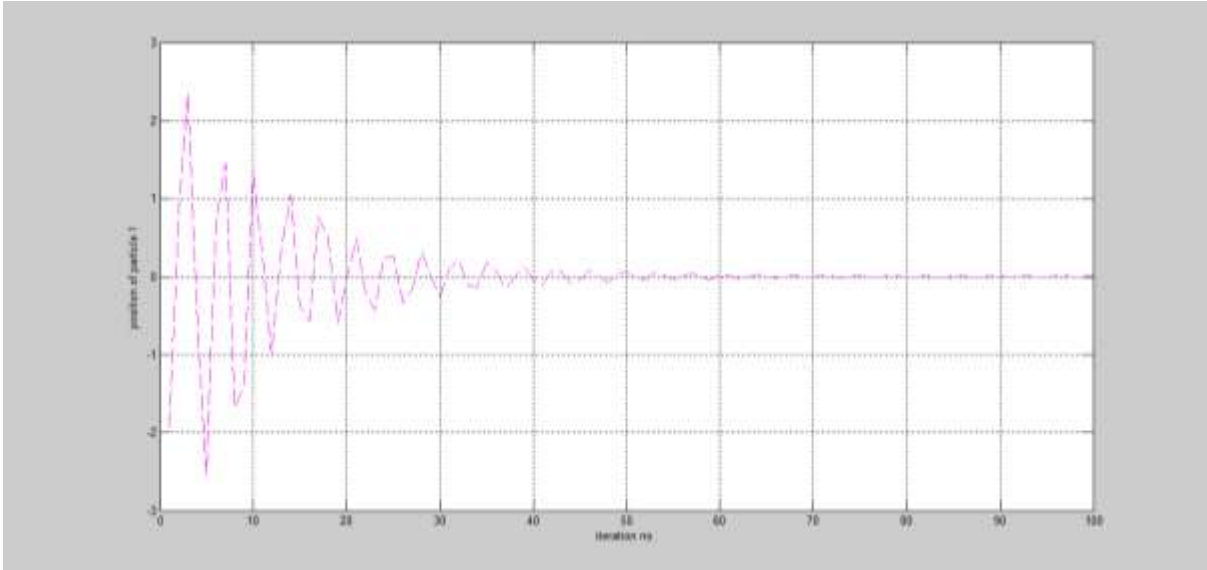


Fig 3.6 Position vs iteration for particle 1 of x1

Fig 3.6 shows the variation in the position of particle 1 i.e. x1 of particle 1. It shows the desired result of approaching zero value with the increase in number of iterations where function approaches its minimum value.

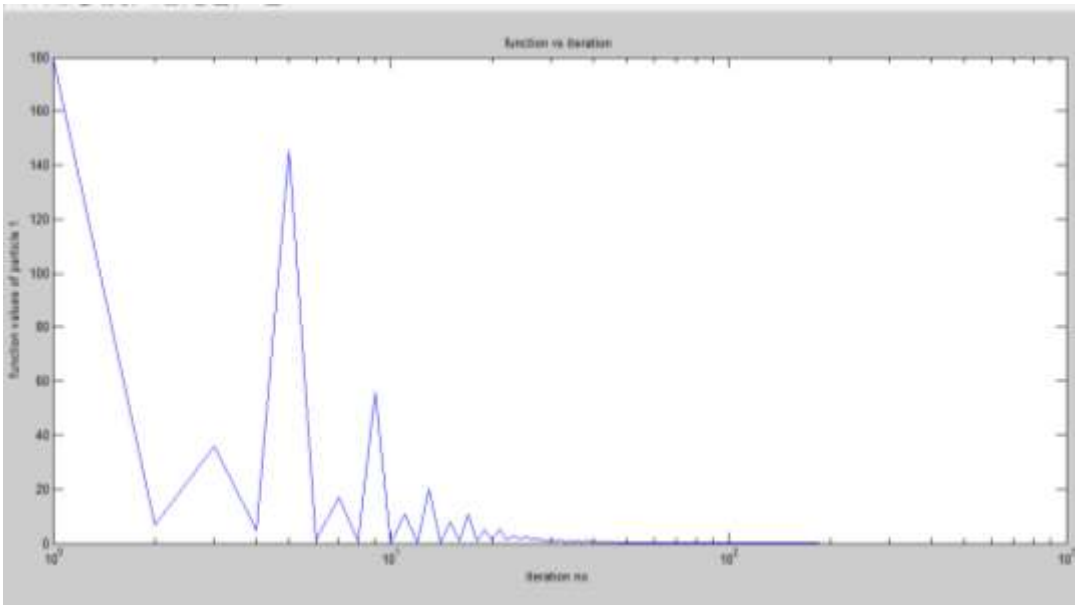


Fig 3.7 Function value vs iteration on log scale

Fig 3.7 Shows the variation of the function value with the increase in iterations and it can be seen that the function approaches its minimum value that is zero in this case as

the iterationscount increases.Log scale helps us to have a closer look on the initial movement of theparticle.

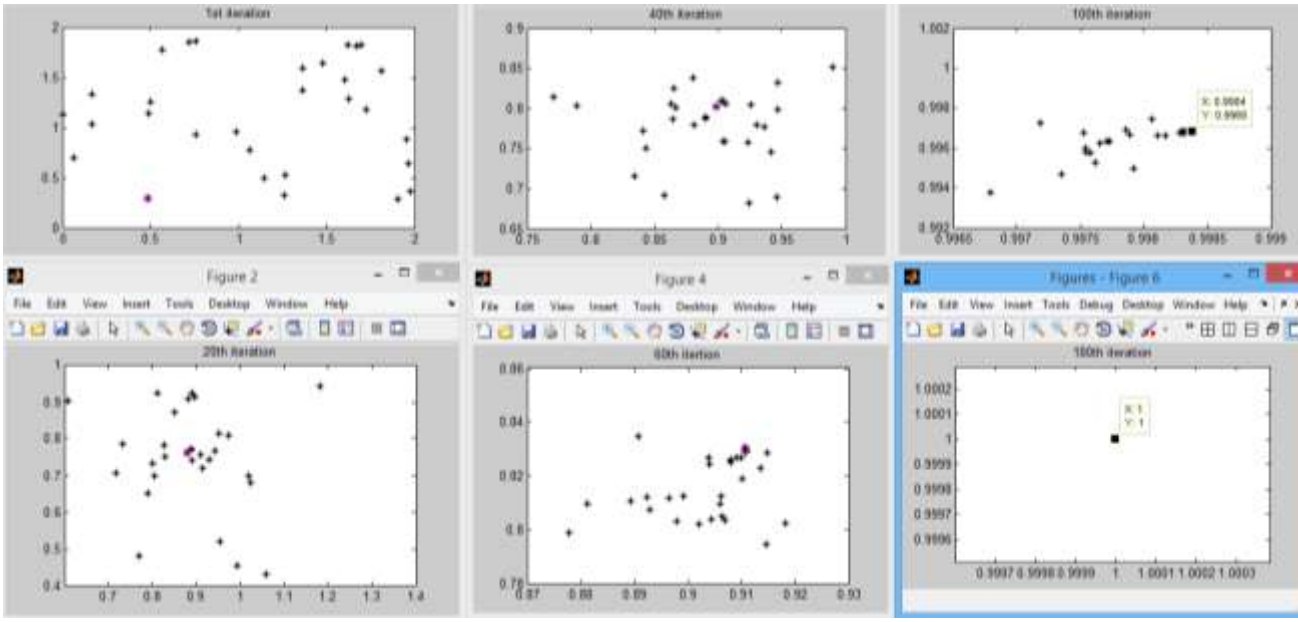


Fig 3.8 Convergence of particles with iterations

Fig 3.8 Shows the convergence of all the particles at the optimum value with increase in the no. of iterations. Initially all the particles were located randomly in the search space but as iterations increases all particles move closer and closer and finally converge at a single point. The figure shows the initial position i.e. the position after 1st iteration, position after 20th iteration, position after 40th iteration, position after 60th iteration and final (global) position of the 30 particles of the swarm for solving rosenbrock‘s function of two variables. The figure also shows the location of global best position represented by the purple coloured circle.

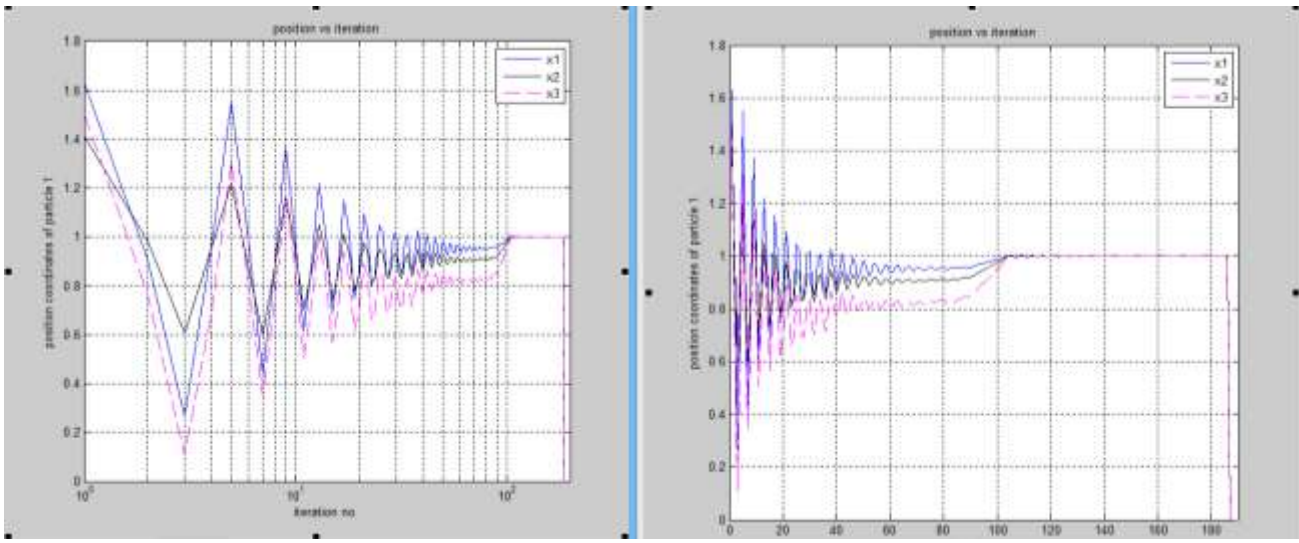


Fig 3.9 All position co-ordinates of particle 1 vs iterations

Fig3.9 shows the variation of the all the position co-ordinates of particle 1 with the increase in no. of iterations.Initially the variation is quite large and to have a beter view of initial movement of particles log scale is used and it is seen that with increase in the number of iterations all the particles converge at a point giving the desired result corresponding to the optimum value of x1=x2=x3=1.

3.4 DISCUSSION

The results obtained when PSO algorithm is applied on mathematically benchmark functions show that the optimum values are obtained successfuly for all the benchmark functions taken into account namely Axis Hyper-Ellipsoid, Beale‘s, Dejong’s, Booth‘s, Rosenbrock‘s, Sphere and Rastrigin’s function. The values for random numbers r1=0.5 and r2=0.6 and constriction factors cp=2 and cg =2 are used in the velocity modification equation. It is observed that while keeping constriction factors and random numbers of constant values and varying the inertia weight(simulated annealing method, constant inertia weight method or linearly decreasing method) gives different computational time. The simulated annealing method was found to be most accurate and in all the cases it took lesser number of iterations as compared to linearly decreasing method and constant inertia weight except in some cases where constant inertia weight of 0.4 provided the least computational time(rastigin’s function, dejong’s function, sphere function).

CHAPTER 4:

ECONOMIC LOAD DISPATCH

4.1 PROBLEM FORMULATION IN 2-D SPACE WITH EQUALITY CONSTRAINTS

Ojective function being used to minimize the cost of generation is given as :

$$F_C = \sum_{i=1}^{NG} F[C_i(P_{gi})] \quad (4.1)$$

Where:

$$C_i(P_{gi}) = \sum_{i=1}^{NG} a_i P_{gi}^2 + b_i P_{gi} + c_i \quad (4.2)$$

Where:

P_{gi} is the active power generation at the i^{th} generator.

C_i is the cost of generation for i^{th} generator.

NG is the total number of generators in the system.

a_i, b_i, c_i are fuel cost coefficients of i^{th} generator.

The objective function used to find the system transmission losses is given as:

$$F_L = \sum_{m=1}^{NG} \sum_{n=1}^{NG} P_{gm} B_{mn} P_{gn} + \sum_{m=1}^{NG} B_{om} P_{gm} + B_{oo} \quad (4.3)$$

Where

P_{gm}, P_{gn} is the active power at the m^{th} and n^{th} generator.

NG is the total number of generators in the system.

B_{mn}, B_{om}, B_{oo} are loss coefficients.

The cost and loss coefficients of various generators are given in Table 4.1 and 4.2.

TABLE 4.1Values of Cost Coefficients

	Coefficient	G1	G2	G3
5-BUS	A	0.0050	0.0050	……
	B	3.510	3.889	……
	C	44.40	40.60	……
14-BUS	A	0.005	0.005	0.005
	B	3.510	3.890	2.450
	C	44.40	40.60	105
30-BUS	A	0.005	0.005	0.005
	B	3.510	3.890	2.450
	C	44.40	40.6	105

TABLE 4.2Values of Loss Coefficients

B_{11}	0.0003489	0.0003489	0.0003069
B_{12}	0.000086	0.000068	0.0001289
B_{13}	---	-0.0000389	0.000002
B_{22}	0.000371	0.0001570	0.0001520
B_{23}	---	0.000015	0.0000110
B_{33}	---	0.000274	0.000189
B_{01}	---	0.000044	---
B_{02}	---	0.000024	---
B_{03}	---	0.000000	---
B_{00}	---	0.000254	---

The method used in this thesis, has been developed by Kron and adopted by Kirchmayer, which is the loss coefficient method.

Mathematically, the problem is to minimize

$F= [F_C (P_{g1}, P_{g2}, P_{g3}... P_{gNG})$

Where

$$F = W_C F_C \quad (4.4)$$

Subject to the constraints:

Equality constraint

$$\sum_{i=1}^{NG} P_{gi} = P_D + P_L \quad (4.5)$$

Inequality constraint

$$P_{gimin} \leq P_{gi} \leq P_{gimax} \quad i = 1, 2 \ldots NG \quad (4.6)$$

Power outputs from the generators are taken as the independent decision variables of the problem.

Where:

F	objective function to be optimized
F_C	cost of the generation
F_L	system transmission losses
$P_{g1}, P_{g2} \ldots P_{gNG}$	are the generations at the generators.
P_D	is the total load demand.
P_L	is the system transmission losses.
NG	is the no. of generators.
P_{gi}	generation from i^{th} generator
P_{gimin}	minimum generation possible from i^{th}generator
P_{gimax}	maximum generation possible from i^{th} generator

4.2 COMPUTATIONAL PROCEDURE FOR APPLICATION OF PSO IN ECONOMIC LOAD DISPATCH

Particle Swarm Optimization (PSO) has been used to perform the optimization of ELD function. To consider the equality constraint of the problem, the function has been modified by inclusion of a parameter K. The objective function becomes as follows:

$$F=W_C F_C+K\ (P_D+P_L-P_G) \qquad (4.7)$$

Where:

Parameter K is fixed at 100 for all three IEEE 5, 14 and 30 bus systems. Different values of K were considered and it was observed that ELD problem converged when it was fixed to 100 for all the systems.

Inequality constraints have been considered in the PSO programming which is done in the MATLAB. The program checks the power output of each particle for each generator in each iterations and the power is tied to the corresponding limit violated. Logic to implement the inequality constraint is as shown below:

for i=1: N_G

for m=1: p

if $P_{gi} < P_{gimin}$

$P_{gi} = P_{gimin}$

end

if $P_{gi} > P_{gimax}$

$P_{gi} = P_{gimax}$

end

end

end

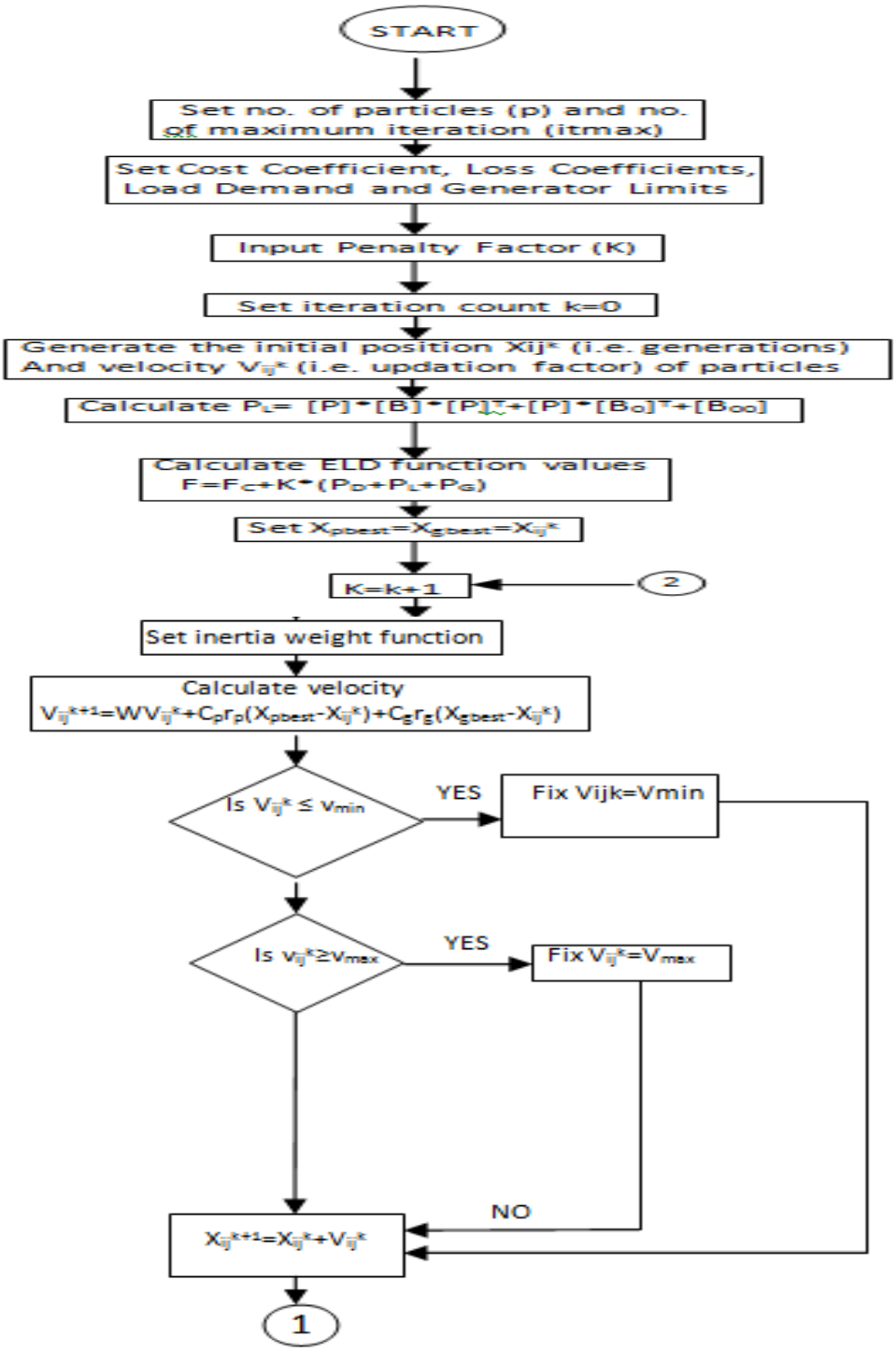
The optimum solution is obtained when the

i. Change in the value of Economic Load Dispatch function during successive iterations is less than the limit specified which is $\varepsilon=10^{-7}$ and
ii. The equality constraint is satisfied such that the absolute value of difference between generation, demand and losses is less than $\varepsilon=10^{-7}$.

Population size of the swarm and the maximum number of iteration can be selected by the user of the program in run time. We have chosen 30 particles in the swarm and 1000 as the maximum number of iterations.

With the help of MATLAB, we generate randomly the initial position and velocity of particles. To increase the convergence rate, limits are imposed on position of particles. Here positions i.e. the generations are decision variables. The maximum and minimum limits on the velocity have been assigned as $V_{min} = -P_{gimin}/2$ and $V_{max} = P_{gimax}/2$ respectively. The velocities are fixed to the values of corresponding limits if violated during the iterations. Initial values of personal best and global best have been taken as the initial value randomly generated by MATLAB.

Flowchart of solution of Economic Load Dispatch problem using PSO is shown in Fig. 4.1.



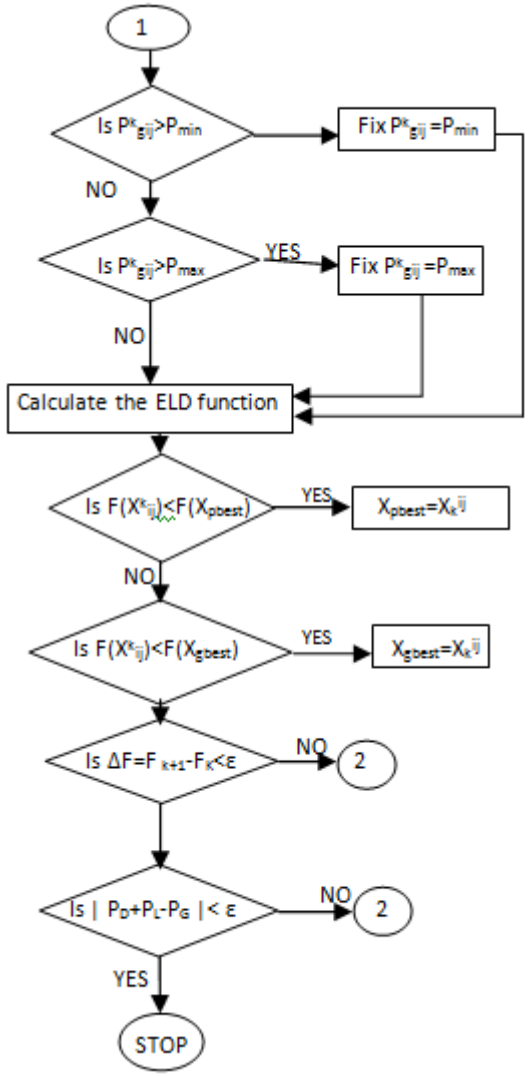


Fig. 4.1 Flow chart of implementation of PSO on ELD

The sequence for the solution of Economic Load Dispatch problem using Particle Swarm Optimization technique is explained as follows:

1. Fix the no. of particles 'p' in swarm and set the no. of maximum iterations it_{max}.
2. Fix the cost coefficients, loss coefficients, and load demand and generator limits of all the generators.
3. Generate X_{ij}^{k} and V_{ij}^{k}, the initial random positions (i.e. generations) and velocity (i.e. updation factor) respectively.
4. Set iteration count K = 0.
5. Calculate the losses for each particle, using the eq. (4.3).
6. Calculate the value of ELD function using eq. (4.7).

7. At 0^{th} iteration the personal and global best positions (i.e. generations) are same as the initial random positions (i.e. generations).
8. Increase the iteration count k by 1 using k=k+1.
9. Calculate the velocity (i.e. positions updation factor) of each particle using eq. (2.1).
10. Check if velocity is within the limits. Fix the velocity to the limit violated.
11. Calculate the new positions (i.e. generations) of the particles by evaluating eq. (2.2).
12. Check if generations (i.e. positions) of each particle are within the generator limits, if not fix the generation to the limit violated.
13. Calculate ELD function for the new positions (i.e. generations) generated.
14. Update Xpbest and Xgbest values by comparing ELD function values.
15. Check if both the stopping criteria are satisfied, if not then go to step 9, else stop.
16. Output the values of cost of generation and system transmission losses.

4.3 COMPUTATIONAL RESULTS IN 2D SPACE

Three standard test systems have been taken into account in the economic load dispatch function in order to examine the cost of generation aspects and detailed studies have been carried out in table 4.3 to 4.6.

TABLE 4.1 Variation of no of iteration required with different IPSO for IEEE 5bus system

	No. of particles	**No. of iterations**	**Fc($/h)**	**P1**	**P2**
Linearly decreasing	30	208	762.44	87.56	77.55
Simulated annealing	30	65	761.19	95.40	69.78
Constant IW					
0.4	30	51	761.4	92.93	72.22
0.6	30	74	761.84	90.19	74.93

0.8	30	162	761.27	94.25	70.92

TABLE 4.2 Variation of no of iteration required with different IPSO for IEEE 14bus system

	No. of particles	No. of iterations	Fc($/h)	P1	P2	P3
Linearly decreasing	30	176	1149.7	160.07	46.71	62.91
Simulated annealing	30	58	1145.502	155.73	58.829	54.961
Constant IW						
0.4	30	52	1146.43	132.86	78.05	57.05
0.6	30	67	1146.54	135.98	70.02	62.04
0.8	30	196	1146.58	139.08	65.63	63.50

TABLE 4.3 Variation of no of iteration required with different IPSO for IEEE 30bus system

	No. of particles	No. of iterations	Fc($/h)	P1	P2	P3
Linearly decreasing	30	222	1259.70	149.67	98.2	47.95
Simulated annealing	30	71	1256.057	154.72	81.15	59.63
Constant IW						
0.4	30	58	1256.21	157.2	77.8	60.61
0.6	30	80	1256.307	155.79	77.34	62.32
0.8	30	202	1277.97	114.19	79.54	98.60

For IEEE 5 bus system simulated annealing inertia weight varying technique gives the lowest cost of production while linearly decreasing gives the worst result along with the largest computational time.

For IEEE 14 bus system simulated annealing inertia weight varying technique again gives the lowest cost of production while worst result is given by linearly decreasing function and largest computational time is taken by constant inertia weight of 0.8.

For IEEE 30 bus system simulated annealing inertia weight varying technique gives the lowest cost of production again while constant inertia weight of 0.8 gives the worst result.

Overall it can be concluded that for all the three IEEE bus systems invoved in this work simulated annealing inertia weight varying technique provides the best computational results.

CHAPTER 5:
CONCLUSION AND FUTURE DIRECTIONS

5.1CONCLUSIONS

The Particle Swarm Optimization technique has been applied to various benchmark functions and optimu values are obtained in each case. It has been experimentally found that simulated annealing inertia weight varying technique gives out the best results.

In this thesis ELD problem has been solved for IEEE 5, 14 and 30 bus systems taking cost of generation as an objective to be minimised. Inequality constraints of the problem have been handled by the PSO programming whereas equality constraint of ELD problem has been considered using penalty parameter k. The results show that the computational time and the number of iterations are considerably reduced while using simulated annealing inertia weight varying technique. This work shows that simulated annealing inertia weight varying technique provides better results when comaperd to constant inertia weight of 06 and 0.8 and linearly decreasing inertia weight function taking computational time and optimum values as the basis of comparision and constant inertia weight of 0.4 also provide as good results as that provided by simulated annealing inertia weight varying technique but constant inertia weight has the disadvantage of getting trapped in the local minima so in multi-objective functions the performance of constant inertia weight will always remain under consideration but no such risk is involved with simulated annealing inertia weight varying technique.

5.2 FUTURE DIRECTION

In this work one of the parameter in the PSO algorithm is varied i.e. inertia weight while keeping other parameters constant.So, there is a lot of scope available in the area of PSO, as one can consider different selection criterias for varying different constant parameters such as random no.s (rp & rg) and acceleration coefficients (cp & cg). In this work for soving the ELD problem only cost of fuel is taken into account while one can work while considering different objectives of power system as loses in the system, security, environmental degradation due to pollution etc.

APPENDIX- I

1) IEEE 5 BUS SYSTEM

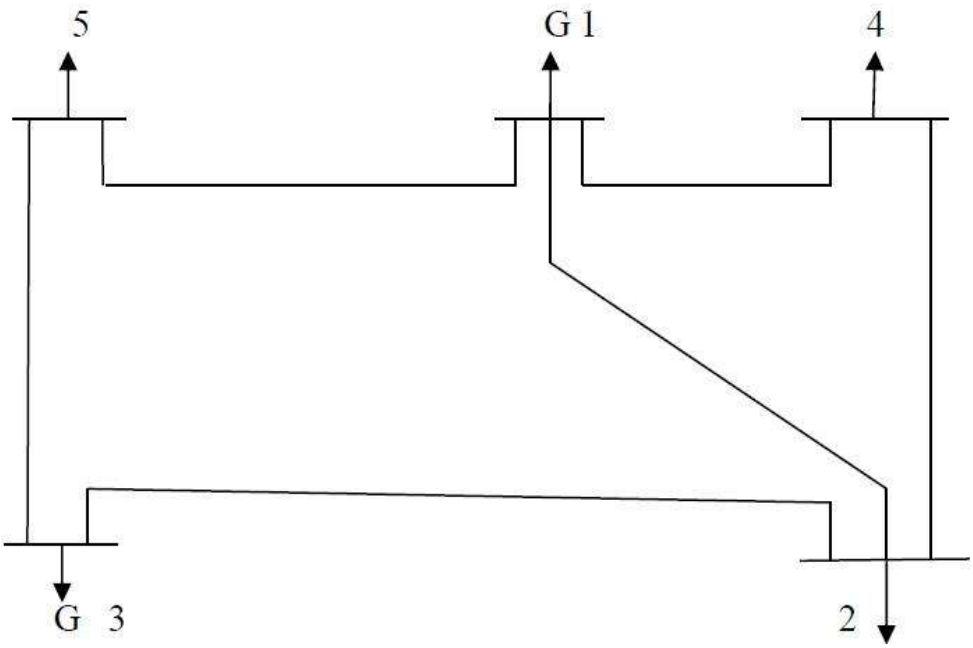


Fig. I-A: BUS-CODE DIAGRAM OF 5 BUS SYSTEM

TABLE I-A: LINE DATA or IMPEDANCE DATA (5 BUS SYSTEM)

LINE DESIGNATION	*R(p.u.)	*X(p.u.)	LINE CHARGING
1-2	0.10	0.4	0.0
1-4	0.15	0.6	0.0
1-5	0.05	0.2	0.0
2-3	0.05	0.2	0.0
2-4	0.10	0.4	0.0
3-5	0.05	0.2	0.0

*The impedance are based on MVA as 100

TABLE I-B: BUS DATA or OPERATING CONDITIONS (5 BUS SYSTEM)

	GENERATION		LOAD	
BUS NO.	MW	VOLTAGE MAGNITUDE	MW	MVAR
1*	- - -	1.02	- - -	- - -
2	- - -	- - -	60	30
3	100	1.04	- - -	- - -
4	- - -	- - -	40	10
5	- - -	- - -	60	20

*Slack Bus

TABLE I-C: REGULATED BUS DATA (5 BUS SYSTEM)

BUS NO.	VOLTAGE MAGNITUDE	MVAR CAPACITY		MW CAPACITY	
		MINIMUM	MAXIMUM	MINIMUM	MAXIMUM
1	1.02	0.0	60	30	120
3	1.04	0.0	60	30	120

The nodal load voltage inequality constraints are $0.9 \leq Vi \leq 1.05$

Cost characteristics of IEEE 5 bus system

The cost characteristics of the IEEE 5 Bus System are as follows: $C_1=50p_1^2+351p_1+44.4$ \$/hr.

$C_3=50p_3^2+389p_3+40.6$ \$/hr.

Here, the total load demand of the system is 160 MW. Maximum and minimum active power constraint on the generator bus for the given system is 120 MW and 30 MW respectively. Voltage magnitude constraint for generator at bus 3 is 1.04 pu.

M-file For Calculating B- Coefficients:

```
Clear
basemva=100
accuracy=0.0001
maxiter=10
busdata=[1 1 1.02 0 0 0 0 0 0 60 0;2 0 1 0 60 30 0 0 0 0 0;3 2 1.04 0 0 0 82 0 0 60
0;4 0 1 0 40 10 0 0 0 0 0;5 0 1 0 60 20 0 0 0 0 0];
Linedata=[1 2 0.10 0.4 0 1;1 4 0.15 0.6 0 1; 1 5 0.05 0.2 0 1;2 3 0.05 0.2 0 1;2 4 0.10
0.4 0 1;3 5 0.05 0.2 0 1];
disp(busdata)
disp(linedata)
mwlimit=[30 120;30
120]; Ifybus
Ifnewton
busout
bloss
```

B-Coefficient Calculated is as:

B11 = 0.00035336 B12 = 0.0000103196

B21 = 0.0000103196 B22 = 0.000368992

2) IEEE 14 BUS SYSTEM

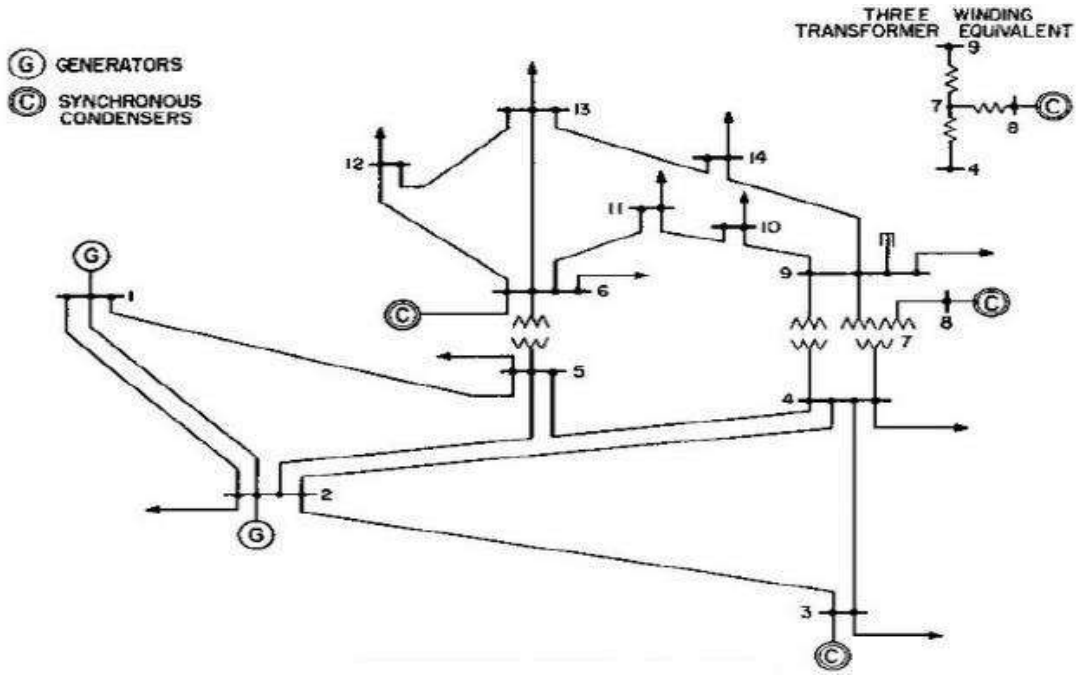


Fig. I-B: BUS-CODE DIAGRAM OF 14 BUS SYSTEM

TABLE I-D: IMPEDANCE & LINE-CHARGING DATA (14 BUS SYSTEM)

Line Designation	Resistance p.u. *	Reactance p.u. *	Line Charging	Tap Setting
1-2	0.019379	0.059170	0.0264	1
1-5	0.054029	0.223040	0.0264	1
2-3	0.046980	0.197970	0.0219	1
2-4	0.058110	0.176320	0.0187	1
2-5	0.056950	0.173880	0.0170	1
3-4	0.067010	0.171030	0.0173	1
4-5	0.013350	0.042110	0.0064	1
4-7	0	0.20912	0	1
4-9	0	0.55618	0	1
5-6	0	0.25202	0	1
6-11	0.09498	0.19890	0	1
6-12	0.12291	0.25581	0	1
6-13	0.06615	0.13027	0	1
7-8	0	0.17615	0	1
7-9	0	0.11001	0	1
9-10	0.03181	0.08450	0	1
9-14	0.12711	0.27038	0	1
10-11	0.08205	0.19207	0	1
12-13	0.22092	0.19988	0	1
13-14	0.17093	0.34802	0	1

* Impedance and line-charging susceptance in p.u. on a 100 MVA base.

TABLE I-E: BUS DATA or OPERATING CONDITIONS (14 BUSSYSTEM)

Bus No.	Voltage		Generation		Load	
	Magnitude (in pu)	Phase angle (deg.)	MW	MVAR	MW	MVAR
1*	1.06	0	0	0	0	0
2	1	0	40	0	21.7	12.7
3	1	0	0	0	94.2	19.0
4	1	0	0	0	47.8	-3.9
5	1	0	0	0	7.6	1.6
6	1	0	0	0	11.2	7.5
7	1	0	0	0	0	0
8	1	0	0	0	0	0
9	1	0	0	0	29.5	16.6
10	1	0	0	0	9.0	5.8
11	1	0	0	0	3.5	1.8
12	1	0	0	0	6.1	1.6
13	1	0	0	0	13.5	5.8
14	1	0	0	0	14.9	5.0

*Slack Bus

TABLE I-F: REGULATED BUS DATA (14 BUS SYSTEM)

Bus no.	Voltage magnitude (in pu)	Minimum MVAR capability	Maximum MVAR capability
2	1.05	-40	50
3	1.010	0	40
6	1.070	-6	24
8	1.090	-6	24

Cost characteristics of IEEE 14 bus system

The cost characteristics of the IEEE 14 Bus System are as follows: $C_1 = 50p_1^2+245p_1+105$ \$/hr.

$C_2 = 50p_2^2+351p_2+44.4$ \$/hr.

$C_6 = 50p_6^2+389p_6+40.6$ \$/hr.

Here, the total load demand of the system is 259 MW. The maximum active power constraint is 200 MW, 100MW and 100 MW for the generators of bus no. 1, 2 and 6 respectively. The minimum active power constraint is 50 MW, 20MW and 20 MW for the generators of bus no. 1, 2 and 6 respectively. Voltage magnitude

constraint for generator at bus 2 is 1.045, for bus no. 6 is 1.070, for bus no. 3 is 1.010 & for bus no. 8 is 1.090.

M-file For Calculating B- Coefficients:

```
Clear
basemva=100
accuracy=0.0001
maxiter=10
busdata=[1 1 1.06 0 0 150 0 0 0 0 0;2 2 1.045 0 21.7 12.7 63.11 0 -40 50 0;3 0 1.01 0
94.2 19 0 0 0 40 0;4 0 1 0 47.8 -3.9 0 0 0 0 0;5 0 1 0 7.6 1.6 0 0 0 0 0;6 2 1.07 0 11.2
7.5 77.12 0 -6 24 0;7 0 1 0 0 0 0 0 0 0 0;8 0 1.09 0 0 0 0 0 -6 24 0 ;9 0 1 0 29.5 16.6 0
0 0 0 0; 10 0 1 0 9 5.8 0 0 0 0 0;11 0 1 0 3.5 1.8 0 0 0 0 0;12 0 1 0 6.1 1.6 0 0 0 0
0;13 0 1 0 13.5 5.8 0 0 0 0 0;14 0 1 0 14.9 5 0 0 0 0 0];
linedata=[1 2 0.01938 0.05917 0.0264 1;1 5 0.05403 0.22304 0.0246 1; 2 3 0.04699
0.19797 0.0219 1; 2 4 0.05811 0.17632 0.0170 1; 2 5 0.05695 0.17388 0.0173 1; 3 4
0.06701 0.17103 0.0064 1; 4 5 0.01335 0.04211 0.0 1; 4 7 0.0 0.20912 0.0 0.978; 4 9
0.0 0.55618 0.0 0.969;5 6 0.0 0.25202 0.0 0.932; 6 11 0.09498 0.19890 0.0 1;6 12
0.12291 0.25581 0.0 1;6 13 0.06615 0.13027 0.0 1;7 8 0.0 0.17615 0.0 1; 7 9 0.0
0.11001 0.0 1; 9 10 0.03181 0.08450 0.0 1;9 14 0.12711 0.27038 0.0 1; 10 11
0.08205 0.19207 0.0 1;12 13 0.22092 0.19988 0.0 1;13 14 0.17093 0.34802 0.0 1];
disp(busdata)
disp(linedata)
mwlimit=[50 200;20 100;20
100] Ifybus
Ifnewton
busout
bloss
```

B-Coefficient Calculated is as:

B11 = 0.0231 B12 = 0.0078 B13 = -0.0007

B21 = 0.0078 B22=0.0182 B23= 0.0022

B31=-0.0007 B32= 0.0022 B33= 0.0329

C) IEEE 30 BUS SYSTEM

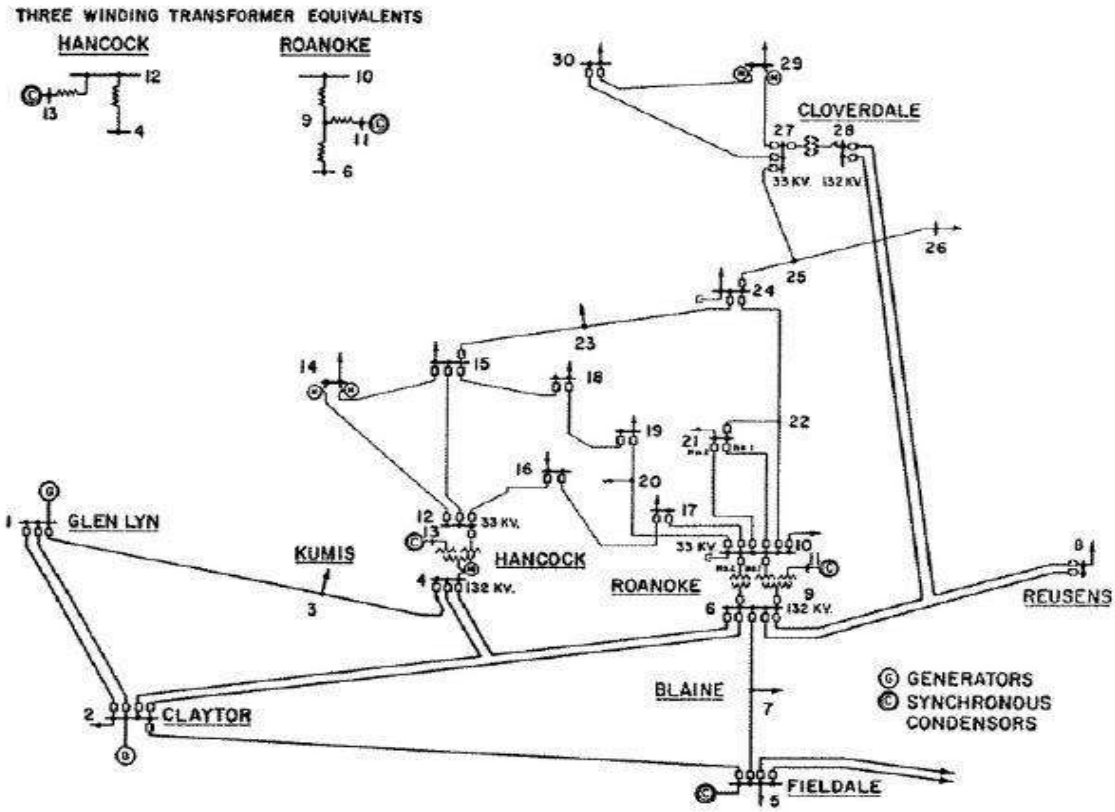


Fig. I-C: BUS-CODE DIAGRAM OF 30 BUS SYSTEM

TABLE I-G: IMPEDANCE & LINE-CHARGING DATA (30 BUS SYSTEM)

Line Designation	Resistance p.u.*	Reactance p.u.*	Line Charging	Tap Setting
1-2	0.0192	0.0575	0.0264	1
1-3	0.0452	0.1852	0.0204	1
2-4	0.0570	0.1737	0.0184	1
3-4	0.0132	0.0379	0.0042	1
2-5	0.0472	0.1983	0.0209	1
2-6	0.0581	0.1763	0.0187	1
4-6	0.0119	0.0414	0.0045	1
5-7	0.0460	0.1160	0.0102	1
6-7	0.0267	0.0820	0.0085	1
6-8	0.0120	0.0420	0.0045	1
6-9	0	0.2080	0	0.978
6-10	0	0.5560	0	0.969
9-11	0	0.2080	0	1
9-10	0	0.1100	0	1
4-12	0	0.2560	0	0.932
12-13	0	0.1400	0	1
12-14	0.1231	0.2559	0	1
12-15	0.0662	0.1304	0	1

12-16	0.0945	0.1987	0	1
14-15	0.2210	0.1997	0	1
16-17	0.0824	0.1923	0	1
15-18	0.1070	0.2185	0	1
18-19	0.0639	0.1292	0	1
19-20	0.0340	0.0680	0	1
10-20	0.0936	0.2090	0	1
10-17	0.0324	0.0845	0	1
10-21	0.0348	0.0749	0	1
10-22	0.0727	0.1499	0	1
21-22	0.0116	0.0236	0	1
15-23	0.1000	0.2020	0	1
22-24	0.1150	0.1790	0	1
23-24	0.1320	0.2700	0	1
24-25	0.1885	0.3292	0	1
25-26	0.2544	0.3800	0	1
25-27	0.1093	0.2087	0	1
27-28	0	0.3960	0	0.968
27-29	0.2198	0.4153	0	1
27-30	0.3202	0.6027	0	1
29-30	0.2399	0.4533	0	1
8-28	0.0636	0.2000	0.0214	1
6-28	0.0169	0.0599	0.0065	1

*Impedance and line-charging susceptance in p.u. on a 100 MVA base.

TABLE I-H: BUS DATA or OPERATING CONDITIONS (30 BUS SYSTEM)

Bus No.	Voltage		Generation		Load	
	Magnitude (in pu)	Phase angle (deg.)	MW	MVAR	MW	MVAR
1*	1.06	0	0	0	0	0
2	1	0	40	0	21.7	12.7
3	1	0	0	0	2.4	
4	1	0	0	0	7.6	
5	1	0	0	0	94.2	
6	1	0	0	0	0	0
7	1	0	0	0	22.8	10.9
8	1	0	0	0	30.0	30.0
9	1	0	0	0	0	0
10	1	0	0	0	5.8	2.0
11	1	0	0	0	0	0
12	1	0	0	0	11.2	7.5
13	1	0	0	0	0	0
14	1	0	0	0	6.2	1.6
15	1	0	0	0	8.2	2.5
16	1	0	0	0	3.5	1.8
17	1	0	0	0	9.0	5.8
18	1	0	0	0	3.2	0.9

19	1	0	0	0	9.5	3.4
20	1	0	0	0	2.2	0.7
21	1	0	0	0	17.5	11.2
22	1	0	0	0	0	0
23	1	0	0	0	3.2	1.6
24	1	0	0	0	8.7	6.7
25	1	0	0	0	0	0
26	1	0	0	0	3.5	2.3
27	1	0	0	0	0	0
28	1	0	0	0	0	0
29	1	0	0	0	2.4	0.9
30	1	0	0	0	10.6	1.9

*Slack Bus

TABLE I-I: REGULATED BUS DATA (30 BUS SYSTEM)

Bus no.	Voltage magnitude (in pu)	Minimum MVAR capability	Maximum MVAR capability
2	1.045	-40	50
5	1.01	-40	40
8	1.01	-10	40
11	1.082	-6	24
13	1.071	-6	24

TABLE I-J: TRANSFORMER DATA (30 BUS SYSTEM)

Transformer designation	Tap setting*
4-12	0.932
6-9	0.978
6-10	0.969
28-27	0.968

*****Off nominal turns ratio, as determined by the actual transformer-tap position and the voltage bases. In the case of nominal turns ratio, this would equal to 1.

TABLE I-K: STATIC CAPACITOR DATA (30 BUS SYSTEM)

Bus no	Susceptance*p.u.
10	0.19
24	0.043

*Susceptance in p.u. on 100 MVA base.

Cost characteristics of IEEE 30 bus system:

The cost characteristics of the IEEE 30 Bus System are as follows:

$C_1 = 50p_1^2+245p_1+105$ \$/hr

$C_2 = 50p_2^2+351p_2+44.4$ \$/hr

$C_8 = 50p_8^2+389p_8+40.6$ \$/hr

The total load demand of the IEEE 30 bus system is 283.4 MW. The maximum active power constraint is 250 MW, 100MW and 100 MW for the generators of bus no. 1, 2 and 8 respectively. The minimum active power constraint is 50 MW, 30MW and 30 MW for the generators of bus no. 1, 2 and 8 respectively. Voltage magnitude constraint for generator at bus 2 is 1.045, for bus no. 5 is 1.01, for bus no. 8 is 1.010, for bus no. 11 is 1.082 &for bus no. 13 is 1.071.

M-file For Calculating B- Coefficients:

```
Clear basemva=100
accuracy=0.0001
maxiter=10
busdata=[1 1 1.06 0 0 0 0 0 0 0 0;2 2 1.045 0 21.7 12.7 90 0 -40 50 0; 3 0 1 0 2.4 1.2 0 0 0 0
0;4 0 1 0 7.6 1.6 0 0 0 0 0;5 0 1.01 0 94.2 19 0 0 -40 40 0; 6 0 1 0 0 0 0 0 0 0 0; 7 0 1 0 22.8
10.9 0 0 0 0 0;8 2 1.010 30 30150 0 -10 40 0; 9 0 1 0 0 0 0 0 0 0 0; 10 0 1 0 5.8 2 0 0 0 0
0.19; 11 0 1.082 0 0 0 0 0 -6 24 0; 12 0 1 0 11.2 7.5 0 0 0 0 0; 13 0 1.071 0 0 0 0 0 -6 24 0; 14
0 1 0 6.2 1.6 0 0 0 0 0;15 0 1 0 8.2 2.5 0 0 0 0 0;16 0 1 0 3.5 1.8 0 0 0 0 0; 17 0 1 0 9 5.8 0 0 0
0 0; 18 0 1 0 3.2 0.9 0 0 0 0 0; 19 0 1 0 9.5 3.4 0 0 0 0 0; 20 0 1 0 2.2 0.7 0 0 0 0 0;21 0 1 0
17.5 11.2 0 0 0 0 0;22 0 1 0 0 0 0 0 0 0 0;23 1 0 3.2 1.6 0 0 0 0 0; 24 0 1 0 8.7 6.7 0 0 0 0
0.043; 25 0 1 0 0 0 0 0 0 0 0;26 0 1 0 3.5 2.3 0 0 0 0 0; 27 0 1 0 0 0 0 0 0 0 0; 28 0 1 0 0 0 0 0
0 0 0;29 0 1 0 2.4 0.9 0 0 0 0 0; 30 0 1 0 10.6 1.9 0 0 0 0 0]; linedata=[1 2 0.0192 0.0575
0.0264 1;1 3 0.0452 0.1852 0.0204 1; 2 4 0.0570 0.19797 0.0219 1; 2 4 0.05811 0.17632
0.0170 1; 2 5 0.05695 0.17388 0.0173 1; 3 4 0.06701 0.17103 0.0064 1; 4 5 0.01335 0.04211
0.0 1; 4 7 0.0 0.20912 0.0 0.978; 4 9 0.0 0.55618 0.0 0.969;5 6 0.0 0.25202 0.0 0.932; 6 11
0.09498 0.19890 0.0 1;6 12 0.12291 0.25581 0.0 1;6 13 0.06615 0.13027 0.0 1;7 8 0.0
0.17615 0.0 1; 7 9 0.0 0.11001 0.0 1; 9 10 0.03181 0.08450 0.0 1;9 14 0.12711 0.27038 0.0
1; 10 11 0.08205 0.19207 0.0 1;12 13 0.22092 0.19988 0.0 1;13 14 0.17093 0.34802 0.0 1];
disp(busdata)
disp(linedata)
mwlimit=[50 150;50 150;50 150]
Ifybus
Ifnewton
busout bloss
```

B-Coefficient Calculated is as:

B11 = 0.0307 B12 = 0.0129 B13 = 0.0002

B21 = 0.0129 B22=0.0152 B23= - 0.0011

B31=0.0002 B32=- 0.0011 B33= 0.0190

APPENDIX II

MATLAB Program for optimization of benchmark functions using PSO.

HYPER-ELLIPSOID FUNCTION

```
clear all
clc
nop=input('enter value for no. of particles=');
%itermax=input('enter value of itermax=');
itermax=1000;
x1=zeros(itermax,nop);
x2=zeros(itermax,nop);
v1=zeros(itermax,nop);
v2=zeros(itermax,nop);
x1(1,:)=unifrnd(-5.12,5.12,1,nop);
x2(1,:)=unifrnd(-5.12,5.12,1,nop);
v1(1,:)=rand(1,nop);
v2(1,:)=rand(1,nop);
for j=1:nop
   pbest(1,j)=x1(1,j);
   pbest(2,j)=x2(1,j);
      y(1,j)=(x1(1,j)).^2 + 2 * (x2(1,j)).^2;
end
%wmax=input('enter value of wmax=');
%wmin=input('enter value of wmin=');
%c1=input('enter value of c1=');
%c2=input('enter value of c2=');
%r1=input('enter value of r1=');
%r2=input('enter value of r2=');
c1=2;
c2=2;
r1=0.5;
r2=0.6;
min=y(1,1);
   b=1;
   for j=2:nop
      if y(1,j)<min
         min=y(1,j);
         b=j;
      end
   end
   q=b;
   gbest(1,1)=x1(1,b);
   gbest(1,2)=x2(1,b);

   gbes(1,1)=gbest(1,1);
   gbes(1,2)=gbest(1,2);

   for j=1:nop
   prevfn(1,j)=y(1,j);
   end
  t=1;
   for i=2:itermax
   for j=1:nop
   %for inertia weight W
 % wmax=0.9;
  % wmin=0.4;
```

```
w(1,1)=0.8;
%w(i,1)= wmin+ (wmax-wmin)* (0.95)^ (i-2);
      %w(i,1)= (wmax-(((wmax-wmin)/itermax)*(i-1)));
      v1(i,j)= w(1,1)*v1(i-1,j)+c1*r1*(pbest(1,j)-(x1(i-1,j)))+c2*r2*(gbest(1,1)-(x1(i-1,j)));
   v2(i,j)= w(1,1)*v2(i-1,j)+c1*r1*(pbest(2,j)-(x2(i-1,j)))+c2*r2*(gbest(1,2)-(x2(i-1,j)));

   x1(i,j)=v1(i,j)+x1(i-1,j);
   x2(i,j)=v2(i,j)+x2(i-1,j);

   newfn(i,j)=(x1(i,j)).^2 + 2 * (x2(i,j)).^2;
   if newfn(i,j)<prevfn(1,j)
         prevfn(1,j)=newfn(i,j);
         pbest(1,j)=x1(i,j);
         pbest(2,j)=x2(i,j);

      end
      y(i,j)=newfn(i,j);
   end
   min=y(i,1);
   p=1;
   for j=2:nop
      if y(i,j)<min
         min=y(i,j);
         p=j;
      end
   end
   gbes(i,1)=x1(i,p);
      gbes(i,2)=x2(i,p);

      if y(i,p)<y(t,q)
         gbest(1,1)=gbes(i,1);
         gbest(1,2)=gbes(i,2);

         t=i;
         q=p;
      end
      for j=1:nop
      df(i,j)= abs(y(i,j)-y((i-1),j)) ;
   end

      ki=0;
   for j=1:nop
      if (df(i,j)<=10^(-7))
         ki=ki+1;
      end
   end
   if ki >= nop
     break
   end
   end
   disp(sprintf('min value of function is %d and at values of x1=%d and x2=%d ',min,gbest(1,1),gbest(1,2)))
```

DEJONG'S FUNCTION

```
clc
disp(' we have to minimize f = 100(x1^2-x2)^2+(1-x1)^2 i.e. rosenbrock function')
p=input('Enter the no. of particles in a swarm');          %no. of particles
it=input('Enter the no. of iterations');
x1=zeros(p,it); %no. of iterations are pre decided that is it will be maximum 50
x2=zeros(p,it);
x3=zeros(p,it);
v1=zeros(p,it);
v2=zeros(p,it);
v3=zeros(p,it);
f=zeros(p,it);
x1g=zeros(p);
x2g=zeros(p);
x3g=zeros(p);
fp=zeros(1,p);
df=zeros(1,(it-1));
rp=0.5;
rg=0.6;
cp=2;
cg=2;
T=input('Enter the tolerance value');
% Initial values i.e. 0th iteration
% x1(:,1)=[1.5605 0.7795 0.4834 0.8078 0.1929 0.2639 1.8841 1.9123 1.1504 0.1196 0.4696 0.7063 1.6424
0.0308 0.0860 0.3380 1.2982 1.4634 1.2955 0.9018];
% x2(:,1)=[1.0940 0.5926 1.4894 0.3779 1.3736 0.3670 0.7370 1.2512 1.5605 0.1623 1.8588 1.5514 0.9736
0.8717 0.8936 0.6127 1.0170 1.0215 1.6353 1.5897];
% v1(:,1)=[0.0326 0.5612 0.8819 0.6692 0.1904 0.3689 0.4607 0.9816 0.1564 0.8555 0.6448 0.3763 0.1909
0.4283 0.4820 0.1206 0.5895 0.2262 0.3846 0.5830];
% v2(:,1)=[0.2518 0.2904 0.6171 0.2653 0.8244 0.9827 0.7302 0.3439 0.5841 0.1078 0.9063 0.8797 0.8178
0.2607 0.5944 0.0225 0.4253 0.3127 0.1615 0.1788];
x1(:,1)=unifrnd(-1,1,1,p);
x2(:,1)=unifrnd(-1,1,1,p);
x3(:,1)=unifrnd(-1,1,1,p);
v1(:,1)=rand(1,p);
v2(:,1)=rand(1,p);
v3(:,1)=rand(1,p);
% i=0;
% disp (sprintf('enter the values of %dth iteration positions of %d particles for variable x1',i,p))
% for j=1:p
%     x1(j,1)=input(sprintf('enter the value of x1(%d,%d)',j,i));
% end
% disp (sprintf('enter the values of 0th iteration positions of %d particles for variable x2',p))
% for j=1:p
%     x2(j,1)=input(sprintf('enter the value of x2(%d,%d)',j,i));
% end
% disp (sprintf('enter the values of 0th iteration positions of %d particles for variable v1',p))
% for j=1:p
%     v1(j,1)=input(sprintf('enter the value of v1(%d,%d)',j,i));
% end
% disp (sprintf('enter the values of 0th iteration positions of %d particles for variable v2',p))
% for j=1:p
%     v2(j,1)=input(sprintf('enter the value of v2(%d,%d)',j,i));
% end
for j=1:p
```

```
    f(j,1)= x1(j,1)^2 + x2(j,1)^2 + x3(j,1)^2;
end


%Initial personal besst values
x1p=x1(:,1);
x2p=x2(:,1);
x3p=x2(:,1);

%for Initial Global best values updation
fmin=min(f(:,1));
for k=1:p
   if f(k,1)==fmin
      gb=k;
   else
   end
end
%Initial global best value
for k=1:p
x1g(k) = x1(gb,1);
x2g(k) = x2(gb,1);
x3g(k) = x3(gb,1);
end
fgm = min(f(:,1));

% fig=zeros(1,485);
% t=zeros(1,485);


for i=1:it
   disp(sprintf('This is %d no. of iteration',i))

%for inertia weight W
%w=0.8;
wmax=0.9;
wmin=0.4;
  w = wmax-i*((wmax-wmin)/it);
 %   w= wmin+ (wmax-wmin)* (0.95)^ (i-2);
   for j=1:p
      v1(j,(i+1)) = w*v1(j,i) + rp*cp*(x1p(j)-x1(j,i)) + rg*cg*(x1g(j)-x1(j,i));
      v2(j,(i+1)) = w*v2(j,i) + rp*cp*(x2p(j)-x2(j,i)) + rg*cg*(x2g(j)-x2(j,i));
      v3(j,(i+1)) = w*v3(j,i) + rp*cp*(x3p(j)-x3(j,i)) + rg*cg*(x3(j)-x3(j,i));
      x1(j,(i+1)) = x1(j,i) + v1(j,(i+1));
      x2(j,(i+1)) = x2(j,i) + v2(j,(i+1));
      x3(j,(i+1)) = x3(j,i) + v3(j,(i+1));
      f(j,(i+1))= x1(j,(i+1))^2 + x2(j,(i+1))^2 + x3(j,(i+1))^2;
   end

%To find change in the values of f
   for j=1:p
      df(j,i)= abs(f(j,(i+1))-f(j,i)) ;
   end

%personal best values updation
   for j=1:p
   fp(j)= x1p(j)^2 + x2p(j)^2;
   end
   for k=1:p
```

```
        if f(k,i)< fp(k)
            x1p(k)=x1(k,i);
            x2p(k)=x2(k,i);
            x3p(k)=x3(k,i);
        else
        end
    end

%for Global best values updation
    if min(f(:,(i+1)))<fgm
        fgm=min(f(:,(i+1)));
    else
    end

    for j=1:i
        for k=1:p
        if f(k,i)==fgm
            for l=1:p
                x1g(l) = x1(k,i);     %global best values
                x2g(l) = x2(k,i);
                x3g(l) = x3(k,i);
            end
        else
        end
        end
    end

    print = [x1(:,i)     x2(:,i)     x3(:,i)     v1(:,i)     v2(:,i)     f(:,i)];
    disp('     x1          x2        x3        v1        v2         f')
    disp(print)

%     fig(i) = figure('Position', [100 100 500 350]);
%     t(i) = uitable('Parent', fig(i), 'Position', [25 25 450 200]);
%     print = [x1(:,i) x2(:,i) v1(:,i) v2(:,i) f(:,i)];
%     set(t(i), 'Data', print);
%     set(t(i), 'ColumnName', {'x1', 'x2', 'v1', 'v2', 'f'});

%Stoping criterion
    ki=0;
    for j=1:p
        if (df(j,i)<=10^(-T))
            ki=ki+1;
        end
    end
    if ki >= p
        break
    end

end
[r,c]=find(f==fgm);
disp(sprintf('min value of function is %d and at values of x1=%d, x2=%d and x3=%d
',fgm,x1g(1),x2g(1),x3g(1)))
```

MATLAB Program for the solution of IEEE 30-bus system using PSO

```
clear all
clc
disp(' we have to minimize the cost function of a 3 machine system')
p=input('Enter the no. of particles in a swarm');           %no. of
particles
it=input('Enter the no. of iterations');
a=10^(-4)*[50 50 50];
b=10^(-2)*[245 351 389];
c=[105 44.4 40.6];
B=10^(-2)*[0.0307 0.0129 -0.0002; 0.0129 0.0152 -0.0011; -0.0002 -0.0011
0.0190];
p1=zeros(p,it);
p2=zeros(p,it);
p3=zeros(p,it);
v1=zeros(p,it);
v2=zeros(p,it);
v3=zeros(p,it);
f=zeros(p,it);
df=zeros(p,it);
sp=zeros(p,it);
csp=zeros(p,it);
pl=zeros(p,it);
c1=zeros(p,it);
c2=zeros(p,it);
c3=zeros(p,it);
C=zeros(p,it);
rp=0.4;
rg=0.5;
cp=2;
cg=2;
pd=283.4;
    p1g=zeros(p);
    p2g=zeros(p);
    p3g=zeros(p);
    fp=zeros(1,p);
    plp=zeros(1,p);

k=100;
w1=1;
w2=0;
% Initial values i.e. 0th iteration
% p1(:,1)=[0.2356 0.5478 1.2453 1.5897];
% p2(:,1)=[1.1254 1.3658 1.9875 1.5632];
n=1;
while n==1
    for j=1:p
    p1(j,1)=unifrnd(50,250,1);
    p2(j,1)=unifrnd(30,100,1);
    p3(j,1)=283.4-p1(j,1)-p2(j,1);
    if p3(j,1)<30&&p3(j,1)>100
        n=1;
        break;
    else
        n=0;
    end
    end
end
% % % v1(:,1)=[-0.2 -0.1 -0.2 -0.2 -0.1 -0.1 -0.2 -0.1 -0.3 -0.2];
% % % v2(:,1)=[-0.2 -0.1 -0.3 -0.1 -0.2 -0.1 -0.3 -0.1 -0.2 -0.1];
v1(:,1)=rand(1,p);
```

```
v2(:,1)=rand(1,p);
v3(:,1)=rand(1,p);

%Total cost calculation
    for j=1:p
         c1(j,1) = a(1)*(p1(j,1))^2 + b(1)*p1(j,1) + c(1);
         c2(j,1) = a(2)*(p2(j,1))^2 + b(2)*p2(j,1) + c(2);
         c3(j,1) = a(3)*(p3(j,1))^2 + b(3)*p3(j,1) + c(3);
         C(j,1) = c1(j,1) + c2(j,1) + c3(j,1);
    end
%To calculate initial value of cost function we need PL
for j=1:p
    pl(j,1)= [p1(j,1) p2(j,1) p3(j,1)]*B*[p1(j,1) p2(j,1) p3(j,1)]';
end
%To calculate initial value of cost function
for j=1:p
    f(j,1)= w1*((a(1)*(p1(j,1))^2 + b(1)*p1(j,1) + c(1)) +
(a(2)*(p2(j,1))^2 + b(2)*p2(j,1) + c(2)) ...
             + (a(3)*(p3(j,1))^2 + b(3)*p3(j,1) + c(3))) + w2*pl(j,1) +
k*abs(pd+pl(j,1)-p1(j,1)-p2(j,1)-p3(j,1));
end
%0th iteration data display
    disp('this is the 0th iteration')
    print0 = [p1(:,1) p2(:,1) p3(:,1) v1(:,1) v2(:,1) v3(:,1) f(:,1)
c1(:,1) c2(:,1) c3(:,1) C(:,1)];
    disp('      P1          P2         P3         V1          v2         V3
f         c1         c2         c3         C ')
    disp(print0)

%Initial personal besst values
p1p=p1(:,1);
p2p=p2(:,1);
p3p=p3(:,1);

%for Initial Global best values updation
fmin=min(f(:,1));
for m=1:p
    if f(m,1)==fmin
        gb=m;
    else
    end
end
%Initial global best value
for m=1:p
p1g(m) = p1(gb,1);
p2g(m) = p2(gb,1);
p3g(m) = p3(gb,1);
end
fgm = min(f(:,1));

%Main iterations starts from here
for i=1:it
    disp(sprintf('This is iteration no.= %d',i))

%for inertia weight W
 %  wmax=0.9;
  %  wmin=0.4;
   % w = wmax-(i-1)*((wmax-wmin)/it);
%w= wmin+ (wmax-wmin)* (0.95)^ (i-2);
w=0.4;
```

```
%For calculatiing velocities for updation
    for j=1:p
        v1(j,(i+1)) = w*v1(j,i) + rp*cp*(p1p(j)-p1(j,i)) + rg*cg*(p1g(j)-
p1(j,i));
        v2(j,(i+1)) = w*v2(j,i) + rp*cp*(p2p(j)-p2(j,i)) + rg*cg*(p2g(j)-
p2(j,i));
        v3(j,(i+1)) = w*v3(j,i) + rp*cp*(p3p(j)-p3(j,i)) + rg*cg*(p3g(j)-
p3(j,i));
    end

    %V(min) and V(max) constraint
    for j=1:p
        if v1(j,(i+1))< -15
            v1(j,(i+1))= -15;
        end
        if v2(j,(i+1))< -15
            v2(j,(i+1))= -15;
        end
        if v3(j,(i+1))< -15
            v3(j,(i+1))= -15;
        end
        if v1(j,(i+1))> 60
            v1(j,(i+1))= 60;
        end
        if v2(j,(i+1))> 60
            v2(j,(i+1))= 60;
        end
        if v3(j,(i+1))> 60
            v3(j,(i+1))= 60;
        end
    end

%Updation of p values
    for j=1:p
        p1(j,(i+1)) = p1(j,i) + v1(j,(i+1));
        p2(j,(i+1)) = p2(j,i) + v2(j,(i+1));
        p3(j,(i+1)) = p3(j,i) + v3(j,(i+1));
    end
    %Pmin and Pmax constraint
    for j=1:p
        if p1(j,(i+1))< 50
            p1(j,(i+1))= 50;
        end
        if p2(j,(i+1))< 30
            p2(j,(i+1))= 30;
        end
        if p3(j,(i+1))< 30
            p3(j,(i+1))= 30;
        end
        if p1(j,(i+1))> 250
            p1(j,(i+1))= 250;
        end
        if p2(j,(i+1))> 100
            p2(j,(i+1))= 100;
        end
        if p3(j,(i+1))> 100
            p3(j,(i+1))= 100;
        end
    end

    %For losses formulation (PL)
```

```
    for j=1:p
        pl(j,(i+1))= [p1(j,(i+1)) p2(j,(i+1)) p3(j,(i+1))]*B*[p1(j,(i+1))
p2(j,(i+1)) p3(j,(i+1))]';
    end

    %Main objective function
    for j=1:p
        f(j,(i+1))= w1*((a(1)*(p1(j,(i+1)))^2 + b(1)*p1(j,(i+1)) + c(1))
+...
                    (a(2)*(p2(j,(i+1)))^2 + b(2)*p2(j,(i+1)) + c(2))+...
                    (a(3)*(p3(j,(i+1)))^2 + b(3)*p3(j,(i+1)) + c(3))) +...
                   w2*pl(j,(i+1)) + k*abs(pd+pl(j,(i+1))-p1(j,(i+1))-
p2(j,(i+1))-p3(j,(i+1)));
    end

%personal best values updation
    %For losses formulation (PL)
    for j=1:p
        plp(j)= [p1p(j) p2p(j) p3p(j)]*B*[p1p(j) p2p(j) p3p(j)]';
    end

    for j=1:p
        fp(j)= w1*((a(1)*(p1p(j))^2 + b(1)*p1p(j) + c(1)) +
(a(2)*(p2p(j))^2 + b(2)*p2p(j) + c(2)) ...
             + (a(3)*(p3p(j))^2 + b(3)*p3p(j) + c(3))) + w2*plp(j) +
k*abs(pd+plp(j)-p1p(j)-p2p(j)-p3p(j));
    end
    for m=1:p
        if f(m,i)< fp(m)
            p1p(m)=p1(m,(i+1));
            p2p(m)=p2(m,(i+1));
            p3p(m)=p3(m,(i+1));
        else
        end
    end

%for Global best values updation
    if min(f(:,(i+1)))<fgm
        fgm=min(f(:,(i+1)));
    else
    end

    for j=1:(i+1)
        for m=1:p
        if f(m,j)==fgm
            for l=1:p
                p1g(l) = p1(m,j);     %global best values
                p2g(l) = p2(m,j);
                p3g(l) = p3(m,j);
            end
        else
        end
        end
    end

    %For cost calculation
    for j=1:p
         c1(j,(i+1)) = a(1)*(p1(j,(i+1)))^2 + b(1)*p1(j,(i+1)) + c(1);
         c2(j,(i+1)) = a(2)*(p2(j,(i+1)))^2 + b(2)*p2(j,(i+1)) + c(2);
         c3(j,(i+1)) = a(3)*(p3(j,(i+1)))^2 + b(3)*p3(j,(i+1)) + c(3);
```

```
          C(j,(i+1)) = c1(j,(i+1)) + c2(j,(i+1)) + c3(j,(i+1));
    end

%To find change in the values of f, equality constraint and change in cost
    for j=1:p
        df(j,i)= abs(f(j,(i+1))-f(j,i)) ;
        sp(j,i)= abs(pd+pl(j,(i+1))-p1(j,(i+1))-p2(j,(i+1))-p3(j,(i+1)));
        csp(j,i)= abs(C(j,(i+1))-C(j,i));
    end

    print = [p1(:,(i+1)) p2(:,(i+1)) p3(:,(i+1)) v1(:,(i+1)) v2(:,(i+1))
v3(:,(i+1)) f(:,(i+1)) c1(:,(i+1)) c2(:,(i+1)) c3(:,(i+1)) C(:,(i+1))];
    disp('      P1          P2         P3         V1          v2         V3
f         c1         c2         c3         C ')
    disp(print)

%Stoping criterion  &&(csp(j,i)<=10^(-6))
    ki=0;
    for j=1:p
        if ((df(j,i)<=10^(-6))&&(sp(j,i)<=10^(-6)))
            ki=ki+1;
        end
    end
    if ki >= p
        break
    end

end

disp(' we have to minimize the cost function of a 3 machine 30 Bus System')
disp(sprintf('No. of particles used in a swarm = %d',p))
disp(sprintf('Max. no. of iterations entered = %d\n',it))

disp(sprintf('Total demand of power Pd = %d \n',pd))


disp('Initial values of generations of 3 generators')
initial=[p1(:,1) p2(:,1) p3(:,1)];
disp('    P1         P2         P3    ')
disp(initial)

disp(sprintf('\nPD+Pl = %d',pd+pl(1,i)))
disp(sprintf('\nP1+P2+P3=%d\n',p1(1,i)+p2(1,i)+p3(1,i)))

disp(sprintf('No. of total Iterations took place = %d \n',i))
disp(sprintf('Total loses in the lines Pl = %d \n',pl(1,i)))
disp(sprintf('Minimum cost incured = %d \n',C(1,i)))
disp('Final values of generations of the three generators')
disp(sprintf('P1=%d',p1(1,i)))
disp(sprintf('P2=%d',p2(1,i)))
disp(sprintf('P3=%d',p3(1,i)))

disp('About this run')
disp('1. The Constraints has been included as absolute value.')
disp('2. Random values between the limits of generation have been taken for
each generator as the different starting point.')
disp('3. Correct values of B coefficients have been fed.')
disp(sprintf('4. K taken = %d',k))
disp(sprintf('5. w1 and w2 taken = %d and %d',w1,w2))
```

REFERENCES

[1] J. Kennedy, R.C. Eberhart, et al., "Particle swarm optimization", In Proceedings of IEEE international conference on neural networks, volume 4, pages 1942–1948. Perth, Australia, 1995

[2] El-Ghazali Talbi, Metaheuristics-From Design to Implementation.: John Wiley and Sons, 2009.

[3] R.C. Eberhart, J. Kennedy, A new optimizer using particle swarm theory, in: Proceedings of the Sixth International Symposium on Micromachine and Human Science, Nagoya, Japan, 1995, pp. 39–43.

[4] Andries P. Engelbrecht, Computational Intelligence: An Introduction.: John Wiley and Sons, 2007, ch. 16, pp. 289-358.

[5] R.C. Eberhart, Y. Shi, Particle swarm optimization: Developments, applications and resources, in: Proceedings of the IEEE Congress on Evolutionary Computation, IEEE Press, Seoul, Korea, 2001.

[6] ` Shi, R.C. Eberhart, A modified particle swarm optimizer, in: Proceedings of the IEEE Congress on Evolutionary Computation, Piscataway, USA, 1998, pp. 69–73.

[7] Anthony Carlisle and Gerry Dozier, "An Off-The-Shelf PSO," in Workshop Particle Swarm Optimization, Indianapolis, 2001.

[8] Y. Shi and R. Eberhart., "A modified particle swarm optimizer", In Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on, pages 69–73. IEEE, 2002.

[9] R.C. Eberhart and Y. Shi., "Tracking and optimizing dynamic systems with particle swarms", In Evolutionary Computation, 2001. Proceedings of the 2001 Congress on, volume 1, pages 94–100. IEEE, 2002.

[10] A.Nikabadi, M.Ebadzadeh , "Particle swarm optimization algorithms with adaptive Inertia Weight : A survey of the state of the art and a Novel method", IEEE journal of evolutionary computation , 2008

[11] R.F. Malik, T.A. Rahman, S.Z.M. Hashim, and R. Ngah, "New Particle Swarm Optimizer with Sigmoid Increasing Inertia Weight", International Journal of Computer Science and Security (IJCSS), 1(2):35, 2007.

[12] J.J. Jamian, M.W. Mustafa, H. Mokhlis, M.N. Abdullah" Comparative Study on Distributed Generator Sizing Using Three Types of Particle Swarm Optimization" 978-0-7695-4668-1/12 $26.00 © 2012 IEEE

[13] Chien-Ching Chiu, Chi-Hsien Sun&Chun-Fu Li "Comparison of Dynamic Differential Evolution and Asynchronous Particle Swarm Optimization for Inverse Scattering" ICCIT 2012.

[14] J. Xin, G. Chen, and Y. Hai., "A Particle Swarm Optimizer with Multistage Linearly-Decreasing Inertia Weight", In Computational Sciences and Optimization, 2009. CSO 2009. International Joint Conference on, volume 1, pages 505–508. IEEE, 2009.

[15] Y. Feng, G.F. Teng, A.X. Wang, and Y.M. Yao., "Chaotic Inertia Weight in Particle Swarm Optimization", In Innovative Computing, Information and Control, 2007. ICICIC'07. Second International Conference on, page 475. IEEE, 2008.

[16] K. Kentzoglanakis and M. Poole., "Particle swarm optimization with an oscillating Inertia Weight", In Proceedings of the 11th Annual conference on Genetic and evolutionary computation, pages 1749– 1750. ACM, 2009.

[17] M.S. Arumugam and MVC Rao., "On the performance of the particle swarm optimization algorithm with various Inertia Weight variants for computing optimal control of a class of hybrid systems", Discrete Dynamics in Nature and Society, 2006, 2006.

[18] Y. Shi and R.C. Eberhart., "Empirical study of particle swarm optimization", In Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on, volume 3. IEEE, 2002.

[21] H.R. Li and Y.L. Gao., "Particle Swarm Optimization Algorithm with Exponent Decreasing Inertia Weight and Stochastic Mutation", In —2009 Second International Conference on Information and Computing Science, pages 66–69. IEEE, 2009.

[22] Y. Gao, X. An, and J. Liu., "A Particle Swarm Optimization Algorithm with Logarithm Decreasing Inertia Weight and Chaos Mutation", In Computational Intelligence and Security, 2008. CIS'08. International Conference on, volume 1, pages 61–65. IEEE, 2008.

[23] James Kennedy and Tim Blackwell Riccardo Poli, "Particle swarm optimization An overview," Swarm Intelligence, vol. 1, no. 1, pp. 33–57, 2007.

[24] Dr. N.K. Jain, Ms. Uma Nangia, Dr. C.L. Wadhwa "Investigaion on Multiobjective Optimal Load Flow Study by Sequential Goal Programming" IE(I) Journal-EL Vol 77, August 1996.

[25] Dr. N.K. Jain, Ms. Uma Nangia, Dr. C. L. Wadhwa "Multiobjective Optimal Load Flow Based on Ideal Distance Minimization In 3D Space" Electrical Power and Energy Systems 23(2001) 847-855 (ELSEVIER)

[26] Singiresu S. Rao "Engineering Optimization Theory and Practice (Third Enlarged Edition)" New Age International Publishers ISBN:978-81-224-2723-3.

[28] M.A. Abido "Environmental/Economic Power Dispatch Using Multiobjective Evolutionary Algorithms" IEEE Transactions on Power Systems Vol. 18. No.4 ,November 2003.

[29] Nangia, U., N.K. Jain , C.L. Wadhwa, 'Multiobjective optimal load flow based on ideal distance minimization in 3D space', Electrical power and energy systems (23), 2001,pp. 847-855

[30] Nangia, Uma, Jain, N.K., Wadhwa, C.L., 'Comprehensive Comparison of Various Multiobjective Techniques', Engineering Intelligent Systems Journal, Vol. 11, No. 3, Sept.,2003, pp. 123-132.