# ECONOMIC LOAD DISPATCH USING IMPROVED PARTICLE SWARM OPTIMIZATION

DISSERTATION
SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE AWARD OF THE DEGREE
OF
MASTER OF TECHNOLOGY
IN
POWER SYSTEM

Submitted by:
NIMISH KUMAR
2K12/PSY/12

Under the supervision of
Prof. Uma Nangia & Prof. N.K. Jain


**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, NIMISH KUMAR, Roll No. 2K12/PSY/12 student of M. Tech. (Power System), hereby declare that the dissertation titled "ECONOMIC LOAD DISPATCH USING IMPROVED PARTICLE SWARM OPTIMIZATION" under the supervision of Prof. Uma Nangia & Prof. N.K. Jain 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: 30.07.2014

**NIMISH KUMAR**
2K12/PSY/12
M.Tech (PSY)

**Prof. Uma Nangia**
Professor
EE Dept., DTU

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

# ACKNOWLEDGEMENT

The writing of this dissertation has been one of the most significant academic challenges I have ever had to face; without GOD's blessing and support, patience and guidance of the following people, this study would not have been completed. It is to them I owe my deepest gratitude.

- **Prof. N.K. Jain** & **Prof. Uma Nangia,** Electrical Engineering Department, DELHI TECHNOLOGICAL UNIVERSITY (Formerly Delhi College of Engineering) for their initiative and valuable guidance in this field of research, for their affection and encouragement for the successful completion of this work. Their kind attitude and sincere sympathies always encouraged me to carry out the present work firmly.
- **Prof. Madhusudan Singh,** HOD, Electrical Engineering Department and **Prof. Narendra Kumar,** Ex-HOD, Electrical Engineering Department, DELHI TECHNOLOGICAL UNIVERSITY (Formerly Delhi College of Engineering), New Delhi, for providing me with the best facilities in the department.
- My parent, **Niranjan Kumar**, my elder brother and **Sanjeev Prabhakar**, my younger brother who have always supported, encouraged and believed in me and patiently waited for my dreams to come true.

NIMISH KUMAR
2K12/PSY/12
M.TECH. (PSY)

# ABSTRACT

Optimization is a mathematical tool to find the maximum or minimum of a function in some feasible region. There is no any industry which not involved the solution of optimization problems. In the operational planning of power system, Economic load dispatch (ELD) is a common task which concern with the optimization problems. The objective of ELD problem is to schedule the output of the connected units of the plant so as to fulfil the load demand at minimum operating cost while satisfying all operational constraints. Recently particle swarm optimization algorithm inspired by collective behaviour of swarm has been applied successfully to solve ELD problem.

In this work three improved PSO algorithms- IPSO-A, IPSO-B and IPSO-C have been developed and implemented to solve ELD for IEEE 5, 14 and 30 bus systems. Conventional PSO (CPSO) using inertia weight and constriction factor individually as well as simultaneously have been also implemented to solve ELD problem. PSO algorithms have been compared for twenty trial runs. The best, worst, average and standard deviation cost for all the algorithms have been determined. The results show that proposed improved PSO techniques gives the optimum operational cost with consistent result.

# CONTENTS

# List of Figures

# List of Tables

# List of Symbols & abbreviations

| | |
|---|---|
| $C_1$ | Acceleration constant for Cognitive component |
| $C_2$ | Acceleration constant for Social component |
| $\chi$ | Constriction Factor |
| w | Inertia Weight |
| $It_{max}$ | Maximum no. of iteration |
| p | Number of particles |
| K | Penalty coefficient |
| $\varepsilon$ | Tolerance Limit |
| $r_1$ & $r_2$ | Uniformly distributed Random Numbers (0,1) |
| CF | Constriction Factor |
| CPSO | Conventional Particle Swarm Optimization |
| ELD | Economic Load Dispatch |
| IPSO | Improved Particle Swarm Optimization |
| IW | Inertia Weight |
| PSO | Particle Swarm Optimization |

# *CHAPTER 1*

# *INTRODUCTION*

## 1.1 OVERVIEW

In Economic Load Dispatch (ELD) Problem we determine the optimal combination of power output of all the connected generating units of the plant to minimize the total production (fuel or operational) cost while fulfilling the load demands and system operational constraints. Proper planning of connected unit outputs can contribute to considerable saving in the plant operating cost. Recently particle swarm optimization algorithms inspired by collective behaviour of swarm has been applied successfully to solve ELD problem. The popularity of PSO is due to its simple concept and easy implementation to both linear as well as non-linear problems.

In this work an attempt has been made to solve ELD problem by using constriction factor and inertia weight individually and simultaneously in the Conventional PSO (CPSO). Various improved PSO (IPSO) algorithm- IPSO-A, IPSO-B and IPSO-C have been developed based on the movement of particles. In IPSO-A, the position and velocity of the particles is not updated when a particle hits the global during the search. In IPSO-B, the velocity of all the particles is updated, but the position of particles achieved global is kept fixed. In IPSO-C, the position of the particles whose current position is better than its personal best is not updated. All the above algorithms have been implemented on IEEE 5, 14, 30 bus systems. Detailed comparisons of all the above algorithms have been carried out for 20 trials. The best, worst and average fuel costs have been determined for each algorithm.

## 1.2 OBJECTIVE AND METHODOLOGY

The objective of this work is to solve economic load dispatch (ELD) problem considering equality and inequality constraints to satisfy the consumers demand. The standard IEEE 5, 14, 30 bus system have been consider for planning of connected unit outputs to contribute considerable saving in the plant operating cost.

Three improved PSO algorithms namely IPSO-A, IPSO-B and IPSO-C have been developed and implemented to solve ELD problem. Conventional PSO (CPSO) using inertia weight and constriction factor individually as well as simultaneously have been also implemented to solve ELD problem. The best, worst, average fitness and their standard deviation for all the algorithms have been determined.

The work has been carried out in the following order:

a. Exploring Particle Swarm Optimization in detail.
b. Solving some mathematical function using basic PSO by hand calculation.
c. Coding the PSO algorithms in MATLAB 2012a.
d. Solution of various mathematical benchmark functions using PSO algorithms by MATLAB 2012a.
e. Formulation of objective function for ELD problem considering cost of generation and penalized demand constraint for IEEE 5,14 & 30 bus system.
f. Solution of ELD problem using PSO algorithms by MATLAB 2012a.
g. Detailed analysis of different PSO algorithms has been done.
h. Conclusions and future directions have been carried out.

# 1.3 LITERATURE REVIEW

## 1.3.1 ECONOMIC LOAD DISPATCH

Scheduling of connected generating units of the plant plays an important role in the power system operation, planning and control. Proper planning of connected unit outputs can contribute to considerable saving in the plant operating cost. A number of techniques have been successfully applied to solve ELD problem during past years [1]-[19]. Some of these use the conventional optimization techniques, whereas others are based on intelligent techniques.

The examples of conventional techniques to solve ELD problems are Lambda iteration method, Fast lambda method, Base point and Participation factors method and Gradient method [1]-[3]. Chem-Lin Chen and Shun-Chung Wang has solved the ELD problem by implementing new branch-and-bound algorithm in conventional technique [1]. The gradient projection method has been used by K.Y.

Lee et. al to solve ELD problem for active and reactive power [2]. The Fast lambda method has been developed by J. P. Zhan et. al to solve ELD problem With Prohibited Operating Zones [3].

All these conventional techniques have limitation on the nature of cost curves. In addition, these techniques have oscillatory problems due to existence of several local minima in the ELD problems with large number of connected units in the systems. Due to complex algorithm of conventional techniques, it takes high computational time.

Modern stochastic techniques such as Linear programming, Quadratic programming, Genetic algorithm, Biogeography based optimization, Chemical reaction optimization, Enhanced Bee swarm optimization, Modified Teacher learning algorithm have been employed successfully to solve the ELD problems [4]-[10]. Rabih A. Jabr et. al presented homogeneous linear programming algorithm for the solution of security constrained ELD problem[4]. Leandro dos Santos Coelho and Viviana Cocco Mariani used combination of chaotic differential evolution and Quadratic Programming for the solution of ELD problem considering the valve point effect [5]. ELD problem with network constraints have been solved Ioannis G. Damousis et. al using real coded genetic algorithm [6]. Biogeography-Based Optimization algorithm has been used by Aniruddha Bhattacharya and Pranab Kumar Chattopadhyay to solve both convex and non-convex ELD Problems [7]. Chemical reaction optimization algorithm has been used by Kuntal Bhattacharjee et. al to solve ELD problems considering different constraints [8]. Enhanced Bee Swarm Optimization Algorithm has been proposed by Taher Niknam and Faranak Golestaneh for the solution of dynamic ELD problems [9]. Taher Niknam et. al introduces a new optimization algorithms named as Modified Teacher learning algorithm for dynamic ELD problem considering reserved constraints [10]. The hybrid of different modern techniques has also been used in ELD problems. Irina Ciornei and Elias Kyriakides proposed the Hybrid genetic algorithm with ant colony optimization to solve the non-convex ELD problem [11]. Aniruddha Bhattacharya and Pranab Kumar Chattopadhyay have presented hybrid differential evolution with biography based optimization to solve ELD problems [12].

These intelligent optimization techniques do not suffer from any limitation on the nature of cost curve, due to their ability to find the optimal solution. But, these

methods have large number of parameters involved in the algorithm, and takes large number of iterations to settle to the global optimum.

The ELD problems have been solved recently by Particle Swarm Optimization (PSO) approaches [13]-[17]. The solution of ELD problem with non-smooth cost function using PSO technique has been described by Jong-Bae Park et. al [13]. The chaotic and Gaussian PSO approach has been described by Leandro dos Santos Coelho and Chu-Sheng Lee to solve ELD problem for better performance of PSO [14]. A new concept 'θ-PSO' for solving ELD problem is developed by Vahid Hosseinnezhad and Ebrahim Babaei [15]. The ELD with environmental emission have been solved by M.A. Abido using PSO [16]. The quantum inspired PSO has been proposed to solve ELD problem for faster convergence by Ke Meng et. al [17]. The hybrids PSO are also used to solve the ELD problems [18]-[19]. The hybrid of binary PSO (for unit commitment) and real coded PSO (for ELD) has been described by T. O. Ting et. al [18]. The hybrid quantum inspired PSO has been proposed by S. Chakraborty et. al to explore search space for solving ELD problem [19].

## 1.3.2: PARTICLE SWARM OPTIMIZATION

The Particle Swarm Optimization (PSO) is a population based stochastic algorithm, developed by Kennedy and Eberhart in 1995, inspired by the collective behaviour of nature such as bird flocking, fish schooling etc. [20]-[29]. A new concept based upon particle swarm methodology for optimization of non-linear function is introduced by James Kennedy and Russell Eberhart and relationships between PSO with artificial life and genetic algorithms have also been described [20]. Ioan Cristian Trelea has described the detailed analysis of PSO algorithms, its parameter selection and convergence characteristics [21]. Constriction function is introduced by Maurice Clerc and James Kennedy in traditional PSO to eliminate the use of particle's velocities to limit in order to control the particle's trajectories and detailed analysis of constriction factor in the PSO update equation has been carried out [22]. A new parameter namely inertia weight has been introduced in PSO by Yuhui Shi and Russell Eberhart to find global optimum within a reasonable number of iteration and time decreasing Inertia weight has been also introduced to improve the performance of PSO [23]. The Comparison of inertia weight and constriction factor and suitable conditions for the use of these two parameters has been also

explained by R. C. Eberhart and Y. Shi. and it has been shown that the application of both parameters simultaneously results in faster convergence overall [24]. The multi-swarm is used by Frans van den Bergh and Andries P. Engelbrecht to improve the performance of traditional PSO based upon the cooperative behaviour [25]. The efficient optimization for multi-model function namely multi-grouped PSO have been proposed by Jang-Ho Seo et. al and its application to various areas including particle electromagnetic optimization have been discussed [26]. J. J. Liang et. al have presented Comprehensive learning PSO base on learning strategy of particles to update particle's velocity, which reduces premature convergence [27]. Multi-objective Optimization problem using modern PSO for optimal operation of plant have been carried out by Jin S. Heo et. al and PSO technique variation, evolutionary PSO, hybrid PSO, constriction approach PSO and comparison of PSO and genetic algorithms have also been discussed [28]. Basic concept of PSO, its variants and its application to power system and another area of electrical engineering have discussed by Yamille del Valle et. alin detailed [29].

## 1.4 ORGANISATION OF THE THESIS

This thesis has been arranged in seven chapters. The contents of the chapters are briefly outlined here.

Chapter 1: This chapter describes the overview, objective and methodology, literature review and the organisation of the thesis.

Chapter 2: This chapter discuss about economic load dispatch in detail. The cost function, constraints and formation of objective function for ELD problem have been explained.

Chapter 3: This chapter gives detailed idea about conventional PSO. It involves development, advantages, disadvantages and the application of PSO.

Chapter 4: In this chapter Improved PSO algorithms have been introduced for the improvements in the performance of the Conventional PSO.

Chapter 5: In this chapter some mathematical benchmark functions have been solved by convention as well as improved PSO algorithms. The steps for MATLAB programming have been also discussed.

Chapter 6: In this chapter IEEE 5, 14 and 30 bus systems have been solved by different PSO algorithms and detailed discussions on the results have been also entertained.

Chapter 7: This chapter deals the outcome of the results and the future scope of this research work.

Appendixes and References are given at the end of the thesis.

# *CHAPTER 2*

# *ECONOMIC LOAD DISPATCH*

## 2.1 INTRODUCTION

In the field of power system analysis, Economic load dispatch is one of the most fundamental optimization problems. In ELD mainly two different problems, one is unit commitment or pre dispatch problem and other is online economic dispatch problem have been solved. The unit commitment is the way to suggest just sufficient number of generating units with sufficient generating capacity to meet a given load economically with sufficient reserve capacity to meet any abnormal condition. The on line economic dispatch problem distribute the load demand among the connected generating units of the plant in such a manner to minimize the total cost of production supplying the minute to minute requirements of the system [30].

Particularly, in thermal power plants the cost of generation is too excessive, hence significant saving in operating cost can be achieved by proper planning of units output of the plant. The ELD describes the optimal scheduling of power generations to match total power demand at minimal possible cost while satisfying the operational constraints i.e. generators and system constraints.

## 2.2 COST FUNCTION

The main purpose of ELD is to determine the power generation of each unit of the plant so that total production costs of the plant should be minimum by fulfilling the required power demand under the given equality and inequality constraints.

The production costs of each unit are generally expressed by a quadratic function of the power output from those generating units. The total production costs of the plant are the sum of production cost of each individual units of the plant.

Mathematically,

$$F_i(P_i) = a_i P_i^2 + b_i P_i + c_i \qquad (2.1)$$

Where $F_i(P_i)$, $P_i$, $(a_i,b_i,c_i)$ are the production cost, power generation ,cost coefficient of ith unit of the plant respectively.

Therefore total production cost of the plant having n units,

$$F_T = \sum_{i=1}^{n} ( F_i(P_i) )$$

$$= \sum_{i=1}^{n} ( a_i P_i^2 + b_i P_i + c_i ) \qquad (2.2)$$

The cost is minimized subjected to the system constraints (generator capacities and active power balance constraints).

## 2.3 SYSTEM CONSTRAINTS

There are two types of system constraints-

i) Equality constraints

ii) Inequality constraints

The inequality constraints are of two types-

i) Hard type

ii) Soft type

The hard type are those which are definite and specific like the tapping range of an on-load tap changing transformer whereas soft type are those which have some flexibility associated with them like the nodal voltages and phase angles between the nodal voltages, etc. Soft inequality constraints have been very efficiently handled by penalty function methods.

## 2.3.1 EQUALITY CONSTRAINTS

From observation we can conclude that cost function is not affected by the reactive power demand. So the full attention is given to the real power balance in the system. For power mismatch, an equality constraint has been introduced i.e. the generated power by the plant should be equal to the total load demand plus the total losses. Thus the power balance equation of ELD problems is given by,

$$P_G - P_D - P_L = 0$$

$$\sum_{i=1}^{n} P_i - P_D - P_L = 0 \qquad (2.3)$$

Where $P_G$ is the total power generation of the plant, $P_D$ is the total power (load) demand by the consumer and $P_L$ is the total losses during process.

The transmission losses can be determined from unit outputs of the plant and loss coefficients as,

$$P_L = \sum_{i=1}^{n} ( \sum_{j=1}^{n} ( P_i B_{ij} P_j)) + \sum_{i=1}^{n} (B_{i0} P_i) + B_{00} \qquad (2.4)$$

Where $B_{ij}$ is the ij-th element of the loss coefficient square matrix, $B_{i0}$ is the i-th element of the loss coefficient vector, and $B_{00}$ is the loss coefficient constant.

## 2.3.2 INEQUALITY CONSTRAINTS

### i) Generator Constraints-

The KVA loading in a generator is given by $\sqrt{(P^2 + Q^2)}$ and this should not exceed a pre-specified value of power because of the temperature rise conditions

- When the active power generation is greater than pre-specified value $P_{max}$ then the source become overheated i.e. thermal consideration limits the maximum active power generation of the source. And if the power generation is less than pre-specified value $P_{min}$ then from the optimal point of view it is not possible to generate that low power. In other words minimum power generation of the source is limited by the flame instability of the boiler and it is not put on the bus bar. Therefore the active power generation P must be within the maximum and minimum limit, stated by the inequality,

  $$P_{min} \leq P \leq P_{max}$$

- Similarly the maximum and minimum reactive power generation of a source is limited. The maximum reactive power is limited because of overheating of rotor and minimum is limited because of the stability limit of machine. Hence the generators reactive powers Q cannot be outside the range stated by inequality,

  $$Q_{min} \leq Q \leq Q_{max}$$

## ii) Voltage Constraints:

It is essential that the voltage magnitudes and phase angles at various nodes should vary with in certain limits. The normal operating angle of transmission lies between 30 to 45 degrees for transient stability reasons. A lower limit of delta assures proper utilization of transmission capacity.

## iii) Running Spare Capacity Constraints:

These constraints are required to meet

a) The forced outages of one or more alternators on the system and

b) The unexpected load on the system

The total generation should be such that in addition to meeting load demand and losses a minimum spare capacity should be available i.e.

$$G \geq P_G + P_{SO}$$

Where G is the total generation and $P_{SO}$ is some pre-specified power. A well planned system is one in which this spare capacity $P_{SO}$ is minimum.

## iv) Transmission Line Constraints:

The flow of active and reactive power through the transmission line circuit is limited by the thermal capability of the circuit and is expressed as,

$$Cp \leq Cp_{max}$$

Where $Cp_{max}$ is the maximum loading capacity of the line.

## v) Transformer Taps Settings:

If an auto-transformer is used, the minimum tap setting could be zero and the maximum one,

$$\text{i.e. } 0 \leq t \leq 1.0$$

Similarly for a two winding transformer if tapping are provided on the secondary side,

$$0 \leq t \leq n$$

Where n is the ratio of transformation.

vi) Network security constraints:

If initially a system is operating satisfactorily and there is an outage, may be scheduled or forced one, it is natural that some of the constraints of the system will be violated. The complexity of these constraints (in terms of number of constraints) is increased when a large system is under study. In this a study is to be made with outage of one branch at a time and then more than one branch at a time. The natures of constraints are same as voltage and transmission line constraints.

## 2.4 OBJECTIVE FUNCTION

The Objective (fitness) function of ELD problem is defined to minimize the sum of the cost of generation function given by equation (2.2) and the penalized demand (equality) constraint given by equation (2.3) as follows:

Minimize the fuel cost,

$$F = \sum_{i=1}^{n} ( a_i P_i^2 + b_i P_i + c_i ) + K*( \sum_{i=1}^{n} P_i - P_D - P_L ) \qquad (2.5)$$

Subjected to generators constraints.

Where K is the penalty coefficient for the plant due to not fulfilling the load demands to the consumers and chosen carefully for the feasible solution. Somewhere K is also termed as Lagrangian multiplier.

# *CHAPTER 3*

# *CONVENTIONAL PARTICLE SWARM OPTIMIZATION*

## 3.1 INTRODUCTION

The Particle Swarm Optimization (PSO) is a population based stochastic algorithm, developed by of Kennedy (a social psychologist) and Eberhart (an electrical engineer) in 1995, inspired by the collective behaviour of nature such as bird flocking, fish schooling etc., to successfully optimize a wide range of linear as well as non-linear functions. The first simulations was done by Kennedy and Eberhart (in 1995), which were influenced by Heppner and Grenander's work (in 1990) and involved analogues of bird flocks searching for corn.

The PSO belongs to the class of swarm intelligence techniques that are used to find approximate solutions to extremely difficult optimization problems. Swarm intelligence is an artificial intelligence technique based around the study of collective behaviour in decentralized, self-organized systems. Examples of systems like this can be found in nature, including bird flocking, fish schooling, ant colonies, animal herding etc.

(a) Bird Flocking (b) Fish Schooling

**Fig. 3.1: Collective behaviour of Swarm**

Craig Reynolds (a biologist) studied the evolutionary algorithms and swarm intelligence for optimization inspired by the social behaviour of birds in late 80s and early 90s and derived a formula for representation of the birds flocking behaviour. This was later used in computer simulations of virtual birds, known as Boids.

Reynolds proposed a behavioural model in which each agent follows three rules:

- Separation- Each agent tries to move away if they are too close.
- Alignment- Each agent steers towards the average heading of its neighbours.
- Cohesion- Each agent tries to go towards the average position of its neighbours.

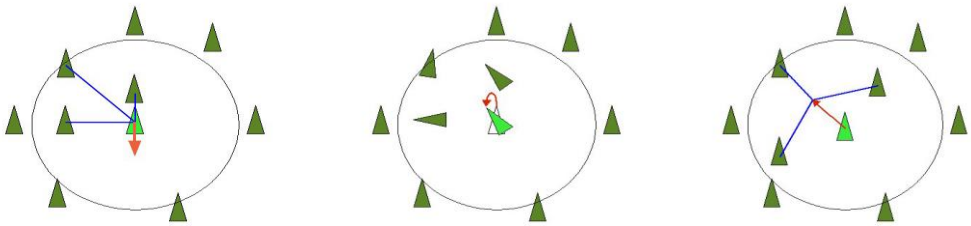

**Fig. 3.2: Reynolds proposed behaviour model**

Kennedy and Eberhart included a 'roost' in a simplified Reynolds-like simulation so that:

- Each agent was attracted towards the location of the roost.
- Each agent 'remembered' where it was closer to the roost.
- Each agent shared information with its neighbours about its closest location to the roost.

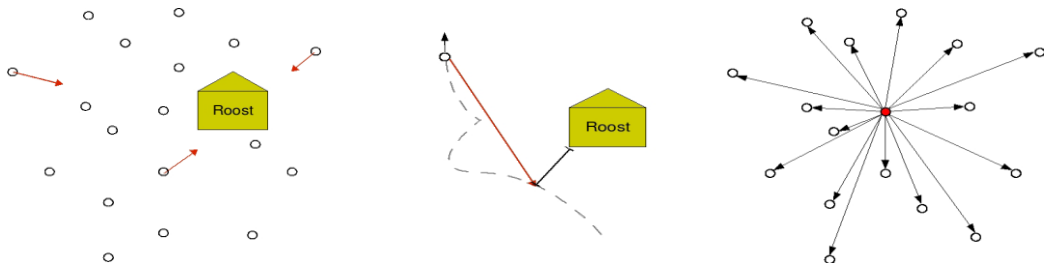


**Fig. 3.3: Kennedy and Eberhart proposed behaviour model**

Eventually, all agents land on the roost.

The particle swarm optimization (PSO) is a parallel evolutionary computation technique inspired by the social behaviour of natural process to successfully optimize a wide range of continuous linear as well as non-linear functions [20]-[24]. The PSO algorithm involves a population of candidate solutions (called particles), collectively called swarm. Each particle of the swarm is initialized with a random position and random velocity and is moved iteratively throughout the search space. It is attracted towards the location of both its personal best fitness achieved by the particle itself and best fitness achieved by the entire swarm so far [21]. Positions and velocities of whole swarm are adjusted and the function evaluated with the updated coordinates at each time step. The velocity and position of each particle is updated by two given rule in Conventional PSO (CPSO) as,

Velocity modification rule,

$$V_{ij}^{t+1} = V_{ij}^{t} + c_1 r_1 (P_{ij}^{best} - X_{ij}^{t}) + c_2 r_2 (G_i^{best} - X_{ij}^{t}) \qquad (3.1)$$

Position modification rule,

$$X_{ij}^{t+1} = X_{ij}^{t} + V_{ij}^{t+1} \qquad (3.2)$$

Where t indicates the number of iteration, i indicates the number of variables, j indicates the number of particles in the swarm, $V_{ij}^{t}$ is the velocity of j-th particle of i-th variable in t-th iteration, $X_{ij}^{t}$ is the position of j-th particle of i-th variable in t-th iteration, $P_{ij}^{best}$ is the previous personal best position of the j-th particle of i-th variable, $G_i^{best}$ is the previous global best position of i-th variable, $V_{ij}^{t+1}$ is the updated velocity of j-th particle of i-th variable, $X_{ij}^{t+1}$ is the updated position of j-th particle of i-th variable, $c_1$ is the cognitive component of acceleration constants which is responsible for the attraction of the particles towards its personal best position, $c_2$ is the social component of acceleration constants which is responsible for the attraction of the particles towards the global best position of the swarm, $r_1$ & $r_2$ are two uniformly distribution random numbers in the range (0,1).

# 3.2 PSO PARAMETERS

There are some parameters in PSO algorithm that may affect its performance [21]. The parameter's value and its selection have large impact on the performance of the PSO algorithm. The basic PSO parameters are swarm size, number of iterations, velocity components, and acceleration coefficients which are described below. The PSO algorithm is also influenced by velocity clamping, constriction factor and inertia weight which are described in the next section.

**i. Swarm size**

Swarm (population) size is the number of particles '*n'* in the swarm. A big swarm i.e. large number of particles cover larger part of the search space and may reduce the number of iterations required for optimum result. It also increases computational efforts per iteration and consuming more time. Whereas a small swarm i.e. less number of particles loss their search ability and may not give optimum result. Therefore proper selection of swarm improves the performance of PSO. In most cases the swarm size is taken between 20 to 100 for better results and computational efforts.

**ii. Iteration numbers**

The number of iterations is an important factor in the PSO algorithms. It is problem dependent to obtain a good result. A too low number of iterations may not give optimum results and too large number of iterations may add unnecessary computational effort and more time.

**iii. Velocity Components**

The velocity components are the main part of the PSO algorithms and important for updating the particle's velocity. The particle's velocity modification equation consists of three terms-

1. The first term '$V_{ij}^{t}$' is called the inertia or momentum component of the velocity that provides a track of the previous direction i.e. movement in the immediate

past. It prevents to a drastically change in the direction of the particles and forced towards the movement of current direction.

2. The second term '$c_1r_1(P_{ij}^{best} – X_{ij}^{t})$' is called cognitive or personal component that tracks or compares the current performance of the particles with its past performances. It is basically the individual memory of the particles and indicates the tendency to return to past best position.

3. The last term '$c_2r_2(G_i^{best} - X_{ij}^{t})$' is called social or collective component that compares the current performance of the particles with the past best performance of any particles in the group. It forces the particles towards the best position tracked by the entire swarm.

**iv. Acceleration coefficients**

The acceleration coefficients $c_1$ and $c_2$, together with the random values $r_1$ and $r_2$, maintain the stochastic influence of the cognitive and social components of the particle's velocity respectively. The constant $c_1$ expresses how much confidence a particle has in itself, while $c_2$ expresses how much confidence a particle has in its neighbours. There are some properties of $c_1$ and $c_2$ –

- When $c_1 = c_2 = 0$, then all particles continue flying at their current speed until they hit the search space's boundary. Therefore, the velocity update equation is calculated as-

$$V_{ij}^{t+1} = V_{ij}^{t}$$

- When $c_1 > 0$ and $c_2 = 0$ , all particles are independent. The velocity update equation will be-

$$V_{ij}^{t+1} = V_{ij}^{t} + c_1r_1(P_{ij}^{best} – X_{ij}^{t})$$

- When $c_1 = 0$ and $c_2 > 0$ , all particles are attracted to a single point in the entire swarm and the update velocity will become-

$$V_{ij}^{t+1} = V_{ij}^{t} + c_2r_2(G_i^{best} - X_{ij}^{t})$$

- When $c_1 = c_2$, all particles are attracted towards the average of $P_{ij}^{best}$ and $G_i^{best}$.

- When $c_1 > c_2$ , each particle is more strongly influenced by its personal best position, resulting in excessive wandering.

- When $c_1 < c_2$, all particles are much more influenced by the global best position, which causes all particles to run prematurely to the optima.

It has been proposed that the two acceleration constants should be $c_1 = c_2 = 2$ for better results [29].

# 3.3 GEOMETRICAL ILLUSTRATION OF PSO

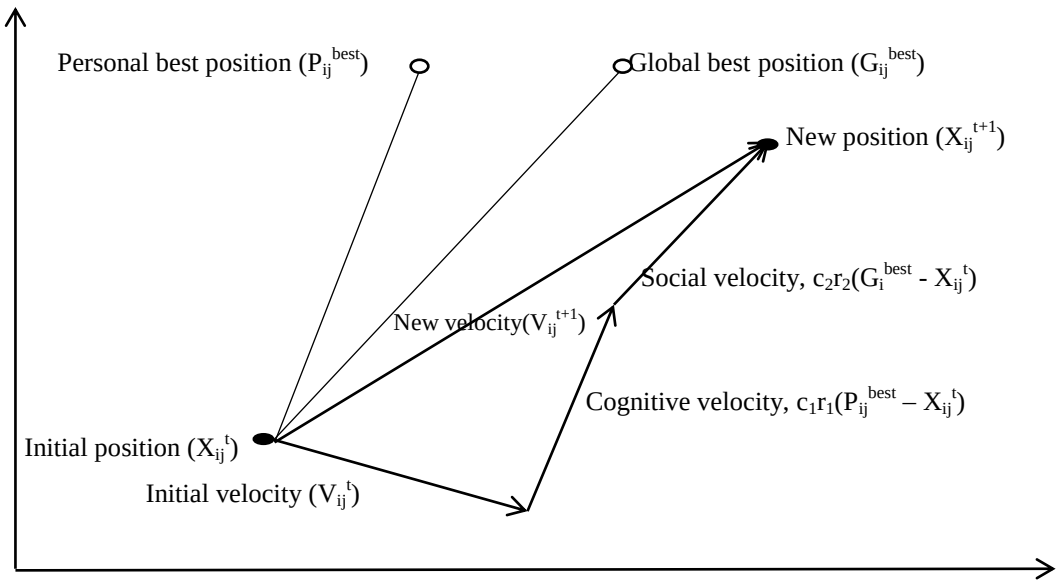


**Fig. 3.4: Velocity and position update for the particles**

The updated velocity consists of three components discussed earlier. Figure 3.4 shows how the three components contribute to the movement of the particle towards the global best position. The initial velocity, cognitive component of velocity and social component of velocity contributes to the new velocity for the particles. This new velocity is responsible for the movement of the particles.

# 3.4 IMPLEMENTATION OF PSO ALGORITHM

The PSO algorithm are summarized by the following steps-

step 1) Initialization of the parameters- In the first step, the parameters of the PSO, like $c_{1,}$ $c_2$, $r_1$, $r_2$, population size P, maximum iteration $It_{max}$ must be chosen.

step 2) Initialization of the population- Initialize a population of particles with a random positions and random velocities.

step 3) Evaluation of Fitness- Calculate the fitness value of the particles in the population.

step 4) Selection of initial personal and global best position- Each initial value of particle is set to be its personal best and optimum of the personal bests is set to be global best.

step 5) Updation of velocity and position- Change the velocities and positions of the particles according to velocity modification rule (3.1) and position modification rule (3.2) respectively.

step 6) Updation of personal and global best position- If positions of the particles give the better fitness value during current iteration then set this as $P^{best}$ and the optimum of $P^{best}$ is set to be $G^{best}$.

step 7) Stopping Criteria- Check the difference in fitness value in the consecutive iteration is less than tolerance limit. If yes stop, otherwise go to step 5.

The PSO algorithm is best understand by the following flow chart-

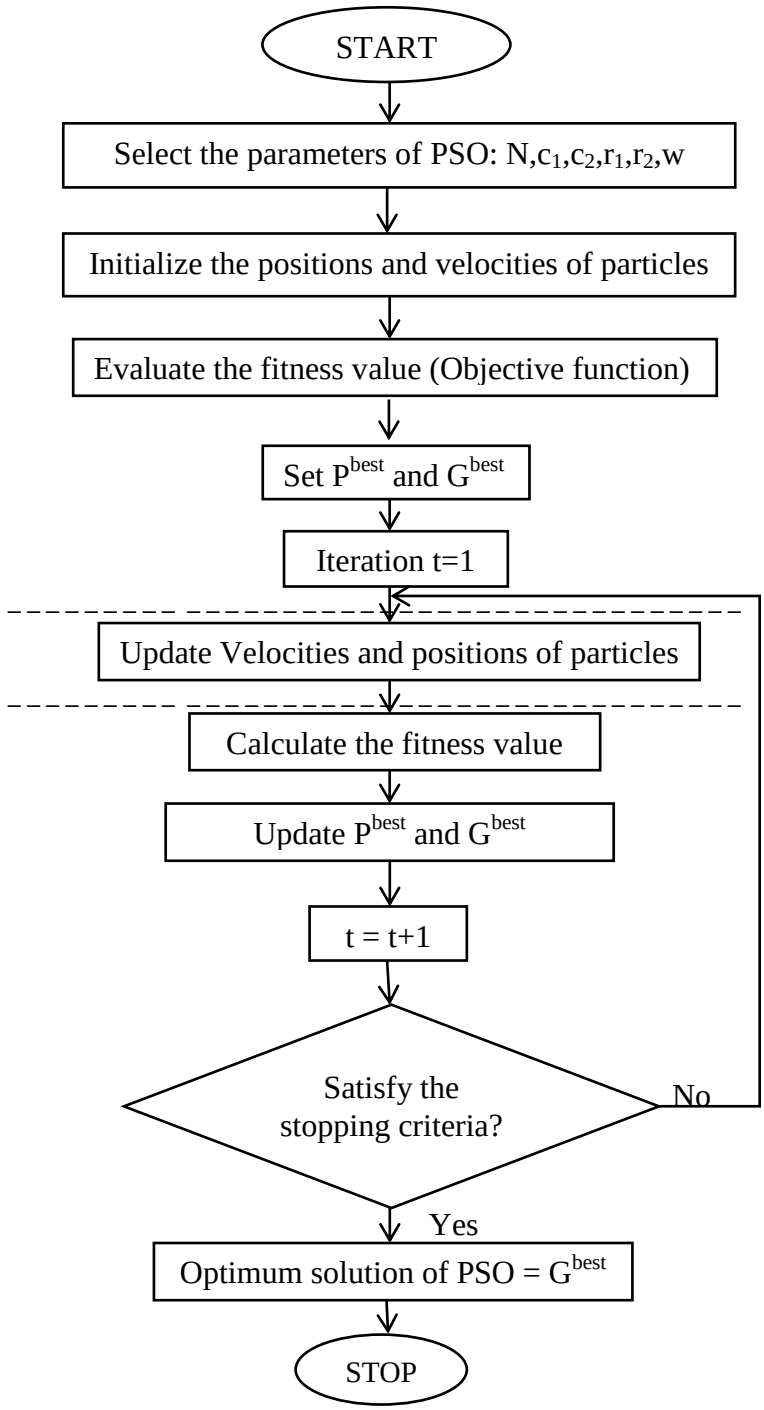


**Fig. 3.5: Flow chart of conventional PSO**

# 3.5 IMPROVEMENTS IN THE CONVERGENCE RATE OF CPSO

The particle velocity is basically the step size of the particle for its movement in the search space. Therefore particle's velocity plays an important role in the performance of PSO. At each step, all the particles adjust its velocity and moves in the search space. Exploration and Exploitation are two characteristics of PSO for searching the best position in the search space. Exploration is the ability to explore different area of the search space for locating a good optimum, while exploitation is the ability to concentrate the search around a searching area for refining a hopeful solution [21]. The particle's position updates quickly, if the velocity increases to large values and the particles may cross the boundaries of the search space. Therefore the velocity and position of the particle are reduced in order to stay within boundary of the search space to control its divergence. To balance the exploration-exploitation trade-off, following techniques for the improvement of speed of convergence have been developed.

## 3.5.1 VELOCITY CLAMPING

The velocity clamping has been first introduced by Eberhart and Kennedy to stay the particles within the boundary of the search space. The Maximum velocity '$v_{max}$' controls the granularity of the search space by clamping velocities and creates a better balance between global exploration and local exploitation [20].

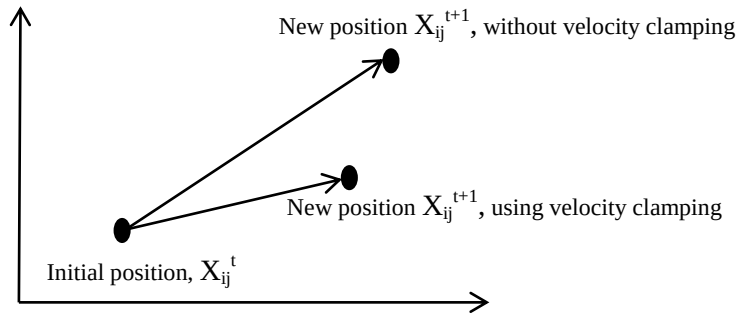


**Fig. 3.6: Effects of Velocity Clamping in two-dimensional search space**

Figure 3.6, shows how velocity clamping resists the particle to stay within the boundary. Now if a particle's velocity goes beyond '$v_{max}$' then it is set to the value

$v_{max}$. If the $v_{max}$ is too large, then the particles may jump over the optimal solution and if is too small, the particle's movement is limited and the swarm may not explore sufficiently or the swarm may become trapped in a local optimum.

By proper selection of acceleration coefficients and clamping the velocity, the performance of CPSO can be improved. Sometimes, the particles may still diverge in conventional PSO even when the maximum velocity and acceleration constants are correctly selected, this phenomenon is known as 'explosion' of the swarm. Two methods have been proposed to control the 'explosion', one is constriction factor and other is inertia weight [28].

## 3.5.2 CONSTRICTION FACTOR

The first method introduces a constriction factor to control the 'explosion' of the swarm, which was developed by Clerc and Kennedy [22]. The constriction factor introduced in the velocity rule (3.1) as,

$$V_{ij}^{t+1} = \chi(V_{ij}^{t} + c_1 r_1(P_{ij}^{best} - X_{ij}^{t}) + c_2 r_2(G_i^{best} - X_{ij}^{t})) \qquad (3.3)$$

Where $\chi$ is the constriction factor and given by,

$$\chi = \frac{2}{|2-\Phi-\sqrt{(\Phi^2-4\Phi)}|}, \Phi = c_1+c_2 > 4 \qquad (3.4)$$

Typically, $\Phi$ is set to 4.1 and $c_1=c_2=2.05$, thus the value of constriction factor has been set to 0.729 (used in this paper). In general, the convergence of the particle improved by the constriction factor once the particle is focused on the best point in the global region.

The constriction factor PSO method (CF-PSO) suffers from the disadvantage that when the individual best performance is far from the neighbourhood's best performance then the particles may follow wider cycles and may not converge.

## 3.5.3 INERTIA WEIGHT

The second method introduces a new parameter called inertia weight (proposed by Shi and Eberhart [23]) which is only multiply in the momentum component of the velocity at the previous time step in the velocity rule (3.1) as,

$$V_{ij}^{t+1} = wV_{ij}^{t} + c_1 r_1(P_{ij}^{best} - X_{ij}^{t}) + c_2 r_2(G_i^{best} - X_{ij}^{t}) \qquad (3.5)$$

Where ‘w’ indicate the inertia weight, which can be a fixed value or changing in each time step. The initial higher value (typically 0.9) allows the particles to move freely in the search space to find the global optimum faster. If once the optimal region is found, the value of inertia weight decreased usually to 0.4 in order to narrow the search space. This shifts search from an exploratory mode to an exploitative mode. Generally, a linearly decreasing inertia weight (introduced by Shi and Eberhart [23]) is used broadly given as,

$$w = w_{max} - t(w_{max}-w_{min})/It_{max} \quad (3.6)$$

Where $w_{max}$= 0.9, $w_{min}$= 0.4, t indicates the current iteration and $It_{max}$ is the maximum number of iteration to be performed.

The disadvantage of inertia weight PSO (IW-PSO) method is that once the inertia weight is decreased, the swarm loses its ability to search new areas because it is not able to recover its exploration mode.

# 3.6 ADVANTAGES OF PSO

There are many advantages of PSO which makes it attractive. Some of them are following-

a. The PSO algorithm is free from the use of derivatives.
b. Its’ concept is very simple.
c. The implementation of this algorithm is easy.
d. It involves less number of parameters compared to other techniques.
e. Its’ calculation is very simple and easy to understand.
f. It is less dependent upon the set of initial points.
g. Any mistakes in the intermediate stage not much more influence the results, but may increases the computational efforts.

# 3.7 DISADVANTAGES OF PSO

PSO is the one of the most powerful methods for solving the non-smooth optimization problems while there are some disadvantages of the PSO algorithm.

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

## 3.8 APPLICATION OF PSO

The PSO algorithms have been successfully applied to solve unconstrained as well as constrained problems, problems with dynamically changing landscapes, multi-objective optimization problems. It is also used to find multiple solutions. The PSO is used to solve a variety of optimization problem effectively in the area of electrical engineering [29]. Some examples are described below-

- Voltage and Reactive Power Control
- Economic Load Dispatch
- Power System Security and Reliability
- Generation Expansion Problem
- State Estimation
- Optimal Power Flow and Load Flow analysis
- Power System Identification and Control
- Electric Machinery
- Capacitor and FACTS Placement
- Unit-Commitment Scheduling and Generator Maintenance
- Short-Term Load Forecasting

The PSO is also used in the following areas-

- Antennas Design
- Signal Processing
- Networking
- Biomedical
- Electronics and electro-magnetic
- Robotics
- Design and Modelling
- Image and Graphics
- Fuzzy systems, Clustering, data mining
- Prediction and forecasting

# *CHAPTER 4*

# *IMPROVED PARTICLE SWARM OPTIMIZATION*

## 4.1 INTRODUCTION

In Conventional PSO, all particles of the population fly in the Search space during the entire run, but it is the behaviour of nature to achieve the optimum point as soon as possible. If anyone gets this position then it doesn't want to move any more. Such type of behaviour has been seen in the swarm intelligence. This behaviour of swarm is implemented in PSO algorithm in three ways in this work. Three improved PSO algorithms have been developed namely IPSO-A, IPSO-B and IPSO and implemented to solve mathematical functions and ELD problems.

## 4.2 IPSO-A

In CPSO, the particles continue to move as per equation (3.5) & (3.2), even after attaining the global best position. In this process the particles might move away from the global position, which may results in poor convergence. Therefore an improved PSO (IPSO-A) has been developed. In this algorithms, a particle which has attained global position is not allowed to move, till some other particles of the swarm achieves new global best position and all other particles will move as per the CPSO algorithm.

Mathematically,

Velocity modification in IPSO-A is,

$V_{ij}^{t+1} = 0$ , if j-th particle is at global position

$= wV_{ij}^{t} + c_1 r_1 (P_{ij}^{best} - X_{ij}^{t}) + c_2 r_2 (G_i^{best} - X_{ij}^{t})$, for other

And Position modification in IPSO-A is,

$X_{ij}^{t+1} = X_{ij}^{t}$ , if j-th particle is at global position

$= X_{ij}^{t} + V_{ij}^{t+1}$, for other particles

This algorithm can be understood by the flow chart. For this the dotted section of flow chart of CPSO is replaced by the section shown below in Fig. 4.1

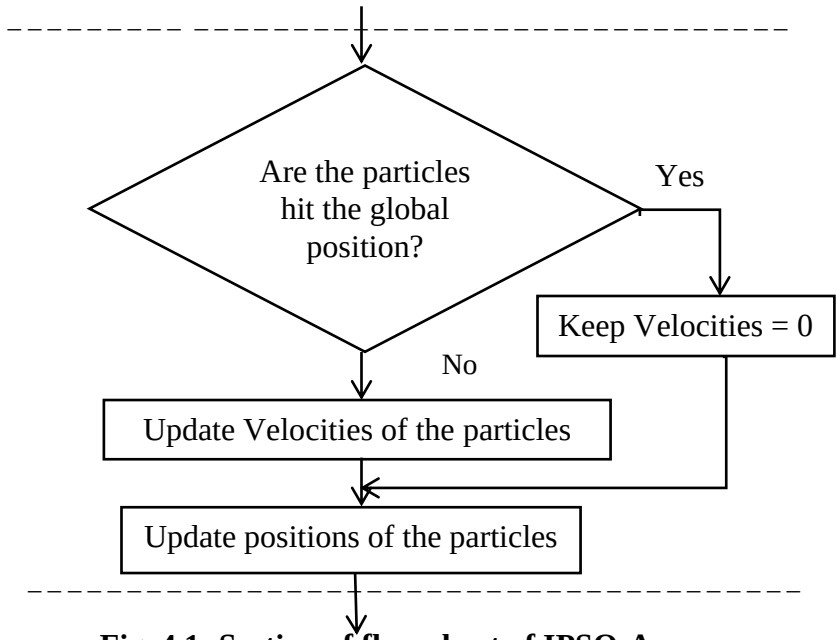


**Fig. 4.1: Section of flow chart of IPSO-A**

This improvement in the CPSO will results in the reduction of the search area of the swarm and oscillation of the particles which has attained global position. This algorithm gives consistent results and faster convergence.

# 4.3 IPSO-B

In IPSO-A, the particles at global position are not allowed to move for some time step/iteration i.e. they will not update their position. But this global position may not be the actual optimum. Therefore the particles which have achieved global position should be kept ready to move. In other words, the particles at global position will update their velocity but not the position. Whenever, such particles move, they will have all the three components of velocity.

Mathematically,

Velocity modification in IPSO-B is,

$$V_{ij}^{t+1} = wV_{ij}^{t} \text{ , if j-th particle is at global position}$$

$$= wV_{ij}^{t} + c_1r_1(P_{ij}^{best} - X_{ij}^{t}) + c_2r_2(G_i^{best} - X_{ij}^{t}), \text{ for other}$$

And Position modification in IPSO-B is,

$X_{ij}^{t+1} = X_{ij}^{t}$, if j-th particle is at global position

$= X_{ij}^{t} + V_{ij}^{t+1}$, for other particles

This algorithm can be understood by the flow chart. For this the dotted section of flow chart of CPSO is replaced by the section shown below in Fig. 4.2

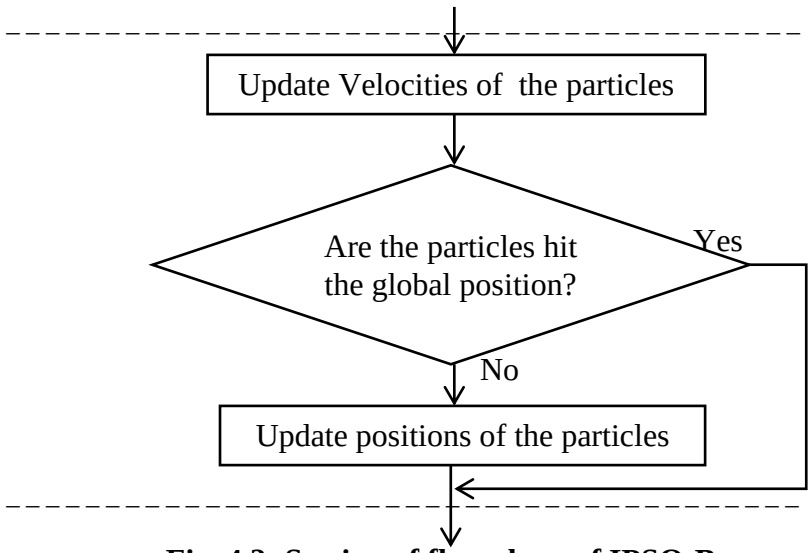


**Fig. 4.2: Section of flow chart of IPSO-B**

This has resulted in lower computational time and faster convergence than that of IPSO-A.

# 4.4 IPSO-C

The improvement in CPSO can also be done based on the function evaluation. If function evaluation at current position is found to be better than that of personal best position, the particle not allowed moving for some time step. However, the velocity of such particles will be updated as in CPSO so as to keep them ready for movement in future iterations.

Mathematically,

Velocity modification in IPSO-A is,

$$V_{ij}^{t+1} = wV_{ij}^{t} + c_1r_1(P_{ij}^{best} - X_{ij}^{t}) + c_2r_2(G_i^{best} - X_{ij}^{t})$$

And position modification in IPSO-C based on condition as,

$$X_{ij}^{t+1} = X_{ij}^{t}, \text{ If F(current position)} < \text{F(personal best)}$$
$$= X_{ij}^{t} + V_{ij}^{t+1}, \text{ for otherwise}$$

This algorithm can be understood by the flow chart. For this the dotted section of flow chart of CPSO is replaced by the section shown below in Fig. 4.3

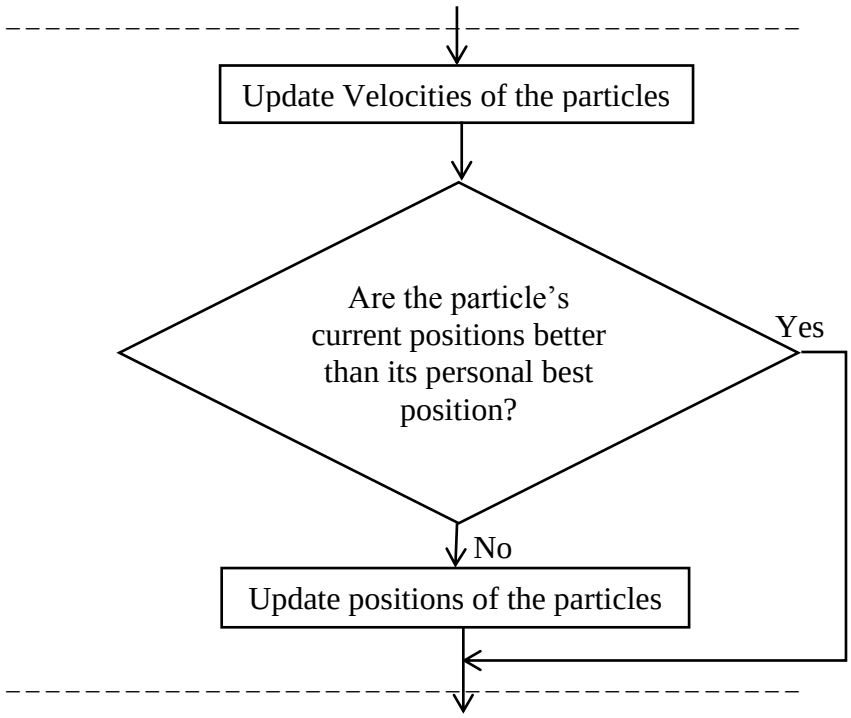


**Fig. 4.3: Section of flow chart of IPSO-C**

This algorithm has resulted in the best fitness evaluation**.**

# *CHAPTER 5*

# *SOLUTION OF BENCHMARK FUNCTIONS*

## 5.1 STEPS TO SOLVE BENCHMARK FUNCTIONS USING PSO IN MATLAB

The basic concepts for solving the optimization problems using PSO are explained in the previous chapter. Here generalized steps for solving different benchmark functions using PSO in MATLAB are discussed.

The steps are-

a. Set the number of particles 'p', maximum number of iteration '$it_{max}$' and tolerance limit 'T'.
b. Initialize all the variables matrices using the 'zeros' command of MATLAB.
c. Set the random numbers value '$r_1$' and '$r_2$' between 0 to 1.
d. Set the value of acceleration constants '$c_1$' and '$c_2$'. (generally both equal to 2)
e. Generate the random value of particles for each variable ($x_1$, $x_2$, …….).
f. Generate random value of velocities of particles for each variable ($v_1$,$v_2$,….. ).
g. Calculate the fitness value for each particle.
h. The position corresponding to best fitness is set to be $P^{best}$ for each particle.
i. The best of $P^{best}$ is set to be $G^{best}$.
j. Update the velocities and positions for each particle using velocity and position modification equation of PSO.
k. Calculate the fitness using new position for each particle.
l. Update $P^{best}$ for each particle and best of $P^{best}$ is set to be $G^{best}$.
m. The difference between the previous and current fitness is calculated and compared with tolerance limit for each particle. If it is within tolerance limit, then stop the iteration process otherwise go to step j.
n. $G^{best}$ is the optimal solution.

# 5.2 PARAMETERS SETTING FOR PSO ALGORITHMS

In this work Constriction Factor PSO (CPSO), Inertia Weight PSO (IW-PSO), combination of Constriction factor and Inertia weight PSO (CI-PSO), Improved PSO namely IPSO-A, IPSO-B and IPSO-C have been applied to solve four different mathematical functions namely BEALE'S, BOOTH'S, ROSENBROCK'S and SPHERE function. The parameters setting for various methods are described below in Table 5.1-

Table 5.1: Parameters setting for PSO algorithms

| | $c_1$ | $c_2$ | $r_1$ | $r_2$ | CF | Inertial Weight (W) | p | $It_{max}$ | ε |
|---|---|---|---|---|---|---|---|---|---|
| CF-PSO | 2.05 | 2.05 | 2 | 2 | 0.729 | | 30 | 1000 | $10^{-6}$ |
| IW-PSO | 2 | 2 | 2 | 2 | | Linearly Decreasing | 30 | 1000 | $10^{-6}$ |
| CI-PSO | 2.05 | 2.05 | 2 | 2 | 0.729 | Linearly Decreasing | 30 | 1000 | $10^{-6}$ |
| IPSO-A | 2 | 2 | 2 | 2 | | Linearly Decreasing | 30 | 1000 | $10^{-6}$ |
| IPSO-B | 2 | 2 | 2 | 2 | | Linearly Decreasing | 30 | 1000 | $10^{-6}$ |
| IPSO-C | 2 | 2 | 2 | 2 | | Linearly Decreasing | 30 | 1000 | $10^{-6}$ |

Where 'p' is the number of particles, 'ε' is the tolerance value and all other notation has their usual meaning as described earlier. The difference of fitness value between two consecutive iterations compared with tolerance value is considered to determine the stopping criteria for the convergence.

# 5.3 SOLUTION OF BEALE'S FUNCTION USING PSO ALGORITHMS

The Beale's function in two-dimension is defined as,

$$f(x_1,x_2) = (1.5 - x_1 + x_1 x_2)^2 + (2.25 - x_1 + x_1 x_2^2)^2 + (2.625 - x_1 + x_1 x_2^3)^2$$

The minimum value of Beale's function is as follows-

$$\text{Minimum } f(3, 0.5) = 0 \quad \text{for} \quad -4.5 < x_1, x_2 < 4.5$$

The 3-D surface plot of Beale's function is shown in the figure 5.1

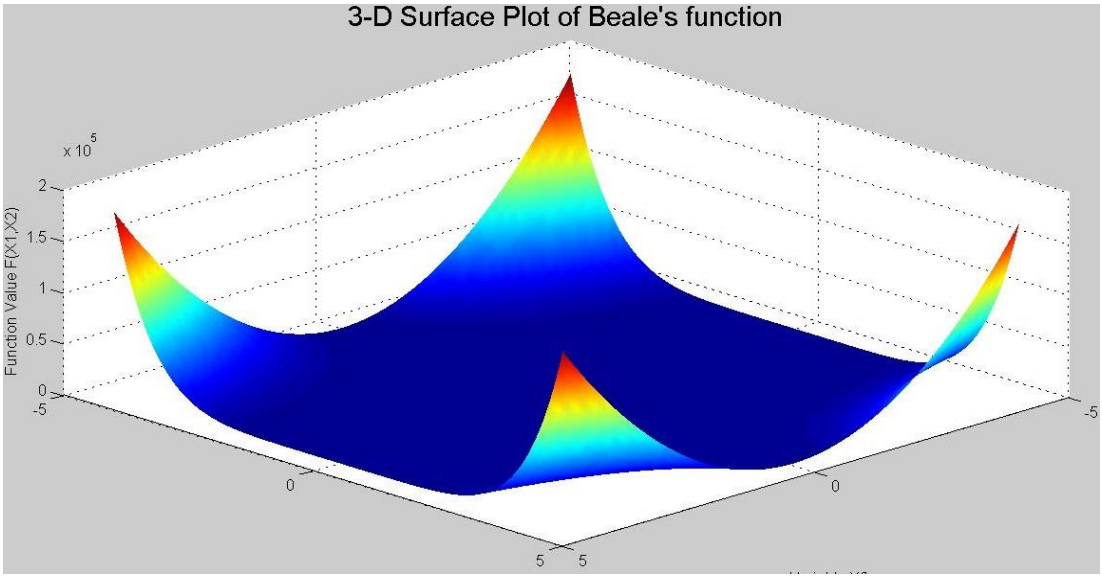


**Fig 5.1: 3-D Surface Plot of Beale's function**

The results of Beale's function using different PSO algorithms are shown in the Table 5.2

Table 5.2: Results for Beale's function

| | $x_1$ | $x_2$ | f ($x_1$,$x_2$) | No. of Iteration | Time (in sec.) |
|---|---|---|---|---|---|
| CF-PSO | 3 | 0.5000003 | $2.505847*10^{-12}$ | 50 | 0.1039 |
| IW-PSO | 3.000004 | 0.5000024 | $4.991952*10^{-11}$ | 134 | 0.3139 |
| CI-PSO | 3.000007 | 0.5000002 | $1.830699*10^{-12}$ | 36 | 0.0763 |
| IPSO-A | 2.999991 | 0.4999985 | $2.727247*10^{-11}$ | 105 | 0.2632 |
| IPSO-B | 3.000001 | 0.5 | $3.077159*10^{-12}$ | 108 | 0.2532 |
| IPSO-C | 3.000006 | 0.5000012 | $3.619538*10^{-11}$ | 112 | 0.2753 |

How the particles of the swarm find the global position is explained pictorially in the fig. 5.2 below. The figure shows the initial position, position after 1st iteration, position after 10th iteration, position after 25th iteration, position after 50th iteration and final (global) position of the swarm of 30 particles solving Beale's function of two variables. The first part of the figure 5.2 shows the initial random position of the particles. After first iteration all the particles come closer to each other as seen in the second part of the figure 5.2. The third part shows that the particles are attracted towards the global position i.e. 3 and 0.5 after 10th iteration. In fourth part, the particles come closer at its global position after 25th iteration. In fifth

part most of particle reached to global position after 50th iteration. In sixth part all the particles reached to the global position.

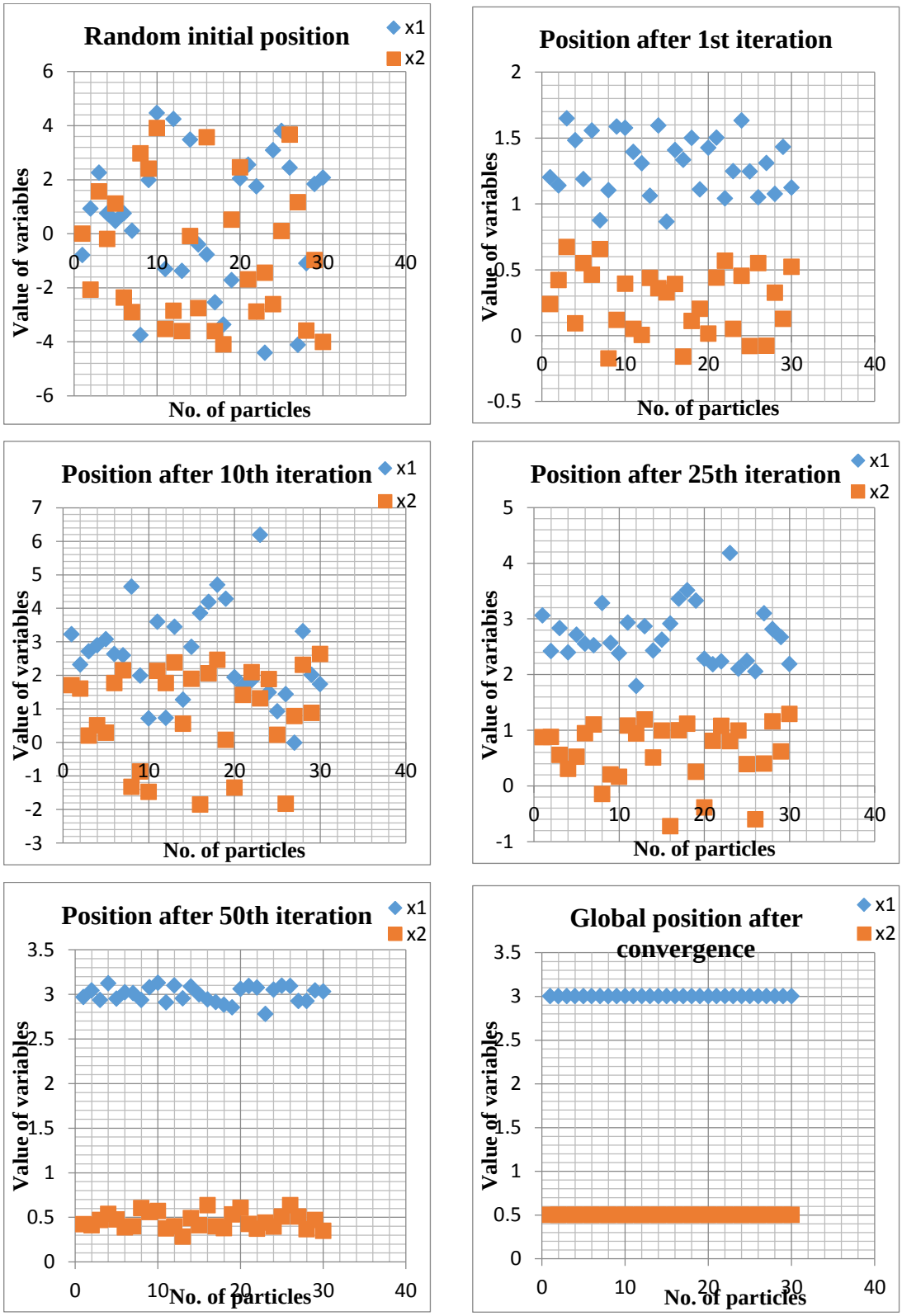


**Fig. 5.2: Position of the particles after different iteration for Beale's function**

# 5.4 SOLUTION OF BOOTH'S FUNCTION USING PSO ALGORITHMS

The Booth's function in two-dimension is defined as,

$$f(x_1,x_2) = (x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2$$

The minimum value of Booth's function is as follows-

$$\text{Minimum } f(1, 3) = 0 \quad \text{for} \quad -10 < x_1, x_2 < 10$$

The 3-D surface plot of Booth's function is shown in the figure 5.3

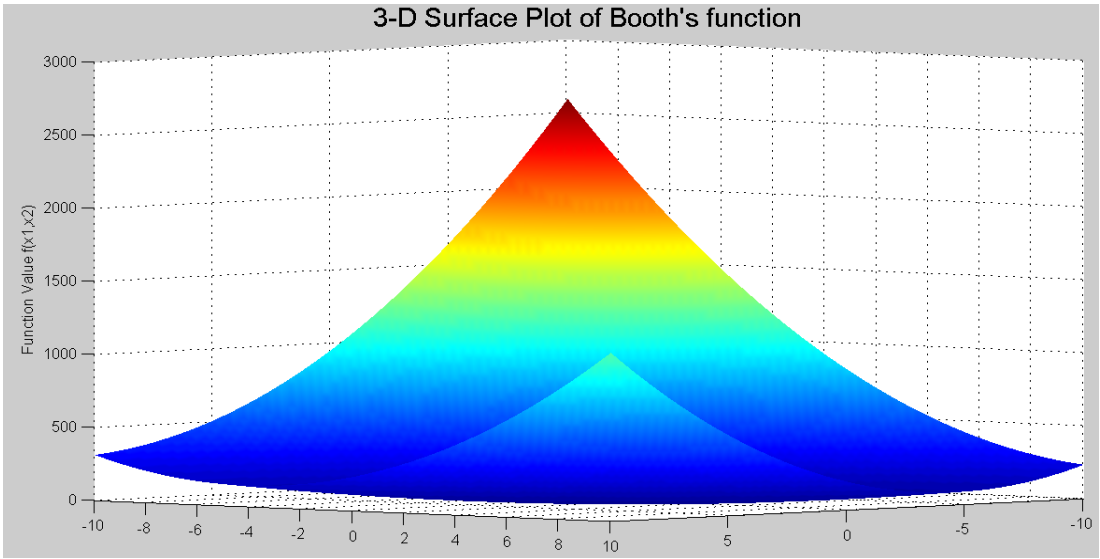


**Fig 5.3: 3-D Surface Plot of Booth's function**

The results of Beale's function using different PSO algorithms are shown in the Table 5.3

Table 5.3: Results for Booth's function

| | $x_1$ | $x_2$ | $f(x_1,x_2)$ | No. of Iteration | Time (in sec.) |
|---|---|---|---|---|---|
| CF-PSO | 0.9999991 | 3 | $2.927818*10^{-12}$ | 46 | 0.0949 |
| IW-PSO | 1.000004 | 2.999992 | $1.200323*10^{-10}$ | 127 | 0.2506 |
| CI-PSO | 0.9999996 | 3.000002 | $1.210196*10^{-11}$ | 34 | 0.0709 |
| IPSO-A | 0.9999989 | 3.000003 | $2.280425*10^{-11}$ | 89 | 0.1734 |
| IPSO-B | 1.000001 | 2.999999 | $1.64735*10^{-12}$ | 93 | 0.1601 |
| IPSO-C | 1.00001 | 2.999986 | $3.757722*10^{-10}$ | 99 | 0.1983 |

# 5.5 SOLUTION OF ROSENBROCK'S FUNCTION USING PSO ALGORITHMS

The Rosenbrock's function in two-dimension is defined as,

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

The minimum value of Rosenbrock's function is as follows-

Minimum $f(1,1) = 0$

The 3-D surface plot of Booth's function is shown in the figure 5.4

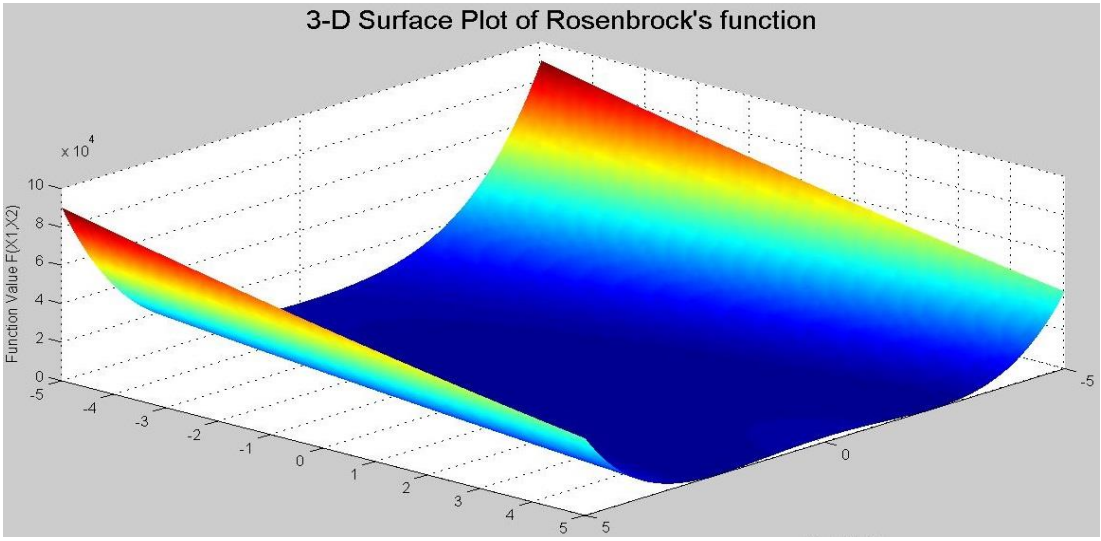


**Fig 5.4: 3-D Surface Plot of Rosenbrock's function**

The results of Rosenbrock's function using different PSO algorithms are shown in the Table 5.4

Table 5.4: Results for Rosenbrock's function

| | $x_1$ | $x_2$ | $f(x_1,x_2)$ | No. of Iteration | Time (in sec.) |
|---|---|---|---|---|---|
| CF-PSO | 1.000016 | 1.000032 | $2.470943*10^{-08}$ | 44 | 0.0634 |
| IW-PSO | 1.000005 | 1.000005 | $2.840467*10^{-09}$ | 132 | 0.3844 |
| CI-PSO | 0.9999997 | 0.9999992 | $1.261402*10^{-09}$ | 35 | 0.0489 |
| IPSO-A | 0.9999936 | 0.9999877 | $4.199148*10^{-09}$ | 102 | 0.2023 |
| IPSO-B | 1.000005 | 1.000006 | $1.890896*10^{-09}$ | 106 | 0.1832 |
| IPSO-C | 1.000009 | 1.000016 | $7.612831*10^{-09}$ | 118 | 0.2236 |

# 5.6 SOLUTION OF SPHERE FUNCTION USING PSO ALGORITHMS

The Sphere function in two-dimension is defined as,

$$f(x_1,x_2) = x_1^2 + x_2^2$$

The minimum value of Sphere function is as follows-

Minimum f (0, 0) = 0

The 3-D surface plot of Sphere function is shown in the figure 5.5

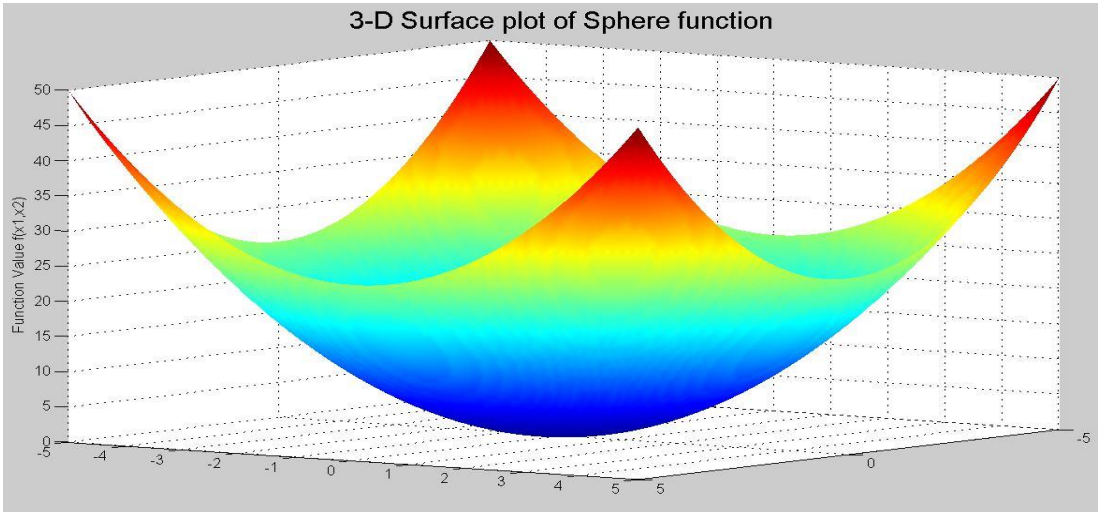


**Fig 5.5: 3-D Surface Plot of Sphere function**

The results of Sphere function using different PSO algorithms are shown in the Table 5.5

Table 5.5: Results for Sphere function

| | $x_1$ | $x_2$ | $f(x_1,x_2)$ | No. of Iteration | Time (in sec.) |
|---|---|---|---|---|---|
| CF-PSO | $2.190126*10^{-06}$ | $1.980476*10^{-06}$ | $8.718937*10^{-12}$ | 39 | 0.0853 |
| IW-PSO | $1.567241*10^{-06}$ | $2.709307*10^{-06}$ | $2.529648*10^{-10}$ | 102 | 0.2139 |
| CI-PSO | $-3.192096*10^{-07}$ | $-7.638486*10^{-06}$ | $5.844836*10^{-11}$ | 31 | 0.0694 |
| IPSO-A | $1.442569*10^{-05}$ | $1.480446*10^{-05}$ | $4.272725*10^{-10}$ | 81 | 0.1189 |
| IPSO-B | $-4.627715*10^{-06}$ | $6.682951*10^{-06}$ | $6.607759*10^{-11}$ | 83 | 0.1101 |
| IPSO-C | $-5.062913*10^{-06}$ | $-5.062913*10^{-06}$ | $1.017701*10^{-09}$ | 88 | 0.1246 |

## 5.7 DISCUSSION

The above results show that, the PSO algorithms (CF-PSO, IW-PSO, CI-PSO, IPSO-A, IPSO-B and IPSO-C) are able to solve the mathematical benchmark functions successfully. The use of constriction factor in the velocity modification equation reduces the number of iteration required for the convergence. Therefore constriction factor reduces the computational time. The linearly decreasing inertia weight shifts search from an exploratory mode to an exploitative mode. It has been seen that the use of both constriction factor and inertia weight simultaneously give most effective and reliable results. The CI-PSO gives result in the minimum number of iteration and hence takes less computational time among all the PSO algorithms considered in this work for mathematical benchmark functions (Beale's, Booth's, Rosenbrock's and Sphere function).

# *CHAPTER 6*

# *ECONOMIC LOAD DISPATCH USING PSO ALGORITHMS*

## 6.1 INTRODUCTION

ELD is the most important optimization problem of power system. The ELD involve a number of non-linear equations. Hence implementation of conventional method to solve such problem becomes complex. PSO is the optimization technique which has successfully applied to solve linear as well as non-linear problems. In this work Constriction Factor PSO (CPSO), Inertia Weight PSO (IW-PSO), combination of Constriction factor and Inertia weight PSO (CI-PSO), Improved PSO namely IPSO-A, IPSO-B and IPSO-C have been applied to solve ELD problem for IEEE 5, 14, 30-bus system. The population size (P) 30, Maximum iteration ($It_{max}$) 1000 and Tolerance $10^{-6}$ is taken in all cases. For Constriction Factor PSO, the value of $c_1=c_2=2.05$ and in all other cases $c_1=c_2=2$ is taken. The value of $r_1=r_2=0.5$ is used in all cases. The value of constriction factor is taken as 0.729. The linearly decreasing inertia weight from 0.9 to 0.4 is used. The parameter setting for different PSO algorithms are described in table 5.1. The penalty coefficient (K) for not satisfying load demand constraint is taken as 50 for all the bus system.

## 6.2 COMPUTATIONAL PROCEDURE

The procedure for the implementation of PSO for ELD problem are summarized by the following steps-

step 1) Initialization of the parameters- In the first step, the parameters of the PSO, like $c_1$, $c_2$, $r_1$, $r_2$, population size P, maximum iteration $It_{max}$ must be chosen.

step 2) Initialization of the population- Each particle are initialized with random positions ‘p’ (Power generated by the generating units) under the boundary

constraints of the generating units and velocities ‘v’ considering velocity clamping for proper convergence.

i.e. $p = \{p_{min} , p_{max}\}$

$v = \{-0.5p_{min} , 0.5p_{max}\}$

step 3) Evaluation of Objective function- The fuel cost of each particle in the population for each generating units and total production cost of the plant satisfying the equality constraints (power generation = power demand + power loss) is to be calculated.

step 4) Selection of initial personal and global best- Each initial value of particle is set to be its personal best and optimum of the personal bests is set to be global best.

step 5) Updation- Change the velocities and positions of the particles by velocity and position modification rule of PSO.

step 6) Boundary condition- Check that the new velocities and position are under the limit. If not, then force its value to the boundary as-

If $v < -0.5p_{min}$ , then $v= -0.5p_{min}$

If $v > 0.5p_{max}$ , then $v= 0.5p_{max}$

Similarly for power generation to be in limit

If $p < p_{min}$ , then $p=p_{min}$

If $p > p_{max}$ , then $p=p_{max}$

step 7) Updation of personal and global best value- If positions of the particles give the better fitness value during current iteration then set this as $P^{best}$ and the optimum of $P^{best}$ is set to be $G^{best}$.

step 8) Stopping Criteria- Check the difference in fitness value in the consecutive iteration is less than tolerance limit. If yes stop, otherwise go to step 5.

# 6.3 IEEE 5, 14 & 30 BUS SYSTEM DATA

The cost and loss coefficients of various generators are taken as given in Table 6.1 and Table 6.2 respectively.

Table 6.1: Data for cost coefficients

| BUS | | G1 | G2 | G3 | $P_D$ (in MW) |
|---|---|---|---|---|---|
| 5 | a ($/MW$^2$) | 0.005 | 0.005 | | 160 |
| | b ($/MW) | 3.51 | 3.89 | | |
| | c ($) | 44.4 | 40.6 | | |
| | $P_{min}$ (MW) | 30 | 30 | | |
| | $P_{max}$ (MW) | 120 | 120 | | |
| 14 | a ($/MW$^2$) | 0.005 | 0.005 | 0.005 | 259 |
| | b ($/MW) | 2.45 | 3.51 | 3.89 | |
| | c ($) | 105 | 44.4 | 40.6 | |
| | $P_{min}$ (MW) | 50 | 20 | 20 | |
| | $P_{max}$ (MW) | 200 | 100 | 100 | |
| 30 | a ($/MW$^2$) | 0.005 | 0.005 | 0.005 | 283.4 |
| | b ($/MW) | 2.45 | 3.51 | 3.89 | |
| | c ($) | 105 | 44.4 | 40.6 | |
| | $P_{min}$ (MW) | 50 | 30 | 30 | |
| | $P_{max}$ (MW) | 250 | 100 | 100 | |

Table 6.2: Data for Loss coefficients

| | |
|---|---|
| 5-BUS | $B = 10^{-4}\begin{bmatrix} 3.49 & 0.86 \\ -0.55 & 3.71 \end{bmatrix}$ |
| 14-BUS | $B = 10^{-4}\begin{bmatrix} 3.49 & 0.68 & -0.39 \\ 0.68 & 1.57 & 0.15 \\ -0.39 & 0.15 & 2.75 \end{bmatrix}$<br>$B_{i0} = 10^{-4}[\ 0.44\ \ 0.24\ \ 0\ ]$<br>$B_{00} = 10^{-4}[2.5408]$ |
| 30-BUS | $B = 10^{-4}\begin{bmatrix} 3.07 & 1.29 & 0.02 \\ 1.29 & 1.52 & -0.11 \\ -0.02 & -0.11 & 1.9 \end{bmatrix}$ |

# 6.4 RESULTS OF IEEE 5, 14 & 30-BUS SYSTEM

The ELD problem of IEEE 5, 14 and 30 bus systems have been solved using six different PSO algorithms described earlier. The results for the twenty trials have been shown below. Table 6.3 shows the results of 20 trials of IEEE 5 bus system. Similarly Table 6.4 and Table 6.5 show the results of 20 trials of IEEE 14 and 30 bus system respectively.

Table 6.3: Results of IEEE 5-BUS System

| CF-PSO | | | |
|---|---|---|---|
| Trial No. | Fuel Cost (in $/hr.) | No. of Iteration | Time (in sec.) |
| 1. | 762.823 | 85 | 0.1688 |
| 2. | 763.1705 | 88 | 0.1734 |
| 3. | 764.8201 | 82 | 0.1621 |
| 4. | 761.4755 | 81 | 0.1612 |
| 5. | **761.1372** | 79 | 0.1607 |
| 6. | 766.0175 | 82 | 0.1627 |
| 7. | 761.2032 | 82 | 0.1597 |
| 8. | 762.2008 | 78 | 0.1506 |
| 9. | 761.138 | 82 | 0.1559 |
| 10. | **765.3894** | 82 | 0.1692 |
| 11. | 761.1782 | 87 | 0.1695 |
| 12. | 761.1662 | 82 | 0.156 |
| 13. | 761.148 | 82 | 0.1582 |
| 14. | 761.25 | 82 | 0.1635 |
| 15. | 763.0206 | 79 | 0.16 |
| 16. | 762.006 | 82 | 0.166 |
| 17. | 763.7499 | 82 | 0.1617 |
| 18. | 761.289 | 83 | 0.1637 |
| 19. | 761.1538 | 82 | 0.1619 |
| 20. | 762.0355 | 82 | 0.1561 |
| Avg. | 762.3686 | 82 | 0.1620 |

| IW-PSO | | | |
|---|---|---|---|
| Trial No. | Fuel Cost (in $/hr.) | No. of Iteration | Time (in sec.) |
| 1. | 762.6597 | 216 | 0.3442 |
| 2. | 762.0023 | 221 | 0.3674 |
| 3. | 761.1963 | 211 | 0.5948 |
| 4. | 761.1653 | 222 | 0.4949 |
| 5. | 761.3246 | 230 | 0.3757 |
| 6. | **764.975** | 222 | 0.3682 |
| 7. | 762.5686 | 273 | 0.44 |
| 8. | 761.317 | 218 | 0.3554 |
| 9. | 761.3084 | 216 | 0.3513 |
| 10. | 761.4428 | 217 | 0.35 |
| 11. | 762.1345 | 221 | 0.3844 |
| 12. | 761.7634 | 240 | 0.3928 |
| 13. | 762.4156 | 275 | 0.4494 |
| 14. | **761.136** | 210 | 0.34 |
| 15. | 762.6994 | 219 | 0.3544 |
| 16. | 761.5892 | 216 | 0.3453 |
| 17. | 761.882 | 216 | 0.3444 |
| 18. | 761.1665 | 224 | 0.3529 |
| 19. | 761.5757 | 223 | 0.3678 |
| 20. | 761.5092 | 215 | 0.3607 |
| Avg. | 761.8916 | 225 | 0.3867 |

| CI-PSO | | | |
|---|---|---|---|
| Trial No. | Fuel Cost (in $/hr.) | No. of Iteration | Time (in sec.) |
| 1. | 761.1465 | 103 | 0.1829 |
| 2. | 761.366 | 99 | 0.1917 |

| IPSO-A | | | |
|---|---|---|---|
| Trial No. | Fuel Cost (in $/hr.) | No. of Iteration | Time (in sec.) |
| 1. | 762.1053 | 216 | 0.5378 |
| 2. | 761.4158 | 215 | 0.5374 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 3. | 761.2888 | 81 | 0.1553 | | 3. | 762.1738 | 201 | 0.4993 |
| 4. | 763.4789 | 96 | 0.178 | | 4. | 762.1022 | 223 | 0.5765 |
| 5. | **761.1357** | 96 | 0.1835 | | 5. | 761.6559 | 183 | 0.3898 |
| 6. | 763.3765 | 97 | 0.1769 | | 6. | 761.2539 | 185 | 0.3876 |
| 7. | 766.2869 | 96 | 0.1761 | | 7. | 761.7013 | 217 | 0.4919 |
| 8. | 766.279 | 101 | 0.1972 | | 8. | **761.17** | 215 | 0.5056 |
| 9. | 762.2343 | 108 | 0.1874 | | 9. | 761.8811 | 217 | 0.4931 |
| 10. | 764.6625 | 91 | 0.1693 | | 10. | 761.3344 | 212 | 0.4334 |
| 11. | 762.1127 | 102 | 0.1909 | | 11. | 761.7206 | 223 | 0.5469 |
| 12. | 761.3658 | 93 | 0.169 | | 12. | 761.6182 | 189 | 0.4511 |
| 13. | **766.6079** | 94 | 0.1733 | | 13. | 761.2224 | 216 | 0.5191 |
| 14. | 762.1437 | 105 | 0.1889 | | 14. | 761.9998 | 214 | 0.4908 |
| 15. | 763.0847 | 106 | 0.1952 | | 15. | 761.177 | 210 | 0.6074 |
| 16. | 761.2003 | 101 | 0.1904 | | 16. | 761.5042 | 216 | 0.5033 |
| 17. | 763.6541 | 100 | 0.1867 | | 17. | 761.4719 | 218 | 0.5628 |
| 18. | 766.1019 | 95 | 0.1729 | | 18. | 761.2443 | 223 | 0.6677 |
| 19. | 763.2724 | 107 | 0.1968 | | 19. | **762.404** | 201 | 0.4976 |
| 20. | 761.1464 | 99 | 0.1754 | | 20. | 761.574 | 216 | 0.5583 |
| Avg. | 763.0972 | 99 | 0.1819 | | Avg. | 761.6365 | 211 | 0.5129 |
| IPSO-B | | | | | IPSO-C | | | |
| Trial No. | Fuel Cost (in $/hr.) | No. of Iteration | Time (in sec.) | | Trial No. | Fuel Cost (in $/hr.) | No. of Iteration | Time (in sec.) |
| 1. | 762.3534 | 218 | 0.3625 | | 1. | 761.1357 | 210 | 0.3434 |
| 2. | 761.1953 | 216 | 0.368 | | 2. | 761.1352 | 183 | 0.2908 |
| 3. | 761.1749 | 202 | 0.339 | | 3. | 761.1352 | 317 | 0.4992 |
| 4. | 761.2884 | 216 | 0.372 | | 4. | 761.1352 | 199 | 0.3549 |
| 5. | **761.1352** | 219 | 0.5882 | | 5. | **762.5331** | 155 | 0.2648 |
| 6. | 761.4505 | 216 | 0.3652 | | 6. | 761.1352 | 161 | 0.2653 |
| 7. | 761.2288 | 216 | 0.377 | | 7. | 761.1354 | 363 | 0.5607 |
| 8. | 761.4053 | 216 | 0.4032 | | 8. | 761.1352 | 182 | 0.3039 |
| 9. | 761.8961 | 221 | 0.513 | | 9. | 761.1352 | 176 | 0.3403 |
| 10. | 762.7156 | 216 | 0.364 | | 10. | 761.1352 | 171 | 0.2785 |
| 11. | 761.6002 | 223 | 0.3702 | | 11. | 761.1352 | 298 | 0.4808 |
| 12. | 761.2266 | 216 | 0.3634 | | 12. | 761.1352 | 270 | 0.4395 |
| 13. | 761.4394 | 217 | 0.3694 | | 13. | **761.1352** | 151 | 0.2585 |
| 14. | 762.5657 | 221 | 0.3736 | | 14. | 761.6407 | 197 | 0.3313 |
| 15. | 761.1357 | 216 | 0.367 | | 15. | 761.1754 | 293 | 0.4623 |
| 16. | 761.2111 | 301 | 0.4999 | | 16. | 761.1353 | 216 | 0.3557 |
| 17. | **762.7848** | 216 | 0.3896 | | 17. | 761.1353 | 275 | 0.4607 |
| 18. | 761.218 | 223 | 0.3676 | | 18. | 761.1352 | 154 | 0.2652 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 19. | 761.9148 | 216 | 0.3607 | 19. | 761.1352 | 174 | 0.3013 |
| 20. | 762.3978 | 223 | 0.3757 | 20. | 761.1417 | 316 | 0.5164 |
| Avg. | 761.6669 | 221 | 0.3945 | Avg. | 761.2327 | 223 | 0.3687 |

Table 6.4: Results of IEEE 14-BUS System

| CF-PSO | | | | IW-PSO | | | |
|---|---|---|---|---|---|---|---|
| Trial No. | Fuel Cost (in $/hr.) | No. of Iteration | Time (in sec.) | Trial No. | Fuel Cost (in $/hr.) | No. of Iteration | Time (in sec.) |
| 1. | 1150.438 | 86 | 0.2141 | 1. | 1164.589 | 216 | 0.6626 |
| 2. | 1163.459 | 92 | 0.2257 | 2. | 1146.295 | 270 | 0.9164 |
| 3. | 1151.778 | 91 | 0.2235 | 3. | 1147.372 | 206 | 0.6412 |
| 4. | 1164.253 | 88 | 0.2118 | 4. | **1143.884** | 206 | 0.6371 |
| 5. | 1147.883 | 88 | 0.2109 | 5. | 1161.208 | 203 | 0.6312 |
| 6. | 1147.105 | 90 | 0.2171 | 6. | 1144.145 | 278 | 0.9934 |
| 7. | 1151.818 | 88 | 0.2181 | 7. | 1149.61 | 208 | 0.6373 |
| 8. | 1151.365 | 88 | 0.2239 | 8. | 1160.572 | 262 | 0.9568 |
| 9. | 1159.595 | 90 | 0.2207 | 9. | 1145.327 | 270 | 0.9741 |
| 10. | **1143.908** | 86 | 0.2116 | 10. | 1156.66 | 210 | 0.6476 |
| 11. | 1155.479 | 91 | 0.2192 | 11. | **1186.491** | 260 | 0.9124 |
| 12. | 1150.767 | 91 | 0.2223 | 12. | 1143.885 | 212 | 0.6706 |
| 13. | 1158.797 | 88 | 0.2094 | 13. | 1172.65 | 217 | 0.6791 |
| 14. | 1149.814 | 92 | 0.2158 | 14. | 1147.763 | 208 | 0.6684 |
| 15. | 1157.591 | 95 | 0.2338 | 15. | 1157.131 | 258 | 0.8996 |
| 16. | 1159.537 | 90 | 0.2158 | 16. | 1147.736 | 251 | 0.8461 |
| 17. | 1150.343 | 88 | 0.2127 | 17. | 1147.269 | 258 | 0.856 |
| 18. | 1148.798 | 88 | 0.2099 | 18. | 1147.161 | 207 | 0.6323 |
| 19. | **1171.506** | 86 | 0.2115 | 19. | 1147.777 | 204 | 0.6331 |
| 20. | 1157.282 | 90 | 0.2167 | 20. | 1147.39 | 255 | 0.878 |
| Avg. | 1154.576 | 89 | 0.2172 | Avg. | 1153.246 | 233 | 0.7687 |
| CI-PSO | | | | IPSO-A | | | |
| Trial No. | Fuel Cost (in $/hr.) | No. of Iteration | Time (in sec.) | Trial No. | Fuel Cost (in $/hr.) | No. of Iteration | Time (in sec.) |
| 1. | 1158.469 | 110 | 0.2551 | 1. | 1150.913 | 214 | 0.7656 |
| 2. | **1144.111** | 111 | 0.3595 | 2. | 1156.915 | 238 | 0.8858 |
| 3. | 1146.389 | 108 | 0.2533 | 3. | 1149.377 | 216 | 0.776 |
| 4. | 1161.444 | 112 | 0.2571 | 4. | 1154.128 | 232 | 0.9445 |
| 5. | 1151.028 | 108 | 0.2519 | 5. | 1145.613 | 229 | 0.9499 |
| 6. | 1154.499 | 93 | 0.2285 | 6. | 1148.896 | 210 | 0.7265 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 7. | 1151.266 | 95 | 0.2235 | 7. | 1145.139 | 208 | 0.7248 |
| 8. | 1150.729 | 117 | 0.2731 | 8. | 1147.484 | 205 | 0.7246 |
| 9. | 1153.375 | 107 | 0.2444 | 9. | **1163.342** | 227 | 0.9193 |
| 10. | **1167.771** | 113 | 0.2596 | 10. | 1147.764 | 233 | 0.9334 |
| 11. | 1150.66 | 113 | 0.2629 | 11. | 1157.427 | 231 | 0.9999 |
| 12. | 1163.097 | 100 | 0.2333 | 12. | 1154.072 | 229 | 0.8359 |
| 13. | 1149.195 | 109 | 0.2569 | 13. | 1158.279 | 227 | 0.8572 |
| 14. | 1145.301 | 109 | 0.2505 | 14. | 1150.968 | 230 | 0.9004 |
| 15. | 1160.012 | 104 | 0.2479 | 15. | **1144.201** | 226 | 0.9161 |
| 16. | 1164.61 | 107 | 0.2479 | 16. | 1160.347 | 236 | 1.007 |
| 17. | 1145.717 | 114 | 0.2657 | 17. | 1149.933 | 233 | 1.005 |
| 18. | 1150.318 | 113 | 0.2584 | 18. | 1154.18 | 229 | 0.8933 |
| 19. | 1153.219 | 104 | 0.246 | 19. | 1148.369 | 226 | 0.8922 |
| 20. | 1146.404 | 115 | 0.2619 | 20. | 1153.952 | 224 | 0.9552 |
| Avg. | 1153.381 | 108 | 0.2569 | Avg. | 1153.515 | 225 | 1.0019 |

| IPSO-B | | | | IPSO-C | | | |
|---|---|---|---|---|---|---|---|
| Trial No. | Fuel Cost (in $/hr.) | No. of Iteration | Time (in sec.) | Trial No. | Fuel Cost (in $/hr.) | No. of Iteration | Time (in sec.) |
| 1. | 1157.753 | 249 | 0.7634 | 1. | 1146.265 | 291 | 0.9707 |
| 2. | 1153.237 | 247 | 0.7068 | 2. | 1143.859 | 297 | 0.7148 |
| 3. | 1153.494 | 215 | 0.6883 | 3. | 1146.419 | 301 | 0.7137 |
| 4. | **1145.833** | 224 | 0.7298 | 4. | 1143.853 | 301 | 0.7305 |
| 5. | 1149.282 | 208 | 0.6816 | 5. | 1144.155 | 307 | 1.353 |
| 6. | 1148.743 | 214 | 0.7009 | 6. | 1143.853 | 294 | 0.9041 |
| 7. | 1153.823 | 236 | 0.7245 | 7. | 1144.177 | 298 | 0.8931 |
| 8. | 1149.545 | 206 | 0.7009 | 8. | 1148.058 | 299 | 0.6884 |
| 9. | 1157.223 | 264 | 0.791 | 9. | **1153.009** | 302 | 0.8275 |
| 10. | 1147.41 | 215 | 0.916 | 10. | **1143.853** | 291 | 0.6653 |
| 11. | 1147.75 | 205 | 0.6804 | 11. | 1144.056 | 301 | 0.7355 |
| 12. | 1145.859 | 296 | 0.6817 | 12. | 1143.924 | 292 | 0.9573 |
| 13. | 1152.194 | 225 | 0.7255 | 13. | 1143.872 | 297 | 0.7486 |
| 14. | 1153.052 | 211 | 0.7048 | 14. | 1146.643 | 306 | 0.8567 |
| 15. | 1159.719 | 201 | 0.6711 | 15. | 1143.854 | 298 | 0.8557 |
| 16. | 1152.559 | 255 | 0.7772 | 16. | 1150.204 | 297 | 0.6609 |
| 17. | **1162.29** | 207 | 0.682 | 17. | 1143.89 | 295 | 0.7556 |
| 18. | 1151.45 | 205 | 0.6884 | 18. | 1143.902 | 294 | 0.7525 |
| 19. | 1150.062 | 219 | 0.7178 | 19. | 1143.864 | 309 | 0.6603 |
| 20. | 1147.135 | 221 | 0.6669 | 20. | 1144.448 | 296 | 0.6977 |
| Avg. | 1151.921 | 226 | 0.7349 | Avg. | 1145.308 | 298 | 1.0058 |

Table 6.5: Results of IEEE 30-BUS System

| CF-PSO | | | | IW-PSO | | | |
|---|---|---|---|---|---|---|---|
| Trial No. | Fuel Cost (in $/hr.) | No. of Iteration | Time (in sec.) | Trial No. | Fuel Cost (in $/hr.) | No. of Iteration | Time (in sec.) |
| 1. | 1271.72 | 89 | 0.1956 | 1. | 1257.35 | 241 | 0.4618 |
| 2. | **1256.401** | 95 | 0.2228 | 2. | 1257.312 | 237 | 0.4716 |
| 3. | 1257.761 | 91 | 0.2075 | 3. | 1256.823 | 232 | 0.4436 |
| 4. | 1278.789 | 89 | 0.1974 | 4. | 1279.779 | 237 | 0.4604 |
| 5. | 1256.876 | 89 | 0.1979 | 5. | 1266.617 | 237 | 0.4666 |
| 6. | 1259.723 | 96 | 0.2081 | 6. | 1279.141 | 232 | 0.4492 |
| 7. | 1269.204 | 89 | 0.2506 | 7. | 1256.279 | 237 | 0.4602 |
| 8. | 1257.477 | 91 | 0.2113 | 8. | 1257.057 | 267 | 0.7334 |
| 9. | 1275.185 | 93 | 0.2304 | 9. | 1264.149 | 234 | 0.4352 |
| 10. | 1278.166 | 88 | 0.2215 | 10. | 1259.052 | 230 | 0.4346 |
| 11. | 1260.94 | 91 | 0.2069 | 11. | **1256.264** | 237 | 0.4552 |
| 12. | 1280.52 | 91 | 0.2442 | 12. | 1257.662 | 239 | 0.5449 |
| 13. | 1260.779 | 91 | 0.2347 | 13. | 1265.116 | 242 | 0.4574 |
| 14. | **1284.09** | 98 | 0.2962 | 14. | 1273.777 | 233 | 0.4904 |
| 15. | 1257.756 | 93 | 0.2047 | 15. | 1256.293 | 233 | 0.4406 |
| 16. | 1264.617 | 92 | 0.2205 | 16. | 1257.8 | 236 | 0.5885 |
| 17. | 1259.847 | 93 | 0.2141 | 17. | 1283.973 | 230 | 0.4464 |
| 18. | 1261.267 | 91 | 0.2605 | 18. | 1262.642 | 237 | 0.4405 |
| 19. | 1276.012 | 92 | 0.2077 | 19. | **1284.362** | 235 | 0.4645 |
| 20. | 1273.753 | 84 | 0.2022 | 20. | 1256.657 | 233 | 0.4508 |
| Avg. | 1267.044 | 91 | 0.2217 | Avg. | 1264.405 | 237 | 0.4798 |
| CI-PSO | | | | IPSO-A | | | |
| Trial No. | Fuel Cost (in $/hr.) | No. of Iteration | Time (in sec.) | Trial No. | Fuel Cost (in $/hr.) | No. of Iteration | Time (in sec.) |
| 1. | 1256.527 | 95 | 0.3126 | 1. | 1261.02 | 230 | 0.6839 |
| 2. | **1256.29** | 105 | 0.3039 | 2. | 1265.735 | 236 | 0.7809 |
| 3. | 1269.379 | 110 | 0.2647 | 3. | 1257.599 | 232 | 0.6824 |
| 4. | 1274.647 | 108 | 0.3179 | 4. | 1259.944 | 232 | 0.6602 |
| 5. | 1257.508 | 113 | 0.2882 | 5. | **1256.264** | 231 | 0.8659 |
| 6. | 1259.848 | 106 | 0.2812 | 6. | 1257.679 | 227 | 0.7342 |
| 7. | 1262.224 | 118 | 0.3448 | 7. | 1261.402 | 234 | 0.6876 |
| 8. | 1257.031 | 117 | 0.2663 | 8. | 1256.708 | 233 | 0.8216 |
| 9. | 1256.642 | 105 | 0.2316 | 9. | 1256.664 | 228 | 0.7329 |
| 10. | 1259.092 | 116 | 0.2553 | 10. | 1258.757 | 236 | 0.7089 |
| 11. | 1279.952 | 106 | 0.2528 | 11. | 1261.222 | 235 | 0.8425 |

| 12. | 1271.287 | 110 | 0.2538 |
|---|---|---|---|
| 13. | 1263.164 | 113 | 0.2422 |
| 14. | 1259.459 | 118 | 0.2627 |
| 15. | **1280.542** | 123 | 0.2652 |
| 16. | 1268.984 | 109 | 0.2507 |
| 17. | 1256.306 | 112 | 0.2586 |
| 18. | 1258.716 | 112 | 0.2491 |
| 19. | 1258.489 | 110 | 0.2553 |
| 20. | 1256.925 | 118 | 0.4929 |
| Avg. | 1263.150 | 111 | 0.2825 |

| 12. | 1256.726 | 238 | 0.7196 |
|---|---|---|---|
| 13. | 1357.143 | 232 | 0.6872 |
| 14. | 1257.154 | 232 | 0.7121 |
| 15. | 1258.03 | 223 | 0.5732 |
| 16. | 1257.826 | 242 | 0.7765 |
| 17. | **1277.472** | 232 | 0.7144 |
| 18. | 1257.421 | 229 | 0.6283 |
| 19. | 1260.135 | 235 | 0.7108 |
| 20. | 1256.627 | 234 | 0.6722 |
| Avg. | 1259.576 | 233 | 0.7198 |

IPSO-B

| Trial No. | Fuel Cost (in $/hr.) | No. of Iteration | Time (in sec.) |
|---|---|---|---|
| 1. | 1260.127 | 233 | 0.4774 |
| 2. | 1258.97 | 234 | 0.5079 |
| 3. | 1258.756 | 238 | 0.7357 |
| 4. | 1256.645 | 234 | 0.4693 |
| 5. | 1256.587 | 239 | 0.4827 |
| 6. | 1256.637 | 238 | 0.5081 |
| 7. | 1256.303 | 236 | 0.4781 |
| 8. | 1256.603 | 235 | 0.6612 |
| 9. | 1265.598 | 244 | 0.5068 |
| 10. | 1257.632 | 231 | 0.4972 |
| 11. | 1256.365 | 233 | 0.5099 |
| 12. | 1258.436 | 233 | 0.4929 |
| 13. | 1262.795 | 234 | 0.5001 |
| 14. | 1257.044 | 235 | 0.5024 |
| 15. | **1267.616** | 232 | 0.4822 |
| 16. | **1256.217** | 231 | 0.4938 |
| 17. | 1261.508 | 233 | 0.4862 |
| 18. | 1256.492 | 231 | 0.4837 |
| 19. | 1258.448 | 237 | 0.4844 |
| 20. | 1265.353 | 235 | 0.4909 |
| Avg. | 1259.207 | 235 | 0.5125 |

IPSO-C

| Trial No. | Fuel Cost (in $/hr.) | No. of Iteration | Time (in sec.) |
|---|---|---|---|
| 1. | 1256.193 | 255 | 0.6141 |
| 2. | 1256.669 | 179 | 0.4731 |
| 3. | 1256.465 | 170 | 0.3347 |
| 4. | 1256.192 | 222 | 0.4968 |
| 5. | 1257.129 | 497 | 1.1552 |
| 6. | 1256.192 | 268 | 0.5054 |
| 7. | 1256.368 | 309 | 0.6067 |
| 8. | 1256.193 | 417 | 0.8118 |
| 9. | 1256.193 | 412 | 0.7755 |
| 10. | **1256.192** | 215 | 0.6425 |
| 11. | 1256.192 | 411 | 0.7638 |
| 12. | 1257.719 | 469 | 1.1079 |
| 13. | 1256.192 | 319 | 0.6454 |
| 14. | **1272.138** | 190 | 0.3645 |
| 15. | 1258.073 | 387 | 0.8488 |
| 16. | 1258.781 | 227 | 0.5138 |
| 17. | 1258.06 | 558 | 1.1207 |
| 18. | 1256.635 | 157 | 0.3922 |
| 19. | 1256.196 | 244 | 0.4796 |
| 20. | 1256.192 | 223 | 0.5084 |
| Avg. | 1257.498 | 306 | 0.6580 |

# 6.5 DISCUSSION

The best cost, average cost, worst cost, average number of iteration, average computational time and standard deviation cost of 20 trials are summarized for different PSO algorithms. Table 6.6 shows the compact results of IEEE 5 bus system. Similarly table 6.7 and table 6.8 show the compact results of IEE 14 and 30 bus systems respectively.

Table 6.6: Compact Results of IEEE 5-bus system of 20 trials

| | Best Fuel Cost (in $/hr.) | Average Fuel Cost (in $/hr.) | Worst Fuel Cost (in $/hr.) | Average no. of Iteration | Average Computational Time (in Sec.) | Standard Deviation (in $/hr.) |
|---|---|---|---|---|---|---|
| CF-PSO | 761.1372 | 762.3686 | 766.0175 | **82** | **0.1620** | 1.544677 |
| IW-PSO | 761.1360 | 761.8916 | 764.9750 | 225 | 0.3867 | 0.894688 |
| CI-PSO | 761.1357 | 763.0972 | 766.6079 | 99 | 0.1819 | 1.941389 |
| IPSO-A | 761.1700 | 761.6365 | **762.4040** | 211 | 0.5129 | 0.370428 |
| IPSO-B | **761.1352** | 761.6669 | 762.7848 | 221 | 0.3945 | 0.581103 |
| IPSO-C | **761.1352** | **761.2327** | 762.5331 | 223 | 0.3687 | **0.326141** |

Table 6.7: Compact Results of IEEE 14-bus system of 20 trials

| | Best Fuel Cost (in $/hr.) | Average Fuel Cost (in $/hr.) | Worst Fuel Cost (in $/hr.) | Average no. of Iteration | Average Computational Time (in Sec.) | Standard Deviation (in $/hr.) |
|---|---|---|---|---|---|---|
| CF-PSO | 1143.908 | 1154.576 | 1171.506 | **89** | **0.2172** | 6.843359 |
| IW-PSO | 1143.884 | 1153.246 | 1186.491 | 233 | 0.7687 | 11.16492 |
| CI-PSO | 1144.111 | 1153.381 | 1167.771 | 108 | 0.2569 | 6.968287 |
| IPSO-A | 1144.201 | 1153.515 | 1163.342 | 225 | 1.0019 | 5.290987 |
| IPSO-B | 1145.833 | 1151.921 | 1162.29 | 226 | 0.7349 | 4.593381 |
| IPSO-C | **1143.853** | **1145.308** | **1153.009** | 298 | 1.0058 | **2.516282** |

Table 6.8: Compact Results of IEEE 30-bus system of 20 trials

| | Best Fuel Cost (in $/hr.) | Average Fuel Cost (in $/hr.) | Worst Fuel Cost (in $/hr.) | Average no. of Iteration | Average Computational Time (in Sec.) | Standard Deviation (in $/hr.) |
|---|---|---|---|---|---|---|
| CF-PSO | 1256.401 | 1267.044 | 1284.090 | **91** | **0.2217** | 9.329782 |

| | | | | | | |
|---|---|---|---|---|---|---|
| IW-PSO | 1256.264 | 1264.405 | 1284.362 | 237 | 0.4798 | 10.04872 |
| CI-PSO | 1256.285 | 1263.150 | 1280.542 | 111 | 0.2825 | 8.037626 |
| IPSO-A | 1256.264 | 1259.576 | 1277.472 | 233 | 0.7198 | 4.819535 |
| IPSO-B | **1256.217** | 1259.207 | **1267.616** | 235 | 0.5125 | **3.528597** |
| IPSO-C | **1256.192** | **1257.498** | 1272.138 | 306 | 0.6580 | 3.536436 |

The Bar chart of average computational time and standard deviation cost for different PSO algorithms are shown below. The bar chart is used for comparisons of data of the groups. The computational time is the measure of the computational efforts. The consistency of the results is indicated by standard deviation cost.

Fig. 6.1 and Fig 6.2 shows the bar chart for average computational time and standard deviation cost of different PSO algorithms for 5, 14 and 30 bus system respectively.

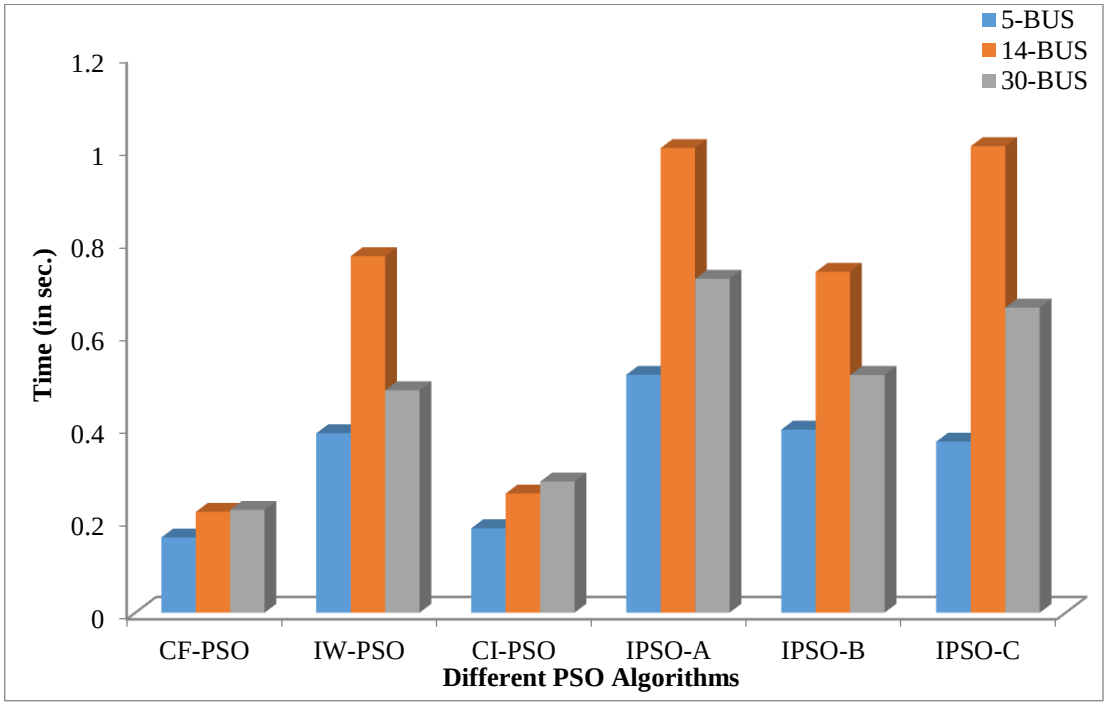


**Fig. 6.1: Bar chart for Average Computational time**

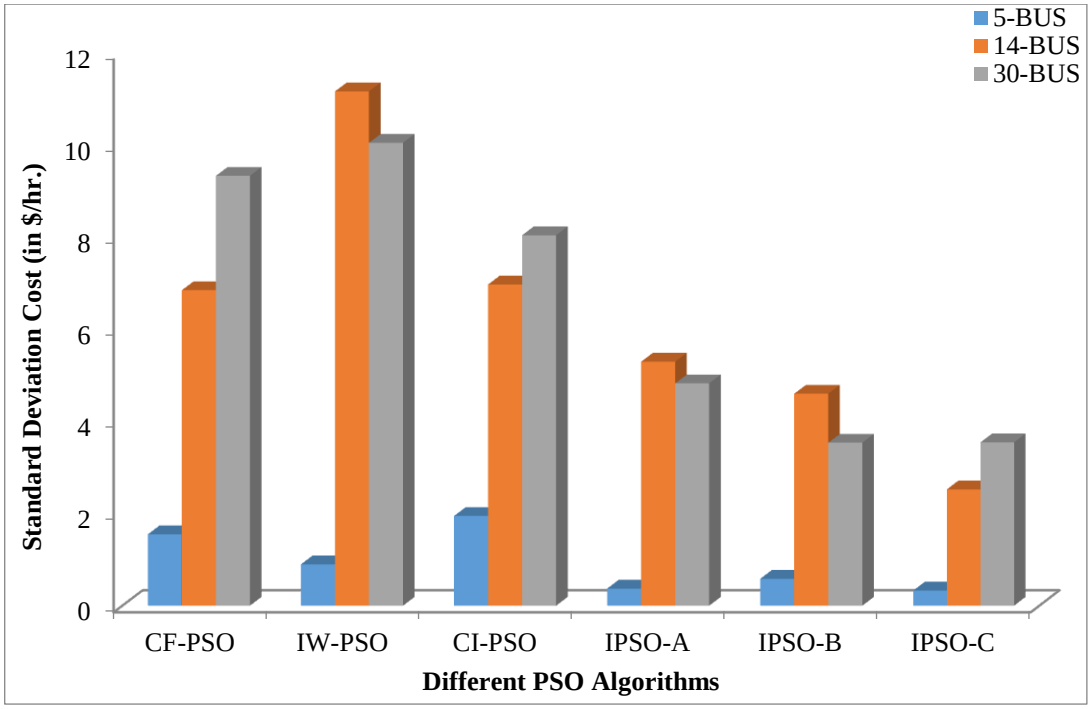


**Fig. 6.2: Bar Chart for Standard Deviation costs**

The results show that the CF-PSO give results in less number of iteration and hence takes less computational time for all the bus system but the standard deviation cost is in considerable amount. The IW-PSO gives maximum standard deviation cost for 14 and 30 bus system. The CI-PSO also takes less number of iteration except CF-PSO, but standard deviation cost in case of 5 bus system is highest. The IPSO-A gives better results than CF-PSO, IW-PSO & CI-PSO in terms of standard deviation cost. The IPSO-B gives optimum cost for 5 bus systems and lowest standard deviation cost for 30 bus systems. The IPSO-C gives optimum (best) cost for all bus system and lowest standard deviation cost for 5 and 14 bus systems.

# *CHAPTER 7*

# *CONCLUSIONS & FUTURE DIRECTIONS*

The results show that constriction factor reduces the computational time and the number of iteration. The linearly decreasing inertia weight shifts search from an exploratory mode to an exploitative mode. It has been seen that the use of constriction factor simultaneously with inertia weight gives better results for mathematical benchmark functions. But the main objective of ELD problem is to minimize the fuel cost of the plant. The improved PSO (IPSO-A, IPSO-B and IPSO-C) described in this work gives better results than conventional PSO (CF-PSO, IW-PSO and CI-PSO) in terms of optimum fuel costs. The improved PSO gives the consistent results than conventional PSO. The comparison between the improved PSO shows that, the IPSO-C gives minimum fuel cost among them with consistency in all cases.

The other objective of ELD can also be considered like- transmission losses, environmental pollution etc. and solved by improved PSO (IPSO-A, IPSO-B and IPSO-C) in future works.

# APPENDIX- I

## 1) IEEE 5 BUS SYSTEM

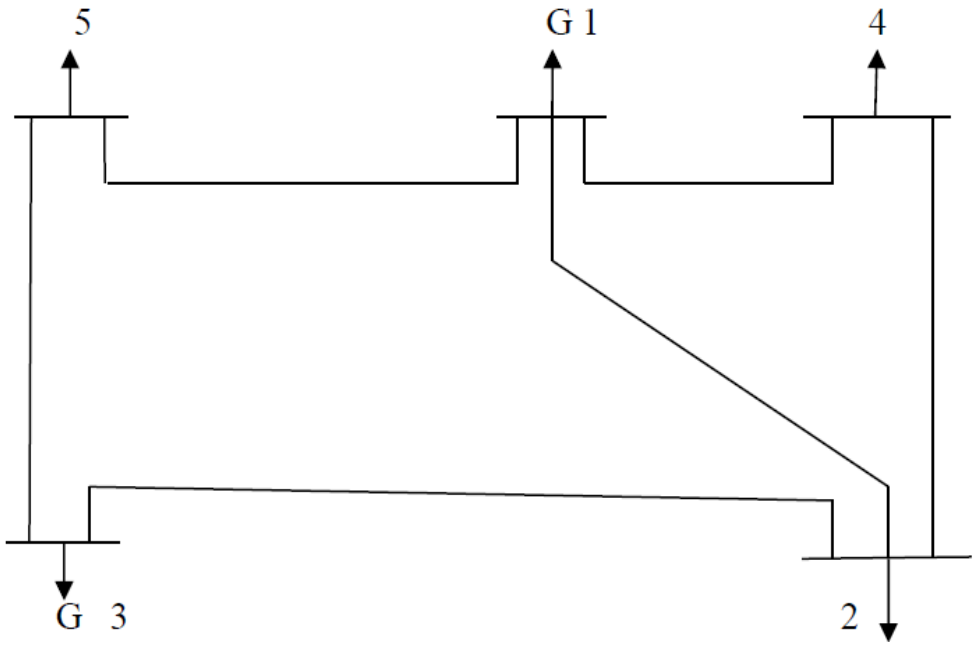


**Fig. I-A: BUS-CODE DIAGRAM OF 5 BUS SYSTEM**

**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

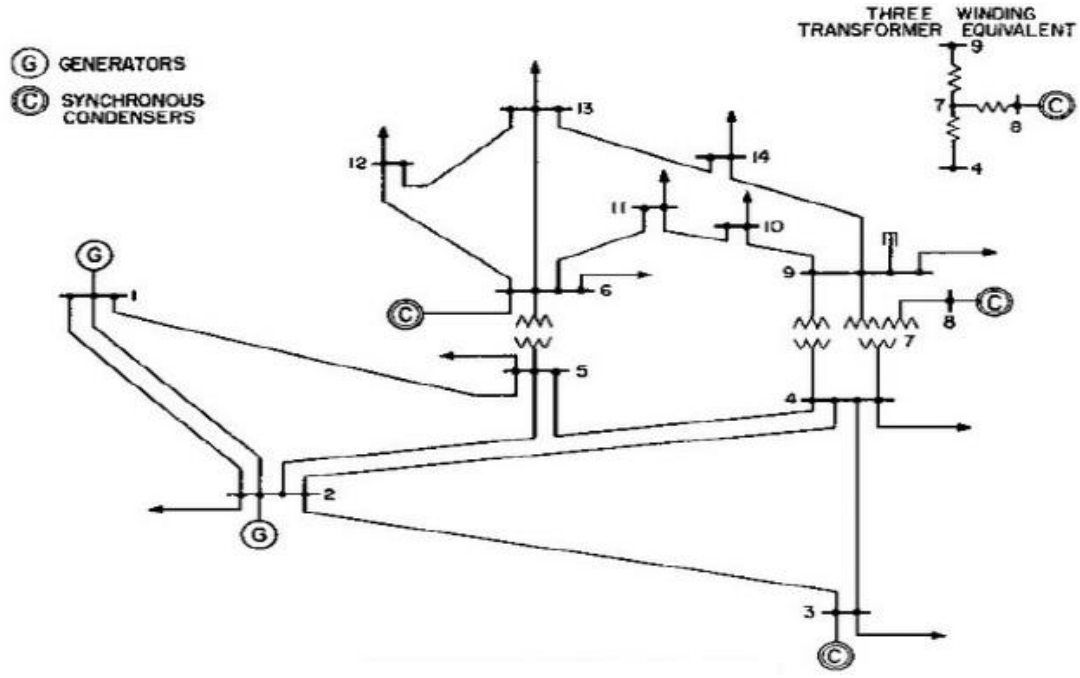


**Fig. I-B: BUS-CODE DIAGRAM OF 14 BUS SYSTEM**

**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

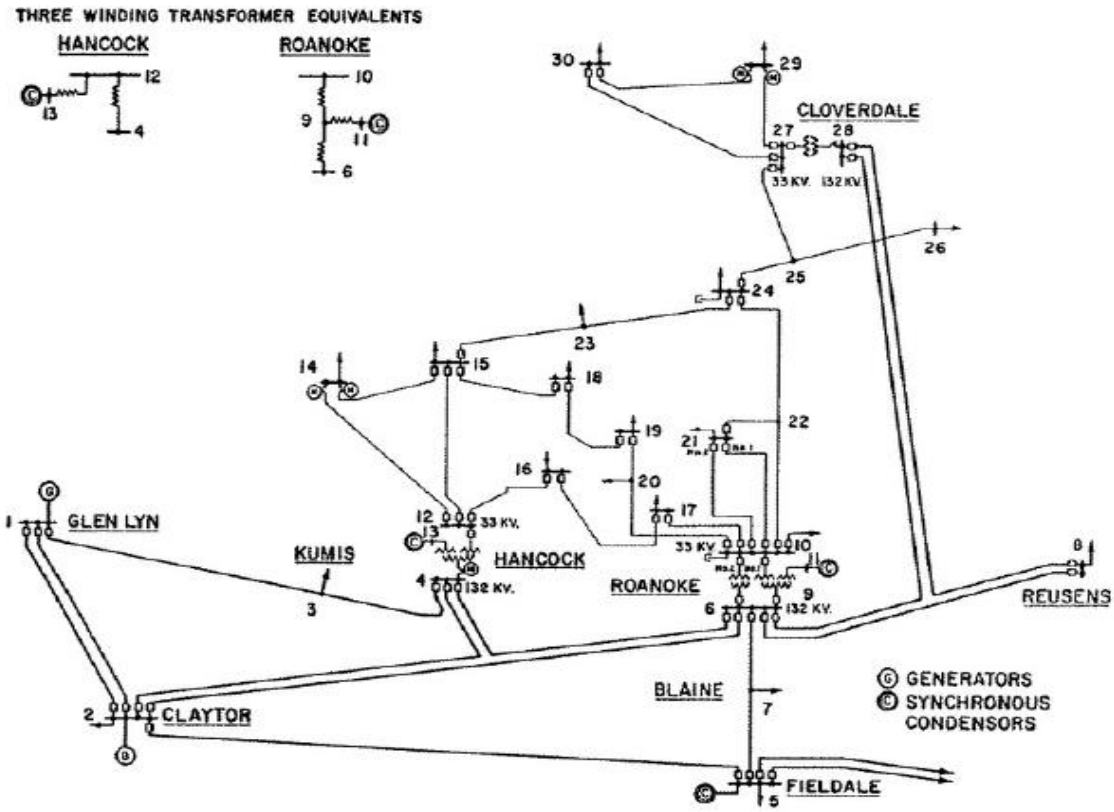


**Fig. I-C: BUS-CODE DIAGRAM OF 30 BUS SYSTEM**

**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

## 1. MATLAB Program for the optimization of benchmark functions using PSO

```
clc
disp('Function to be minimize f = (1.5-x1+x1*x2)^2 +(2.25-x1+x1*x2^2)^2
+ (2.625-x1+x1*x2^3)^2 i.e. beale's function');
%disp('Function to be minimize f = (x1+2*x2-7)^2 +(2*x1+x2-5)^2 i.e.
booth's function');
%disp('Function to be minimize f = 100(x1^2-x2)^2+(1-x1)^2 i.e.
rosenbrock function');
%disp('Function to be minimize f = x1^2 + x2^2 i.e. Sphere function');
p=input('Enter the no. of particles in a swarm= ');
it=input('Enter the maximum no. of iterations to be performed= ');
T=input('Enter the tolerance value= ');
tic;
% Initialization of variables
x1=zeros(p,it);
x2=zeros(p,it);
v1=zeros(p,it);
v2=zeros(p,it);
f=zeros(p,it);
fp=zeros(1,p);
df=zeros(1,(it-1));
r1=0.5;
r2=0.5;
c1=2;
c2=2;
%c1=2.05 %when constriction factor is used
%c2=2.05 %when constriction factor is used
%Initial random position and velocity selection
x1(:,1)=unifrnd(-4.5,4.5,1,p);
x2(:,1)=unifrnd(-4.5,4.5,1,p);
v1(:,1)=rand(1,p);
v2(:,1)=rand(1,p);
print = [x1(:,1) x2(:,1) v1(:,1) v2(:,1) f(:,1)];
disp('     x1        x2        v1        v2        f')
disp(print)
% fitness evaluation
for j=1:p
f(j,1)=(1.5-x1(j,1)+x1(j,1)*x2(j,1))^2 + (2.25-
x1(j,1)+x1(j,1)*x2(j,1)^2)^2 + (2.625-x1(j,1)+x1(j,1)*x2(j,1)^3)^2;
%f(j,1) = (x1(j,1)+2*x2(j,1)-7)^2 +(2*x1(j,1)+x2(j,1)-5)^2 ;
%f(j,1)= 100*((x1(j,1))^2 - x2(j,1) )^2 + (1- x1(j,1))^2 ;
%f(j,1)= x1(j,1)^2 + x2(j,1)^2 ;
end
%Initial personal best values
x1p=x1(:,1);
x2p=x2(:,1);
%for Initial Global best values
fmin=min(f(:,1));
for j=1:p
    if f(j,1)==fmin
        x1g = x1(j,1);
        x2g = x2(j,1);
    else
    end
end
```

```
%Updation by PSO algorithm
for i=1:it
disp(sprintf('This is %d no. of iteration',i))
%for inertia weight W
  wmax=0.9;
  wmin=0.4;
%Linearly Decreasing inertia weight W
  w = wmax-i*((wmax-wmin)/it);
%constriction factor
  cf = 0.729;
for j=1:p
%For velocity updation
%Only active inertia weight(IW-PSO), IPSO-B, IPSO-C
v1(j,(i+1)) = w*v1(j,i) + r1*c1*(x1p(j)-x1(j,i)) + r2*c2*(x1g-x1(j,i));
v2(j,(i+1)) = w*v2(j,i) + r1*c1*(x2p(j)-x2(j,i)) + r2*c2*(x2g-x2(j,i));
%Only constriction factor(CF-PSO)
%v1(j,(i+1))= cf *(v1(j,i)+r1*c1*(x1p(j)-x1(j,i))+r2*c2*(x1g-x1(j,i)));
%v2(j,(i+1))= cf *(v2(j,i)+r1*c1*(x2p(j)-x2(j,i))+r2*c2*(x2g-x2(j,i)));
%Both inertia weight and constriction factor (CI-PSO)
%v1(j,(i+1))=cf*(w*v1(j,i)+r1*c1*(x1p(j)-x1(j,i))+r2*c2*(x1g-x1(j,i)));
%v2(j,(i+1))=cf*(w*v2(j,i)+r1*c1*(x2p(j)-x2(j,i))+r2*c2*(x2g-x2(j,i)));
%For IPSO-A
 %kk=1;
  %if f(j,i)==fmin
  %   kk= 0;
  %end
%v1(j,(i+1))= w*v1(j,i) + r1*c1*(x1p(j)-x1(j,i)) + r2*c2*(x1g-x1(j,i));
%v2(j,(i+1))= w*v2(j,i) + r1*c1*(x2p(j)-x2(j,i)) + r2*c2*(x2g-x2(j,i));

%For Position updation
%IW-PSO, CF-PSO, CI-PSO, IPSO-A
x1(j,(i+1)) = x1(j,i) + v1(j,(i+1));
x2(j,(i+1)) = x2(j,i) + v2(j,(i+1));
%IPSO-B
 %kk=1;
  %if f(j,i)==fmin
  %   kk= 0;
  %end
%x1(j,(i+1)) = x1(j,i) + kk*v1(j,(i+1));
%x2(j,(i+1)) = x2(j,i) + kk*v2(j,(i+1));
%IPSO-C
 %kk=1;
  %if f(j,i)<fp(j)
  %   kk= 0;
  %end
%x1(j,(i+1)) = x1(j,i) + kk*v1(j,(i+1));
%x2(j,(i+1)) = x2(j,i) + kk*v2(j,(i+1));

%Fitness evaluation
f(j,(i+1))=(1.5-x1(j,(i+1))+x1(j,(i+1))*x2(j,(i+1)))^2+(2.25-
x1(j,(i+1))+ x1(j,(i+1))*x2(j,(i+1))^2)^2 +(2.625-
x1(j,(i+1))+x1(j,(i+1))*x2(j,(i+1))^3)^2;
%f(j,(i+1)) = (x1(j,(i+1))+2*x2(j,(i+1))-7)^2
+(2*x1(j,(i+1))+x2(j,(i+1))-5)^2;
%f(j,(i+1))= 100*( (x1(j,(i+1)))^2 - x2(j,(i+1)) )^2 + (1-
x1(j,(i+1)))^2 ;
%f(j,(i+1))=  x1(j,(i+1))^2 + x2(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)= (1.5-x1p(j)+x1p(j)*x2p(j))^2+(2.25-
x1p(j)+x1p(j)*x2p(j)^2)^2+(2.625-x1p(j)+x1p(j)*x2p(j)^3)^2;
 %fp(j)=(x1p(j)+2*x2p(j)-7)^2 +(2*x1p(j)+x2p(j)-5)^2;
 %fp(j)= 100*( (x1p(j))^2 - x2p(j) )^2 + (1- x1p(j))^2;
 %fp(j)= x1p(j)^2 + x2p(j)^2;
 end
 for j=1:p
   if f(j,i+1)< fp(j)
        x1p(j)=x1(j,i+1);     %personal best values
        x2p(j)=x2(j,i+1);
        else
   end
  end

%for Global best values updation
   if min(f(:,(i+1)))<fmin
        fmin=min(f(:,(i+1)));
   else
   end
  for j=1:p
    if f(j,i+1)==fmin
         x1g = x1(j,i+1);     %global best values
         x2g = x2(j,i+1);
      else
    end
   end
print = [x1(:,i) x2(:,i) v1(:,i) v2(:,i) f(:,i)];
disp('     x1        x2        v1        v2        f')
disp(print)

%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
t=toc;
[r,c]=find(f==fmin);
minf = (1.5-x1g+x1g*x2g)^2 +(2.25-x1g+x1g*x2g^2)^2 +(2.625-
x1g+x1g*x2g^3)^2;
%minf = (x1g+2*x2g-7)^2 +(2*x1g+x2g-5)^2;
%minf = 100*( (x1g^2 - x2g )^2 + (1- x1g)^2);
%minf = (x1g^2 + x2g^2);
disp(sprintf('min value of function is %d and at values of x1=%d and
x2=%d ',minf,x1g,x2g));
disp(sprintf('time = %d ',t));
```

## 2. MATLAB Program for the solution of IEEE 5-bus system using PSO

```
clear all
clc
disp(' we have to minimize the cost function of a IEEE 5-BUS system')
p=input('Enter the no. of particles in a swarm= '); %no. of particles
it=input('Enter the no. of iterations= ');
tic; %time calculation
%Input Data
a=10^(-4)*[50 50];
b=10^(-2)*[351 389];
c=[44.4 40.6];
B=10^(-2)*[0.0349 0.0086; -0.0055 0.0371];
%Initialization of variables
p1=zeros(p,it);
p2=zeros(p,it);
v1=zeros(p,it);
v2=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);
C=zeros(p,it);
r1=0.5;
r2=0.5;
c1=2;
c2=2;
pd=160;
p1g=zeros(p);
p2g=zeros(p);
fp=zeros(1,p);
plp=zeros(1,p);
k=50;
w1=1;
n=1;
while n==1
    for j=1:p
    p1(j,1)=unifrnd(30,120,1);
    p2(j,1)=pd-p1(j,1);
    if p2(j,1)<30&&p2(j,1)>120
        n=1;
        break;
    else
        n=0;
    end
    end
end
v1(:,1)=rand(1,p);
v2(:,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);
         C(j,1) = C1(j,1) + C2(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)]*B*[p1(j,1) p2(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)))+ k*abs(pd+pl(j,1)-p1(j,1)-p2(j,1));
end
%0th iteration data display
disp('this is the 0th iteration')
print0 = [p1(:,1) p2(:,1) v1(:,1) v2(:,1) f(:,1) C1(:,1) C2(:,1)
C(:,1)];
disp('      P1          P2          V1          v2        f        C1
C2         C ')
disp(print0)
%Initial personal besst values
p1p=p1(:,1);
p2p=p2(:,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);
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;
%Linearly Decreasing inertia weight W
w = wmax-i*((wmax-wmin)/it);
% for constraction Factor
 cf = 0.729;
%For calculatiing velocities for updation
 for j=1:p
 kk=1;
 %if f(j,i)==fgm
  %  kk= 0;
 %end
v1(j,(i+1)) = kk*(w*v1(j,i) + r1*c1*(p1p(j)-p1(j,i)) + r2*c2*(p1g(j)-
p1(j,i)));
v2(j,(i+1)) = kk*(w*v2(j,i) + r1*c1*(p2p(j)-p2(j,i)) + r2*c2*(p2g(j)-
p2(j,i)));
 %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 v1(j,(i+1))> 60
```

```
            v1(j,(i+1))= 60;
        end
        if v2(j,(i+1))> 60
            v2(j,(i+1))= 60;
        end
 end
%Updation of p values
for j=1:p
         kk=1;
        % if f(j,i)==fgm
         %   kk= 0;
        %end
        %if f(j,i)<fp(j)
         %   kk= 0;
        %end
        p1(j,(i+1)) = p1(j,i) + kk*v1(j,(i+1));
        p2(j,(i+1)) = p2(j,i) + kk*v2(j,(i+1));
 end
 %Pmin and Pmax constraint
 for j=1:p
        if p1(j,(i+1))< 30
            p1(j,(i+1))= 30;
        end
        if p2(j,(i+1))< 30
            p2(j,(i+1))= 30;
        end
        if p1(j,(i+1))> 120
            p1(j,(i+1))= 120;
        end
        if p2(j,(i+1))> 120
            p2(j,(i+1))= 120;
        end
 end
 %For losses formulation (PL)
 for j=1:p
   pl(j,(i+1))= [p1(j,(i+1)) p2(j,(i+1))]*B*[p1(j,(i+1)) p2(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)))+
k*abs(pd+pl(j,(i+1))-p1(j,(i+1))-p2(j,(i+1)));
end
%personal best values updation
%For losses formulation (PL)
 for j=1:p
    plp(j)= [p1p(j) p2p(j)]*B*[p1p(j) p2p(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))) +  k*abs(pd+plp(j)-
p1p(j)-p2p(j));
 end
 for m=1:p
        if f(m,i)< fp(m)
            p1p(m)=p1(m,(i+1));
            p2p(m)=p2(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);
                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);
         C(j,(i+1)) = C1(j,(i+1)) + C2(j,(i+1));
    end
    %To find change in the values of f
    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)));
        csp(j,i)= abs(C(j,(i+1))-C(j,i));
    end
print=[p1(:,(i+1)) p2(:,(i+1)) v1(:,(i+1)) v2(:,(i+1)) f(:,(i+1))
C1(:,(i+1)) C2(:,(i+1))  C(:,(i+1))];
disp('P1     P2      V1      v2        f       c1       c2       C ')
disp(print)
%Stoping criterion (df(j,i)<=10^(-6))&& &&(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
t=toc;
disp('Initial values of generations of 2 generators')
initial=[p1(:,1) p2(:,1) ];
disp('    P1        P2    ')
disp(initial)
disp(sprintf('\nPD+Pl = %d',pd+pl(1,i)))
disp(sprintf('\nP1+P2=%d\n',p1(1,i)+p2(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('\nFinal values of generations of the three generators')
disp(sprintf('P1=%d',p1(1,i)))
disp(sprintf('P2=%d',p2(1,i)))
disp(sprintf('Computational Time = %d ',t));
```

### 3. MATLAB Program for the solution of IEEE 14-bus system using PSO

```
clear all
clc
disp('we have to minimize the cost function of IEEE 14-BUS system')
p=input('Enter the no. of particles in a swarm= ');%no. of particles
it=input('Enter the maximum no. of iterations to be performed= ');
tic; %time calculation
%Input Data
a=10^(-4)*[50 50 50];
b=10^(-2)*[245 351 389];
c=[105 44.4 40.6];
B=10^(-2)*[0.0349 0.0068 -0.0039; 0.0068 0.0157 0.0015; -0.0039 0.0015
0.0275];
B0=10^(-2)*[0.0044 0.0024 0.0000];
B00=2.5408*10^(-4);
%Initialization of variables
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);
r1=0.5;
r2=0.5;
c1=2;
c2=2;
pd=259;
p1g=zeros(p);
p2g=zeros(p);
p3g=zeros(p);
fp=zeros(1,p;
plp=zeros(1,p;
k=50;
w1=1;
n=1;
while n==1
    for j=1:p
    p1(j,1)=unifrnd(50,200,1);
    p2(j,1)=unifrnd(20,100,1);
    p3(j,1)=pd-p1(j,1)-p2(j,1);
    if p3(j,1)<20&&p3(j,1)>100
        n=1;
        break;
    else
        n=0;
    end
    end
end
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)]'
+[p1(j,1) p2(j,1) p3(j,1)]*B0'+B00;
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))) + k*abs(pd+pl(j,1)-p1(j,1)-p2(j,1)-p3(j,1));
end
%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;
%Linearly Decreasing inertia weight W
w = wmax-i*((wmax-wmin)/it);
% for constraction Factor
 cf = 0.729;
%For calculatiing velocities for updation
 for j=1:p
 kk=1;
 %if f(j,i)==fgm
  %  kk= 0;
 %end
 v1(j,(i+1)) = kk*(w*v1(j,i) + r1*c1*(p1p(j)-p1(j,i)) + r2*c2*(p1g(j)-
p1(j,i)));
 v2(j,(i+1)) = kk*(w*v2(j,i) + r1*c1*(p2p(j)-p2(j,i)) + r2*c2*(p2g(j)-
p2(j,i)));
```

```
 v3(j,(i+1)) = kk*(w*v3(j,i) + r1*c1*(p3p(j)-p3(j,i)) + r2*c2*(p3g(j)-
p3(j,i)));
 end
   %V(min) and V(max) constraint
    for j=1:p
        if v1(j,(i+1))< -25
            v1(j,(i+1))= -25;
        end
        if v2(j,(i+1))< -10
            v2(j,(i+1))= -10;
        end
        if v3(j,(i+1))< -10
            v3(j,(i+1))= -10;
        end
        if v1(j,(i+1))> 100
            v1(j,(i+1))= 100;
        end
        if v2(j,(i+1))> 50
            v2(j,(i+1))= 50;
        end
        if v3(j,(i+1))> 50
            v3(j,(i+1))= 50;
        end
    end
%Updation of p values
    for j=1:p
        kk=1;
       if f(j,i)==fgm
            kk= 0;
       end
       %if f(j,i)<fp(j)
        %    kk= 0;
       %end
        p1(j,(i+1)) = p1(j,i) + kk*v1(j,(i+1));
        p2(j,(i+1)) = p2(j,i) + kk*v2(j,(i+1));
        p3(j,(i+1)) = p3(j,i) + kk*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))< 20
            p2(j,(i+1))= 20;
        end
        if p3(j,(i+1))< 20
            p3(j,(i+1))= 20;
        end
        if p1(j,(i+1))> 200
            p1(j,(i+1))= 200;
        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))]'+[p1(j,(i+1)) p2(j,(i+1))
p3(j,(i+1))]*B0'+B00;
 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))) + 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)]'+[p1p(j)
p2p(j) p3p(j)]*B0'+B00;
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))) +
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
    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
```

```
%Stoping criterion &&(csp(j,i)<=10^(-6))  (df(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
t=toc;
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('\nFinal 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(sprintf('Computational Time = %d ',t));
```

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

```
clear all
clc
disp(' we have to minimize the cost function of IEEE 30-BUS system')
p=input('Enter the no. of particles in a swarm= '); %no. of particles
it=input('Enter the maximum no. of iterations to be performed= ');
tic; %time calculation
%Input Data
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];
%Initialization of variables
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);
```

```
r1=0.5;
r2=0.5;
c1=2;
c2=2;
pd=283.4;
p1g=zeros(p);
p2g=zeros(p);
p3g=zeros(p);
fp=zeros(1,p);
plp=zeros(1,p);
k=50;
w1=1;
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)=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))) + k*abs(pd+pl(j,1)-p1(j,1)-p2(j,1)-p3(j,1));
end
%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;
%Linearly Decreasing inertia weight W
w = wmax-i*((wmax-wmin)/it);
% for constraction Factor
 cf = 0.729;
%For calculatiing velocities for updation
 for j=1:p
 kk=1;
 %if f(j,i)==fgm
  %  kk= 0;
 %end
 v1(j,(i+1)) = kk*(w*v1(j,i) + r1*c1*(p1p(j)-p1(j,i)) + r2*c2*(p1g(j)-
p1(j,i)));
 v2(j,(i+1)) = kk*(w*v2(j,i) + r1*c1*(p2p(j)-p2(j,i)) + r2*c2*(p2g(j)-
p2(j,i)));
 v3(j,(i+1)) = kk*(w*v3(j,i) + r1*c1*(p3p(j)-p3(j,i)) + r2*c2*(p3g(j)-
p3(j,i)));
 end
   %V(min) and V(max) constraint
    for j=1:p
        if v1(j,(i+1))< -25
            v1(j,(i+1))= -25;
        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))> 125
            v1(j,(i+1))= 125;
        end
        if v2(j,(i+1))> 50
            v2(j,(i+1))= 50;
        end
        if v3(j,(i+1))> 50
            v3(j,(i+1))= 50;
        end
    end
%Updation of p values
    for j=1:p
        kk=1;
       if f(j,i)==fgm
            kk= 0;
       end
       %if f(j,i)<fp(j)
        %    kk= 0;
       %end
        p1(j,(i+1)) = p1(j,i) + kk*v1(j,(i+1));
        p2(j,(i+1)) = p2(j,i) + kk*v2(j,(i+1));
```

```
        p3(j,(i+1)) = p3(j,i) + kk*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))) +
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))) +
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
    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
    %Stoping criterion &&(csp(j,i)<=10^(-6))  (df(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
t=toc;
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('\nFinal 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(sprintf('Computational Time = %d ',t));
```

# REFERENCES

[1] Chem-Lin Chen and Shun-Chung Wang, "BRANCH-AND-BOUND SCHEDULING FOR THERMAL GENERATING UNITS", IEEE Trans. on Energy Conversion, Vol. 8, No. 2, June 1993

[2] K.Y. Lee., Y.M. Park. and J.L. Ortiz., "Fuel-cost minimisation for both real- and reactive-power dispatches", IEE Proceeding, Vol. 131, Pt. C, No. 3, May 1984

[3] J.P.Zhan, Q.H. Wu, C.X.Guo, and X.X. Zhou, "Fast -Iteration Method for Economic Dispatch With Prohibited Operating Zones", IEEE Trans. on power system, Vol. 29, No. 2, March 2014

[4] Rabih A. Jabr, Alun H. Coonick and Brian J. Cory, "A Homogeneous Linear Programming Algorithm for the Security Constrained Economic Dispatch Problem", IEEE Trans. on power system, Vol. 15, No. 3, August 2000

[5] Leandro dos Santos Coelho and Viviana Cocco Mariani, "Combining of Chaotic Differential Evolution and Quadratic Programming for Economic Dispatch Optimization With Valve-Point Effect", IEEE Trans. on power system, Vol.21, No.2, May 2006

[6] Ioannis G. Damousis, Anastasios G. Bakirtzis, and Petros S. Dokopoulos, "Network-Constrained Economic Dispatch Using Real-Coded Genetic Algorithm", IEEE Trans. on power system, Vol. 18, No. 1, Feb. 2003

[7] Aniruddha Bhattacharya and Pranab Kumar Chattopadhyay, "Biogeography-Based Optimization for Different Economic Load Dispatch Problems", IEEE Trans. on power system, Vol. 25, No. 2, May 2010

[8] Kuntal Bhattacharjee, Aniruddha Bhattacharya and Sunita Halder nee Dey, "Chemical reaction optimisation for different economic dispatch problems", IET Gener. Transm. Distrib.,2014, Vol.8, Iss.3, pp. 530–541

[9] Taher Niknam and Faranak Golestaneh, "Enhanced Bee Swarm Optimization Algorithm for Dynamic Economic Dispatch", IEEE Systems Journal, Vol. 7, No. 4, December 2013

[10] Taher Niknam, Rasoul Azizipanah, Abarghooee, and Jamshid Aghaei, "A New Modified Teaching-Learning Algorithm for Reserve Constrained Dynamic Economic Dispatch", IEEE Trans. on power system, Vol. 28, No. 2, May 2013

[11] Irina Ciornei, and Elias Kyriakides, “A GA-API Solution for the Economic Dispatch of Generation in Power System Operation”, IEEE Trans. on power system, Vol. 27, No. 1, February 2012

[12] Aniruddha Bhattacharya and Pranab Kumar Chattopadhyay, “Hybrid Differential Evolution With Biogeography-Based Optimization for Solution of Economic Load Dispatch”, IEEE Trans. on power system, Vol. 25, No. 4, Nov. 2010

[13] Jong-Bae Park, Ki-Song Lee, Joong-Rin Shin, and Kwang Y. Lee, “A Particle Swarm Optimization for Economic Dispatch With Nonsmooth Cost Functions”, IEEE Transaction on power system, Vol. 20, No. 1, Feb. 2005

[14] Leandro dos Santos Coelho and Chu-Sheng Lee, “Solving economic load dispatch problems in power systems using chaotic and Gaussian particle swarm optimization approaches”, Electrical Power and Energy Systems 30 (2008) 297–307

[15] Vahid Hosseinnezhad and Ebrahim Babaei, “Economic load dispatch using θ-PSO”, Electrical Power and Energy Systems 49 (2013) 160–169

[16] M.A. Abido, “Multiobjective particle swarm optimization for environmental/economic dispatch problem”, Electric Power Systems Research 79 (2009) 1105–1113

[17] Ke Meng, Hong Gang Wang, ZhaoYang Dong, and Kit Po Wong, “Quantum-Inspired Particle Swarm Optimization for Valve-Point Economic Load Dispatch”, IEEE Trans. on power system, Vol. 25, No. 1, Feb. 2010

[18] T.O. Ting, M.V.C. Rao and C.K.Loo, “A Novel Approach for Unit Commitment Problem via an Effective Hybrid Particle Swarm Optimization”, IEEE Transaction on power system, Vol. 21, No. 1, February 2006

[19] S. Chakraborty, T. Senjyu, A. Yona, A.Y. Saber and T. Funabashi, “Solving economic load dispatch problem with valve-point effects using a hybrid quantum mechanics inspired particle swarm optimization”, IET Gener. Transm. Distrib., 2011, Vol. 5, Iss. 10, pp. 1042–1052

[20] James Kennedy and Russell Eberhart, “Particle Swarm Optimization”, 0-7803-2768-3/95/$4.00 1995 IEEE

[21] Ioan Cristian Trelea, "The particle swarm optimization algorithm: convergence analysis and parameter selection", Information Processing Letters 85 (2003) 317–325

[22] Maurice Clerc and James Kennedy, "The Particle Swarm—Explosion, Stability, and Convergence in a Multidimensional Complex Space", IEEE Trans. on evolutionary computation, Vol. 6, No.1, February 2002

[23] Yuhui Shi and Russell Eberhart, "A Modified Particle Swarm Optimizer", 0-7803-4869-9198 $10.0001998 IEEE

[24] R. C. Eberhart and Y. Shi, "Comparing Inertia Weights and Constriction Factors in Particle Swarm Optimization", 0-7803-6375-2/00/$10.00 02000 IEEE.

[25] Frans van den Bergh and Andries P. Engelbrecht, "A Cooperative Approach to Particle Swarm Optimization", IEEE Trans. on evolutionary computation,Vol. 8, No. 3, June 2004

[26] Jang-Ho Seo, Chang-Hwan Im, Chang-Geun Heo, Jae-Kwang Kim, Hyun-Kyo Jung and Cheol-Gyun Lee4, "Multimodal Function Optimization Based on Particle Swarm Optimization", IEEE Trans. on magnetics, Vol. 42, No. 4, April 2006

[27] J. J. Liang, A. K. Qin, Ponnuthurai Nagaratnam Suganthan and S. Baskar, "Comprehensive Learning Particle Swarm Optimizer for Global Optimization of Multimodal Functions", IEEE Trans. on evolutionary computation, Vol. 10, No. 3, June 2006

[28] Jin S. Heo, Kwang Y. Lee and Raul Garduno-Ramirez, "Multiobjective Control of Power Plants Using Particle Swarm Optimization Techniques", IEEE Trans. on energy conversion, Vol. 21, No. 2, June 2006

[29] Yamille del Valle, Ganesh Kumar Venayagamoorthy, Salman Mohagheghi, Jean-Carlos Hernandez, and Ronald G. Harley, Particle Swarm Optimization: Basic Concepts, Variants and Applications in Power Systems, IEEE Trans. on evolutionary computation, Vol. 12, No. 2, April 2008

[30] C.L. WADHWA, "Electrical Power Systems", New Age International(P) Limited, Publishers, 6th Edition, Page 628-661