EQUAL PERCENTAGE SAVING BASED MULTI-OBJECTIVE ECONOMIC LOAD DISPATCH USING PARTICLE SWARM OPTIMIZATION

A DISSERTATION SUBMITTED IN THE PARTIAL FULFILLMENT FOR THE DEGREE OF

MASTER OF TECHNOLOGY
(POWER SYSTEM)
(2011 – 2013)

Submitted by

AISHWARY JAIN
2K11/PSY/01
M.Tech. (Power System)

Under the guidance of

Prof. N.K. Jain
&
Prof. Uma Nangia

Electrical Engineering Department

Delhi Technological University
(Formerly Delhi College of Engineering)

CERTIFICATE

It is certified that **Mr. AISHWARY JAIN**, Roll No 2K11/PSY/01, student of M.TECH., Power System, Department of Electrical Engineering, Delhi Technological University (Formerly Delhi College of Engineering), has submitted the dissertation entitled **"EQUAL PERCENTAGE SAVING BASED MULTI-OBJECTIVE ECONOMIC LOAD DISPATCH USING PARTICLE SWARM OPTIMIZATION"** under our guidance towards partial fulfillment of the requirements for the award of the degree of Master of Engineering (Power System).

This dissertation is a bonafide record of project work carried out by him under our guidance and supervision. His work is found to be outstanding and has not been done earlier.

We wish him success in all his endeavors.

New Delhi
JULY , 2013

Prof. N.K Jain
Electrical Engineering Department
Delhi Technlogical University
(Formerly Delhi College of Engineering

Prof. Uma Nangia
Electrical Engineering Department
Delhi Technlogical University
(Formerly Delhi College of Engineering

ACKNOWLEDGEMENT

The writing of this dissertation has been one of the most significant academic challenges I have ever had to face; without GOD's blessings and support, patience and guidance of the following people, this study would not have been completed. It is to them that 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 in this field of research, for their valuable guidance, encouragement and affection for the successful completion of this work. Their sincere sympathies and kind attitude always encouraged me to carry out the present work firmly.
- **Prof. Narendra Kumar,** Ex-HOD, Electrical Engineering Department and **Prof. Madhusudan Singh,** HOD, Electrical Engineering Department, Delhi Technological University (Formerly Delhi College of Engineering), New Delhi, for providing me with the best facilities in the Department and timely suggestions.
- My Parents and Vaibhav Jain, my elder brother, who have always supported, encouraged and believed in me and patiently waited for my dreams come true.

AISHWARY JAIN
2K11/PSY/01
M.TECH. (PSY)

ABSTRACT

In this thesis, Multi-objective Economic Load Dispatch (MELD) considering Cost of generation and System transmission losses has been formulated using Priority Goal Programming (PGP). Equality constraint of the problem has been considered in the PGP formulation by inclusion of parameter K in the multiobjective function. The non-inferior set has been generated for IEEE 5, 14 and 30-bus systems using one of the intelligent optimization techniques, Particle Swarm Optimization (PSO) for the first time. Extensive trade off analysis has been carried out and an attempt has been made to achieve the Target Point based on equal percentage saving in both the objectives.

A MATLAB program has been developed for Evolutionary Programming and Evolutionary Computation such as Particle Swarm Optimization (PSO) to solve economic load dispatch problem considering cost of generation and transmission losses.

LIST OF FIGURES:

LIST OF TABLES:

TABLE OF CONTENT

CHAPTER 1:

INTRODUCTION

1.1 OVERVIEW

Economic load dispatch (***ELD***) is an important function in power system planning and operation. The basic object of economic load dispatch is the distribution of total generation of power in the network such that the cost of power delivered is minimum. By economic load dispatch we mean to find the generation of the different generators or plants so that the total fuel cost is minimum, and at the same time the total demand and the losses at any instant must be met by the total generation. In case of economic load dispatch the generations are not fixed but they are allowed to take values again within certain limits so as to meet a particular load demand with minimum fuel consumption. This means economic load dispatch problem is really the solution of large number of load flow problems and choosing the one which is optimum in the sense that it needs minimum cost of generation.

In this work two objectives of power systems- Cost of generation and System transmission losses have been considered and Multi-objective Economic Load Dispatch (MELD) problem has been formulated using Priority Goal Programming (PGP) technique. The non-inferior sets for IEEE 5, 14 and 30 bus systems have been generated using Particle Swarm Optimization (PSO) which is one of the intelligent optimization techniques. The Target Points have been achieved based on equal percentage saving of all the objectives.

1.2 OBJECTIVES AND METHODOLOGY

Our objective in this work is to solve multi-objective economic load dispatch (MELD) problem considering cost of generation and system transmission losses. For this IEEE 5, 14 & 30 bus systems have been considered. The target point or the best compromise solutions have been achieved by using equal percentage saving method in which we have calculated the percentage saving in the cost of generation and the system transmission losses.

The work has been carried out in the following order:

a. Exploring Particle Swarm Optimization and coding its algorithm in MATLAB R2010a.
b. Solution of various mathematical benchmark functions using PSO.
c. Formulation of Multi-objective Economic Load Dispatch (MELD) considering cost of generation and system transmission losses for IEEE 5, 14 & 30 Bus Systems using PGP technique.

d. Generation of non-inferior sets of IEEE 5, 14 and 30-bus systems using PSO.
e. Achievement of target point (TPs) for IEEE 5, 14 and 30 bus systems, based on equal percentage saving method.

1.3 LITERATURE SURVEY

1.3.1 MULTI-OBJECTIVE ECONOMIC LOAD DISPATCH

Economic load dispatch (ELD) is one of the major issues in power system operation [1]. It is defined as a process of allocating the output of generators to satisfy electrical demand in a power system in the most economic way considering all constraints [2]. The complexity of the ELD problem depends upon many factors, such as the size of the system, system constraints, and generator characteristics.

Multiobjective techniques are used to generate and evaluate more than one alternative. These techniques indicate to decision makers a range of choices beyond one ***optimal*** alternative identified by single objective method. A general rule for decision making is that more information ***carefully presented*** is better than less information. The decision to accept or reject a single optimal alternative is an **uninformed decision.** Informed decision making requires knowledge of full range of possibilities provided by multiobjective analysis. Multiobjective analysis allows several non-commensurable effects to be treated without artificially combining them.

Several techniques have been introduced to solve the optimization of ELD, which can be divided into conventional and stochastic methods. Conventional methods use a deterministic approach, such as the LaGrange multiplier [3], Linear Programming (LP) [4] and Dynamic Programming (DP) [5]. These methods have limitations or drawbacks when coping with more complex problems.

Recent techniques have been developed using stochastic approaches for solving optimization problems. Examples are an Adaptive Hopfield Neural Network [6], the Simulated Annealing method [7], Genetic Algorithms (GA) [8], Particle swarm Optimization [9] and Ant colony Optimization [10] amongst others. These new methods offer alternative techniques which attempt to overcome the drawbacks of conventional methods.

The optimal power system operation is achieved when both the objectives of power systems i.e. cost of generation and system transmission losses simultaneously attain their minimum values. But these objectives are conflicting in nature and cannot be handled by conventional single objective optimization techniques. Single objective optimization techniques give optimal solution in respect of a single aspect, i.e. they give the best value of the objective function under consideration. The values of other objectives at such a solution may be intolerably bad. In such a situation there is no other solution to facilitate the decision making process. The way out, therefore lies in the multiobjective approach [11-24] to problem solving. Uma Nangia, N. K. Jain and C. L.Wadhawa [11- 22] proposed the multi-objective load flow studies using various intelligent optimisation techniques and various objectives like cost of generation, system transmission losses, emission etc. Abido et. al [23] solved the multi-objective load dispatch problem using evolutionary algorithm.

1.3.2 PARTICLE SWARM OPTIMIZATION

Particle swarm optimization (PSO) is a recently proposed population based stochastic optimization algorithm, which is inspired by the social behaviors of animals like fish schooling and bird flocking.

PSO technique has been successfully applied to solve economic load dispatch as well as multiobjective economic load dispatch problem. Jong et. al [25] solved practical ELD problem with valve–point and multi-fuel effects by using modified PSO (MPSO) technique. Selvakumar et. al [26] developed new PSO (NPSO) approach to solve nonconvex Economic Dispatch problem. Amita et. al [27] have summarized the work carried out in the field of economic load dispatch using PSO. Ke Meng et. al [28] proposed Quantum- inspired PSO (QPSO) which has stronger search ability and faster convergence to solve ELD problem with nonsmooth cost function. Hybrid quantum based multi-population PSO (HQPSO) was proposed by Chakraborty et. al [29] to explore more search space for ELD problem with non-linear constraints. Niknam and Golestaneh [30] developed enhanced PSO to solve dynamic economic dispatch problem. An enhanced particle swarm optimization (EPSO) was proposed by Chao-Ming and Fu-Lu Wang [31] to solve real-time power dispatch problem considering emission, fuel cost and power wheeling cost as the objectives. Niknam and Doagou [32] presented modified adaptive θ-particle swarm optimization algorithm to solve multiobjective economic emission dispatch problem.

Niknam et. al [33] also developed improved particle swarm optimization (IPSO) to solve multiobjective optimal power flow problem considering cost, loss, voltage stability and emission impacts as the objective functions.

In this thesis, two important objectives of power systems – cost of generation and system transmission losses have been considered. Multi-objective Economic Load Dispatch (MELD) problem has been formulated using Priority Goal Programming (PGP) technique. The non-inferior set has been generated by using one of the intelligent optimization techniques-Particle Swarm Optimization (PSO) and the target points have been obtained by equal percentage saving method.

1.4 PLAN OF THESIS

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

Chapter 1: Discusses the introduction to economic load dispatch and Research objectives of the thesis. Literature survey of the covered topics has also been presented.

Chapter 2: Discusses the Multiobjective optimization. This presents formulation of general multiobjective problem, the concept of Non-inferiority, and PGP technique, which has been used to formulate Multi-objective Economic Load Dispatch problem.

Chapter 3: Presents the Particle Swarm Optimization and its applications.

Chapter 4: Explores the concepts of Particle Swarm Optimization algorithm in MATLAB R2010a. Analysis of various parameters in PSO algorithm has been carried out.

Chapter 5: Discusses Multiobjective Approach to Economic Load Dispatch and deals with problem formulation in 2-D space for IEEE 5, 14 and 30 bus systems. Non-inferior sets and Target Points for IEEE 5, 14 and 30- bus systems are presented.

Chapter 6: Conclusion and the prospects for Future Directions have been discussed.

Appendix and references are at the end of the thesis.

CHAPTER 2:

MULTIOBJECTIVE OPTIMIZATION

2.1 INTRODUCTION

The various objectives of power systems are cost of generation, system transmission losses, environmental pollution, security etc. Most of these objectives are conflicting in nature and cannot be handled by conventional single objective optimization techniques. Single objective optimization techniques give optimal solution in respect of an objective function under consideration and the values of other objectives at such a solution may be intolerably bad. Therefore, the easy answer to this problem is the multiobjective approach to problem solving.

2.1.1 ADVANTAGES OF MULTIOBJECTIVE PLANNING

The consideration of many objectives in the planning process accomplishes three major improvements in the problem solving:

i. Multiobjective programming and planning promotes more appropriate roles for the participants in the planning and decision making process.
ii. A wider range of alternatives is usually identified.
iii. The power system analyst's perception of a problem will be more realistic if many objectives are considered.

There are two parts of multiobjective decision making process; ***analysis*** and ***decision making.*** Analysis of a problem provides information about the problem for making decisions. Multiobjective approaches pursue an important decision making process: an explicit consideration of the relative impacts of the different objectives on the solution of the problem. These approaches emphasize the range of choice associated with a decision problem. The responsibility of assigning relative values to the objectives remains with the decision maker. The beauty of multiobjective approaches is that these provide sufficient information and facilitate the decision making process.

Regardless of the actual nature of decision making process, multiobjective approaches can be useful in promoting the explicit consideration of the value judgments which are implicitly made in the application of single objective approaches.

Multiobjective techniques are used to generate and evaluate more than one alternative. These techniques indicate to decision makers a range of choices beyond one ***optimal*** alternative

identified by single objective method. A general rule for decision making which is assumed is that more information ***carefully presented*** is better than less information. The decision to accept or reject a single optimal alternative is an **uninformed decision.** Informed decision making requires knowledge of full range of possibilities provided by multiobjective analysis. Multiobjective analysis allows several noncommensurable effects to be treated without artificially combining them.

2.2 FORMULATION OF GENERAL MULTIOBJECTIVE PROGRAMMING PROBLEM

The general multiobjective optimization problem with n decision variables, m constraints and h objectives is

Minimize

$$Z(X_1, X_2,\ldots\ldots\ldots X_n) = [Z_1 (X_1, X_2,\ldots\ldots\ldots..X_n); \quad Z_2 (X_1, X_2,\ldots\ldots\ldots...X_n); \quad \ldots\ldots\ldots\ldots\ldots\ldots; \quad Z_h(X_1, X_2,\ldots\ldots..\ldots\ldots X_n)] \tag{2.2a}$$

subject to

$$g_i(X_1,X_2,\ldots..X_n) \leq 0 \qquad i= 1,2,\ldots\ldots\ldots q \tag{2.2b}$$

$$U_j \geq 0 \qquad i= q+1, q+2,\ldots\ldots\ldots m \tag{2.2c}$$

Where $Z(X_1,X_2,\ldots..X_n)$ is the multi-objective function and $Z_1(X_1,X_2\ldots\ldots.X_n)$, $Z_2(X_1,X_2,\ldots\ldots X_n),\ldots\ldots\ldots Z_h(X_1,X_2,\ldots\ldots.X_n)$ are the h individual objective functions. In the multiobjective function Z, the various individual functions $Z_1,Z_2,\ldots\ldots.Z_h$ have just been written, but it does not imply any kind of operation say multiplication, addition or anything else whatsoever in general. In particular, Z can be designed to incorporate Z_1, Z_2 $\ldots.Z_h$ depending upon the approach.

Multiobjective approach to economic load dispatch has been carried out on IEEE 5, 14 and 30 bus systems [34] in 2-D space. The data of IEEE 5, 14 and 30 bus systems is given in {appendix I}. In 2–D space, two objectives i.e. cost of generation (F_C) and system transmission losses (F_L) have been considered.

The ideal situation where one would like to operate the power systems is one where both the objectives i.e. cost of generation (F_C) and system transmission losses (F_L) are minimum. Such a point is called the ***Ideal Point.*** In 2-D space, it is represented by (F_{CMIN}, F_{LMIN}). But such a point is not feasible. If it was, then there would not be any conflict among the objectives.

A strategy has to be adopted by the power system operator to achieve optimum values as per his satisfaction level and requirements. The operating point so obtained is called ***Target Point*** (TP) or the best-compromise solution.

2.3 NON-INFERIORITY

A feasible solution to a multiobjective programming problem is non-inferior, if there exists no other feasible solution that will yield an improvement in one objective without causing degradation in at least one of the other objectives [35]. A given non inferior solution may or may not be acceptable to the decision maker. However, it is important to note that, it is one of these non-inferior solutions for which decision maker looks for.

2.3.1 GRAPHICAL EXPLANATION OF NON-INFERIORTY

Let us explain this definition graphically. An arbitrary collection of feasible alternatives for a two objective minimization problem is shown in Fig 2.1. Curve 1 forms the boundary of the feasible region. The definition of non-inferiority can be used to find non-inferior solutions in Fig 2.1. All the feasible solutions above curve 1 are inferior because they yield more of both Z_1 (F_C) and Z_2 (F_L). Consider an exterior point C in Fig 2.1, which is inferior. Alternative A gives less of Z_1 (F_C) than does C without increasing the amount of Z_2 (F_L). Alternative B gives less amount of Z_2 (F_L) without increasing the amount of Z_1 (F_C). Consider point D on curve AB. Suppose it is desired to achieve lesser value of Z_1 (F_C) than the value at point D. Since it is not desirable to move to the left of curve AB as even through it gives lesser value of Z_1 (F_C), yet it lies in the infeasible region. Therefore, it is desirable to move upward only along the curve AB to have lesser value of Z_1 (F_C). Let us say, we get point E. At this point, we get less value of Z_1 (F_C) but there is some increase in Z_2 (F_L). In other words, in order to gain on Z_1 (F_C), we have to sacrifice ΔZ_2 (ΔF_L) units of Z_2 (F_L). Similarly, in moving from D to F, we have to sacrifice ΔZ_1 (ΔF_C)

units of Z_1 (F_C) to gain on Z_2 (F_L). So we can say that points D, E and F are non-inferior. The mathematical statement of non-inferiority also is given in {Appendix II}.

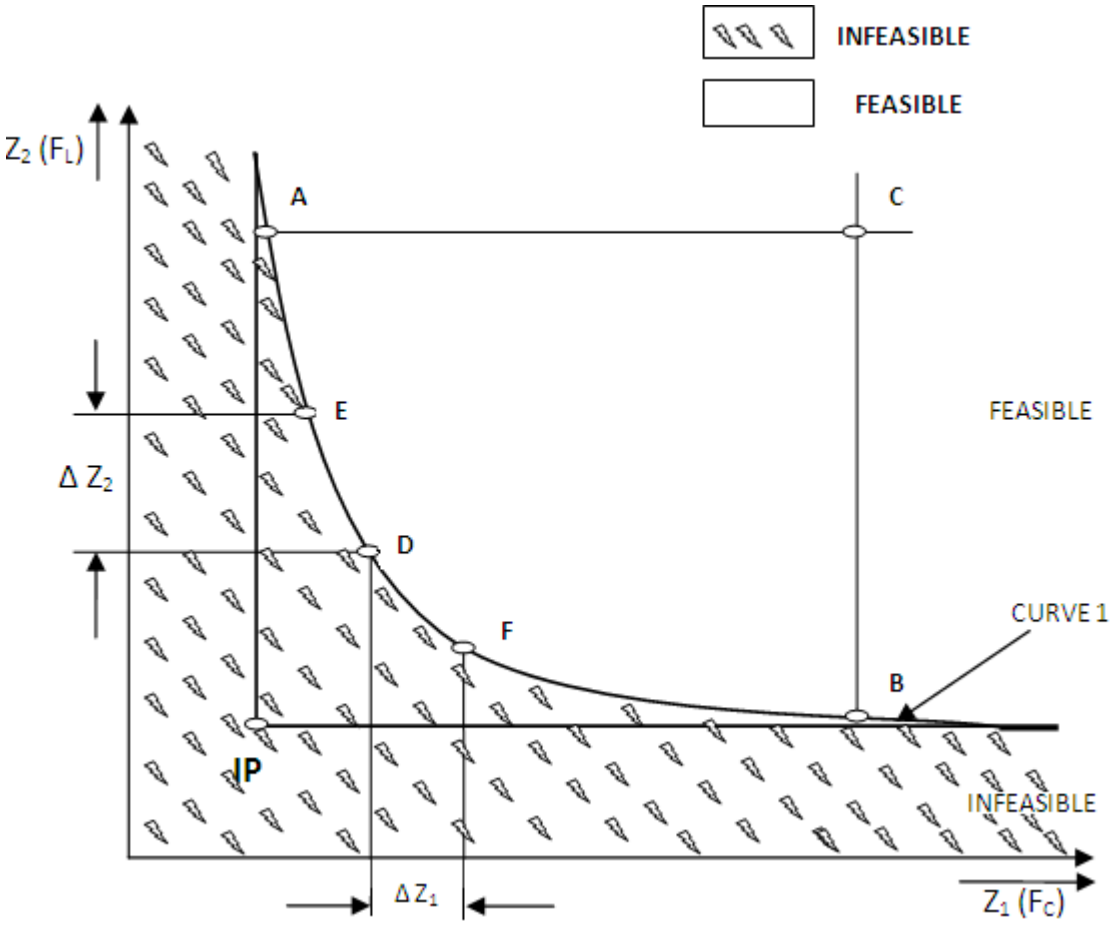


Fig 2.1: Graphical Explanation of Non-Inferiority

2.4 PRIORITY GOAL PROGRAMMING

Priority goal programming [12, 13] is a mathematical programming technique in which a number of objective functions are minimized/ maximized simultaneously. Sometimes, one may be confronted with important decision making problems. Correct decisions become important because of social, economic, ecological and various other reasons. In general, multiobjective decision making problems involve four important elements, namely, the set of alternatives available, the set of criteria for making decisions, the outcome of each of the choices, and finally the preference structure of the decision maker.

Human goal seeking behaviour may involve two basic concepts: (i) the satisfying model and (ii) compromise solutions such as goal programming. The satisfying model is the procedure of identifying a satisfactory solution, where the optimal solution may be assumed to have been reached i.e. the problem solver should be happy or at least satisfied to have achieved an acceptable solution. As the problem is to achieve more than one goal, which it may not be possible to achieve simultaneously, a compromise among the various goals is sought. As a matter of compromise the problem solver may be happy to have achieved the best possible solution.

The ideal situation where one would like to operate the Power Systems is one where both the objectives i.e. cost of generation (F_C) and system transmission losses (F_L) are minimum. Such a point is called the ***Ideal Point*** and is represented by (F_{Cmin}, F_{Lmin}) in 2D space. But such a point is not feasible as the objectives are conflicting in nature. Therefore, while considering multiobjective problems a strategy has to be adopted by the analyser to achieve optimum values as per his satisfaction level and requirements. Such a point is named Target Point (TP) or best-compromise solution. This has been achieved by equal percentage saving method.

Also a, special consideration has been given to equality constraint of the MELD problem. The multiobjective function has been modified by the inclusion of a penalty parameter K. The inequality constraints have been considered in the PSO programming.

2.5 ACHIEVEMENT OF TARGET POINT BY EQUAL PERCENTAGE SAVING METHOD

Target point has been achieved by determining the percentage savings in the all objectives of MELD problem. It has been defined as the one where percentage savings of all objectives become equal. The percentage savings represent the trade-off among various objectives i.e. the amount of gain being achieved in one objective by sacrificing one unit of other objective. Trade-off enables us to measure these gain and sacrifice quantitatively.

For calculation of percentage savings in F_C and F_L, the algorithm for the procedure is as follows:

1. First calculate the maximum possible savings in F_C and F_L (i.e. MPSC and MPSL) by keeping W_C=1 & W_L=0 and W_C=0 & W_L=1 respectively.
2. Secondly compute the savings in F_C and F_L for all the non-inferior set generated.
3. Compute the savings of all the non-inferior sets as the percentages of the respective maximum possible saving.

$$PSC = \frac{SFC}{MPSC} * 100 \quad (2.5a)$$

where;

PSC = Percentage Saving in Cost

$SFC = F_{Cmax} - F_C$

$MPSC = F_{Cmax} - F_{Cmin}$

$$PSL = \frac{SFL}{MPSL} * 100 \quad (2.5b)$$

where;

PSL = Percentage Saving in Loss

$SFL = F_{Lmax} - F_L$

$MPSL = F_{Lmax} - F_{Lmin}$

4. The target point is one for which the PSC and PSL are equal (i.e. equal compromise in both the savings of F_C and F_L).

The non-inferior sets are generated by using PGP technique method. Different priorities are assigned to different objectives in a multi-objective optimization problem to study the effect of various objectives on the optimized solution of a particular MELD problem. By assigning different weights to different objectives we change the priority given to one objective with respect to other objectives, which alters the results of the optimization problem. The non-inferior sets generated are analyzed by the percentage change in the results for different objectives of the MELD problem. Now the target point can be chosen from this set of non-inferior solution, on the basis of the satisfaction level and compromise between the conflicting objectives.

CHAPTER 3:

PARTICLE SWARM OPTIMIZATION

3.1 INTRODUCTION

PSO is a population-based, self-adaptive, stochastic optimization technique [36, 37]. The basic idea of PSO is the mathematical modelling and simulation of food searching activities of a swarm of birds (particles). In the multidimensional space where the optimal solution is sought, each particle in the swarm is moved towards the optimal point by adding a velocity to its position. The velocity of a particle is influenced by three components, namely, inertial, cognitive, and social The inertial component simulates the inertial behaviour of the bird to fly in the previous direction. The cognitive component models the memory of the bird about its previous best position, and the social component models the memory of the bird about the best position among the particles (interaction inside the swarm). At each iteration the particle move towards optimum solution, through its present velocity, personal best solution obtained by themselves and global best solution obtained by all the particles until they find the food (optimal solution). In an n-dimensional search space, position and velocity of particle j are represented by vectors $X_{ij} = (X_{i1,} X_{i2} \ldots X_{ip})$ and $V_{ij} = (V_{i1}, V_{i2} \ldots V_{ip})$ respectively. Let Xpbest vector and Xgbest be the personal and global best positions of the particles for i^{th} variable. In case of MELD problem, the generations are the independent variables of the problem. The modified velocity and position of each particle can be calculated using current velocity and distance from Xpbest and Xgbest as follows:

$$V_{ij}^{k} = V_{ij}^{k-1} + C_1 * r_1 * \left(Xpbest_{ij}^{k-1} - X_{ij}^{k-1} \right) + C_2 * r_2 * \left(Xgbest_{i}^{k-1} - X_{ij}^{k-1} \right)$$

$$i=1, 2 \ldots NG, j=1, 2 \ldots p \qquad (3.1a)$$

Position update equation is given by

$$X_{ij}^{k} = X_{ij}^{k-1} + V_{ij}^{k}$$

$$i=1, 2 \ldots NG, \ j=1, 2 \ldots p \qquad (3.1b)$$

where

k	Iteration count.
V_{ij}^{k}	Value of velocity of j^{th} particle (of i^{th} generator) at k^{th} iteration.
X_{ij}^{k}	Value of position of j^{th} particle (of i^{th} generator) at k^{th} iteration.
C_1, C_2	Acceleration coefficients.

$Xpbest_{ij}^{k}$ Value of personal best position of j^{th} particle (for i^{th} generator) in k^{th} iteration.

$Xgbest_{i}^{k}$ Value of best position of swarm (for i^{th} generator) in k^{th} iteration.

NG No. of Generating buses.

p No. of particles in the swarm.

r_1, r_2 Two separately generated random numbers from the uniformly distributed range of (0, 1).

Velocities are updated by the equation (3.1a) and the positions of each particle for each decision variable are calculated by equation (3.1b) [26].

To increase the convergence rate of the PSO algorithm the inertia weight is proposed in the velocity equation [38, 39]. By using the equation for the velocity with the inertia weight 'W', the suggested particle velocity will be changed to:

$$V_{ij}^{k} = W*V_{ij}^{k-1}+C_1 * r_1 \left(Xpbest_{ij}-X_{ij}^{k-1}\right)+C_2 * r_2(Xgbest_{i}-X_{ij}^{k-1})$$

$$i=1, 2 \dots NG, j=1, 2 \dots p \qquad (3.1c)$$

where: W is the inertia weight.

Due to this inertia weight some of the particles maintain their velocity from previous iteration to new iteration. In order to use the inertia weight in this paper, a descending linear function is used. There are other methods of using inertia weights. But, details of the same have not been investigated in the work reported in this dissertation. The best range for changing this function value for the convergence and obtaining the best possible solution is between 0.9 and 0.4 [26]. Using the inertia weight in velocity equation enables the swarm to fly in larger area of the search space (W = 0.9) and at the end of the iterations, the search space will be smaller (W = 0.4). By using the inertia weight the chance to obtain a best solution for an optimization problem will be more. In general, a linear descending function for inertia weight equation is shown in the following equation [36-39].

$$W = W_{max}-k*(W_{max}-W_{min})/iter_{max}$$

(3.1d)

Where:

W inertia weight factor

W_{max} maximum value of velocity weighting factor

W_{min} minimum value of velocity weighting factor

$iter_{max}$ maximum number of iteration
k current number of iteration

3.2 PARTICLE SWARM OPTIMIZATION (PSO) AND TRADITIONAL SEARCH METHODS

- A comparison between Conventional Optimization Techniques and evolutionary algorithms (like GA and PSO) is presented in Table 3.1 below [40, 41].
- Unlike other random search algorithms, each potential solution (called a particle) is also assigned a randomized velocity and then flown through the problem hyperspace.
- The most striking difference between PSO and the other evolutionary soft computing algorithms is that PSO chooses the path of cooperation over competition. The other algorithms commonly use some form of decimation, survival of the fittest. In contrast, the PSO population is stable and individuals are not destroyed or created. Individuals are influenced by the best performance of their neighbors. Individuals eventually converge on optimal points in the problem domain.
- The PSO traditionally does not have any genetic operators like crossover between individuals and mutation, and other individuals never substitute particles during the run. Instead, the PSO refines its search by attracting the particles to positions with good solutions.
- Particles update themselves with the internal velocity.
- They also have memory, in terms of pbest and gbest which is important to the algorithm.
- Compared with GAs, the information sharing mechanism in PSO is significantly different. In GAs, chromosomes share information with each other. So, the whole population moves like a one group toward an optimal area. In PSO, only gbest and pbest gives out the information to others. It is a one-way information sharing mechanism. The evolution only looks for the best solution.
- Compared to the GA, the advantages of PSO are that PSO is easy to implement and there are few parameters to adjust.

Table 3.1: Comparison between Conventional Optimization Procedures and Evolutionary Algorithms

Property	Evolutionary	Traditional
Search space	Population of potential solutions	Trajectory by a single point

Motivation	Natural selection and Social adaptation	Mathematical properties (gradient, Hessian)
Applicability	Domain independent, Applicable to variety of problems	Applicable to a specific problem domain
Point Transition	Probabilistic	Deterministic
Prerequisites	An objective function to be optimized	Auxiliary knowledge such as gradient vectors
Initial guess	Automatically generated by the algorithm	Provided by user
Flow of control	Mostly parallel	Mostly serial
CPU time	Large	Small
Results	Global optimum more probable	Local optimum, dependant of initial guess
Advantages	Global search, parallel, speed	Convergence proof
Drawbacks	No general formal convergence proof	Locality, computational cost

3.3 BIOLOGICAL TERMINOLOGY

Particle Swarm Optimization (PSO) is a biologically inspired computational search and optimization method developed by Eberhart and Kennedy in 1995 based on the social behaviors of birds flocking and fish schooling.

Swarm: It is an apparently disorganized population of moving particles that tend to cluster together towards a common optimum while each particle seems to be moving in a random direction.

Particle (X): It is a candidate solution, in an i-dimensional space. At time t, the j^{th} particle $X_j(t)$ can be described as $X_j\ (t)=[X_{1j}(t),X_{2j}(t),\ldots,X_{ij}(t)]$, where X_{ij} are the optimized parameters and $X_{ij}(t)$ is the position of the j^{th} particle with respect to the i^{th} dimension; i.e. the value of the i^{th} optimized parameter in the j^{th} candidate solution.

Velocity (V): It is the velocity of a moving particle, can be represented by an i-dimensional vector, where i is the number of optimized parameters. At time t, the j^{th} particle $V_j(t)$ can be described as $V_j\ (t)=[V_{ij}(t),V_{2j}(t),\ldots,V_{ij}(t)]$.

Personal best (Pbest): The personal best position associated with j^{th} particle is the best position that the particle has visited yielding the highest fitness value for that particle.

Global best (Gbest): The best position associated with j^{th} particle that any particle in the swarm has visited yielding the highest fitness value for that particle. This represents the best fitness of all the particles of a swarm at any point of time.

The optimization process uses a number of particles constituting a swarm that moves around a pre-defined search space looking for the best solution. Each particle is treated as a point in the D-dimensional space in which the particle adjusts its "flying" according to its own flying experience as well as the flying experience of other neighboring particles of the swarm. Each particle keeps track of its coordinates in the pre-defined space which are associated with the best solution (fitness) that it has achieved so far. This value is called pbest. Another best value that is tracked by the PSO is the best value obtained so far by any particle in the whole swarm. This value is called gbest. The concept consists of changing the velocity of each particle toward its pbest and the gbest position at the end of each iteration. Each particle tries to modify its current position and velocity according to the distance between its current position and pbest, and the distance between its current position and gbest [42].

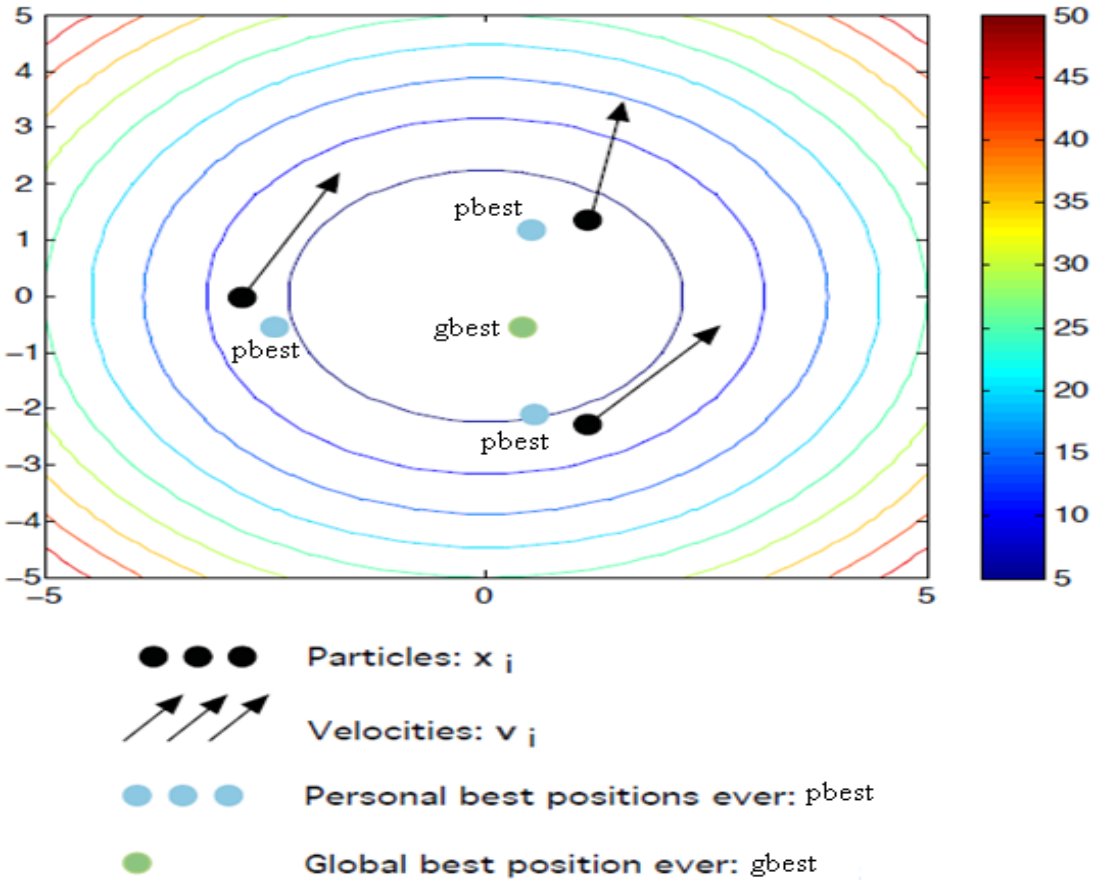


Fig 3.1: Various Biological Terminologies

3.4 COMPUTATIONAL PROCEDURE

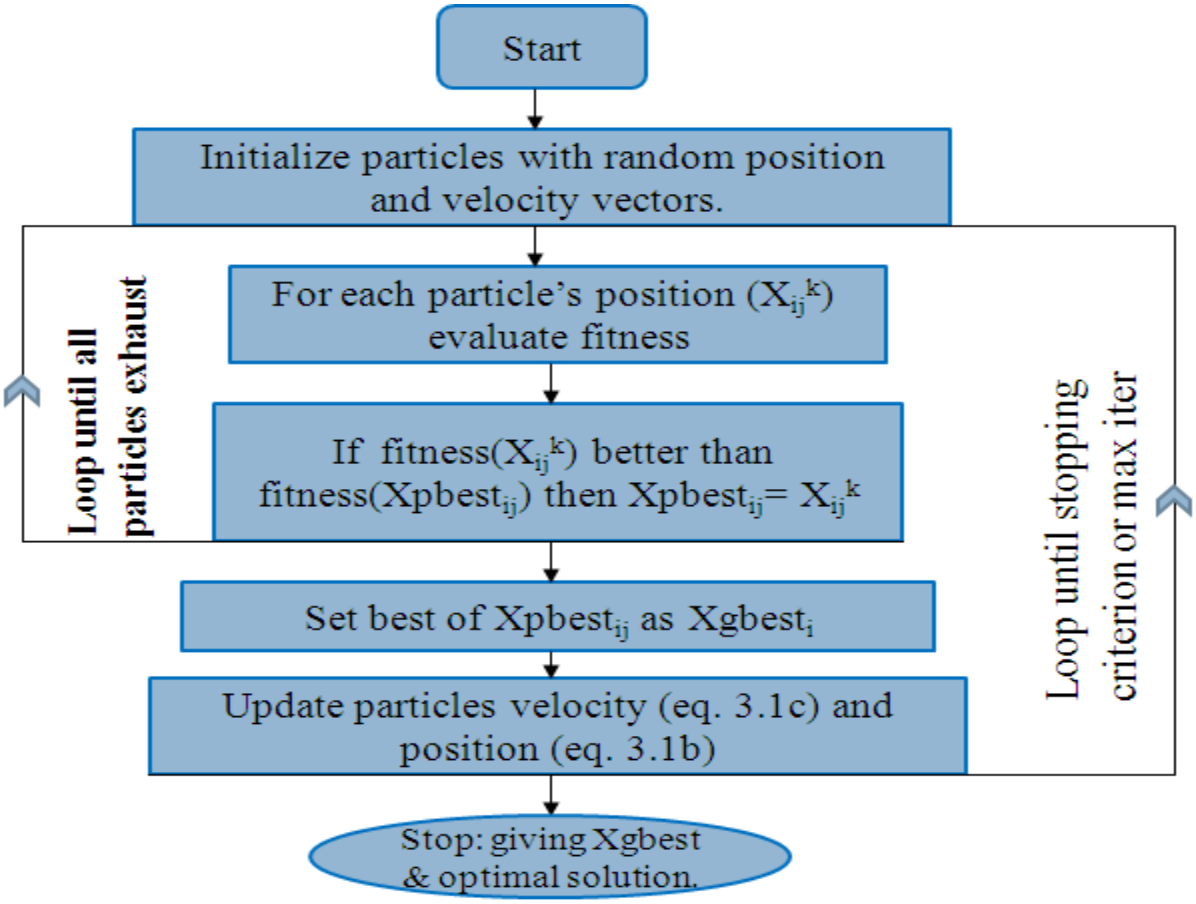


Fig 3.2: Generalized Flowchart for Particle Swarm Optimization Algorithm

The general computational procedure for the Particle Swarm Optimization is as follows:

1. Before the iteration starts, initialize the particles with random position and velocity vectors.
2. For each of the particle's position (X_{ij}^k) calculate the value of the objective function (F; we also call it fitness function).
3. If $F(X_{ij}^k)$ is less than $F(Xpbest_{ij})$, then assign the value of $Xpbest_{ij}$ as X_i^k (do it for all the particles).
4. Determine the best value of $Xpbest_{ij}$ considering its fitness value. If F(best of $Xpbest_{ij}$) is less than $F(Xgbest_i)$, than assign the value of best of $Xpbest_{ij}$ to the $Xgbest_i$.
5. Calculate the new velocity vector using equation (3.1c) and the new position vector using equation (3.1b).
6. Iterates the loop until either the stopping criteria met or the max iteration is achieved.
7. After iteration completes, give $Xgbest_i$ as the optimal solution and the fitness corresponding to it as the optimum value.

3.5 ADVANTAGES AND LIMITATIONS OF PARTICLE SWARM OPTIMIZATION

A PSO is considered as one of the most powerful methods for resolving the non-smooth global optimization problems and has many key advantages as follows:

I. PSO is a derivative-free technique just like as other heuristic optimization techniques.

II. PSO is easy in its concept and coding implementation compared to other heuristic optimization techniques.

III. PSO is less sensitive to the nature of the objective function compared to the conventional mathematical approaches and other heuristic methods.

IV. PSO has limited number of parameters including only inertia weight factor and two acceleration coefficients in comparison with other competing heuristic optimization methods. Also, the impact of parameters to the solutions is considered to be less sensitive compared to other heuristic algorithms [41].

V. PSO seems to be somewhat less dependent on a set of initial points compared to other evolutionary methods, implying that convergence algorithm is robust.

VI. PSO techniques can generate high-quality solutions within shorter calculation time and stable convergence characteristics than other stochastic methods [43].

The major drawback of PSO, like in other heuristic optimization techniques, is that it lacks somewhat a solid mathematical foundation for analysis to be overcome in the future development of relevant theories. Also, it can have some limitations for real-time ELD applications such as 5-minute dispatch considering network constraints since the PSO is also a variant of stochastic optimization techniques requiring relatively a longer computation time than mathematical approaches. However, it is believed that the PSO-based approach can be applied in the off-line real-world ELD problems such as day-ahead electricity markets. Also, the PSO-based approach is believed that it has less negative impact on the solutions than other heuristic-based approaches. However, it still has the problems of dependency on initial points and parameters, difficulty in finding their optimal design parameters, and the stochastic characteristic of the final outputs.

CHAPTER 4

SOLUTION OF MATHEMATICAL BENCHMARK FUNCTIONS USING PSO

4.1 STEPS OF PARTICLE SWARM OPTIMIZATION IN MATLAB

The basic steps for solving the optimization problem using Particle Swarm Optimization is same as explained in previous chapter. But with some modification we can use it for any type of objective function. Here PSO has been used for optimizing some mathematical functions, which are:

1. Rosenbrock function: $100(X_2^2-X_1)^2+(X_1-1)^2$
2. Beale function: $(1.5–X_1+X_1X_2)^2 + (2.25–X_1+X_1X_2^2)^2 + (2.625–X_1+X_1X_2^3)^2$
3. Sphere function: $X_1^2 + X_2^2 + X_3^2$
4. Booth's Function: $(X_1+2X_2-7)^2 + (2X_1+X_2-5)^2$

The steps for optimizing any mathematical benchmark test function:

a. Initialize all the variable matrices using the '`zeros`' command of MATLAB.
b. Set the values of random numbers '`rp`' and '`rg`' assigned to personal and global best expressions respectively.
c. Set the values of acceleration constants '`cp`' and '`cg`' assigned to personal and global best expressions respectively and set the tolerance value.
d. Generate the random values of particles for both the X_1 and X_2 variables, also generate random velocity vectors V_1 & V_2 respectively.
e. Calculate the fitness for the assumed values of the positions of the particles.
f. Using the above fitness, Xpbest vector for both the variables X_1 and X_2 and Xgbest value is deduced.
g. Using the previous iteration values of personal best, global best and velocity vector, new velocity vector is generated in the current iteration using eq. (3.1a).
h. Using the new velocity vector and the old position vector a new position vector is generated for both the variables using eq. (3.1b).
i. Calculate fitness is using the new position vectors in the current iteration.
j. Using the new fitness values, the personal and global best values are updated.
k. The difference between the previous and the current fitness is calculated and checked against the tolerance value, if within the tolerance iteration stops and global best value is the solution else iteration flow goes back to step g.

In this way optimized value of objective function and the corresponding variable values are found.

4.2 DIFFERENT PARAMETERS OF PARTICLE SWARM OPTIMIZATION

The various parameters of Particle Swarm Optimization are

1. No. of particles in the swarm, `p`.
2. Max. no. of iteration, `it`.
3. Random no. for personal & global factors `rp & rg`.
4. Acceleration constant for the personal and global factors, `cp & cg`.
5. Tolerance value, `T`.

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

1. `p` = to be fixed by user in run time.
2. `it = 1000`
3. `rp=0.4 & rg=0.5.`
4. `cp=2 & cg=2.`
5. `T = 10^(-6).`

4.3 APPLICATION OF PARTICLE SWARM OPTIMIZATION TO MATHEMATICAL BENCHMARK TEST FUNCTIONS

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

The various benchmark test function optimized are as follows:

1. Rosenbrock Function:

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

2. Beale's Function:

$$f(x_1, x_2) = (1.5 - x_1 + x_1x_2)^2 + (2.25 - x_1 + x_1x_2^2)^2 + (2.625 - x_1 + x_1x_2^3)^2$$

3. Sphere Function:

$$f(x_1, x_2, x_3) = x_1^2 + x_2^2 + x_3^2$$

4. Booth's Function:

$$f(x, y) = (x + 2y - 7)^2 + (2x + y - 5)^2$$

A general form of the equation, a plot of the objective function, constraints of the objective functions and the coordinates of global minima actual and with PSO are given herein. The effect of particle size is studied on various mathematical functions.

4.4 COMPUTATIONAL RESULTS

1. Rosenbrock's Function:

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

The minimum values for the Rosenbrock's function are as shown below:

$$\text{Minimum} = \begin{cases} n = 2 & \rightarrow \quad f(1,1) = 0, \\ n = 3 & \rightarrow \quad f(1,1,1) = 0, \\ n > 3 & \rightarrow \quad f\left(-1, \underbrace{1, \ldots, 1}_{(n-1)\text{ times}}\right) = 0. \end{cases}$$

for $-\infty \leq x_i \leq \infty, 1 \leq i \leq n$

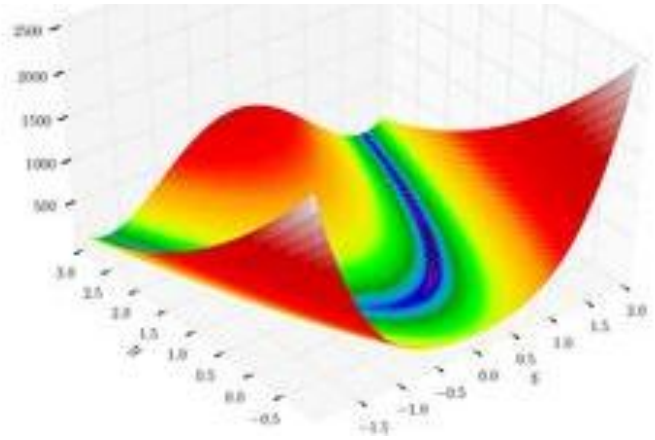


Fig 4.1: 3d Repesentation of Rosenbrock's Function

Here Rosenbrock function with n=2 has been considered i.e.

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

Detail discussion to optimize Rosenbrock function with PSO is given below:

1. Generate the random Position vectors for the two variables, i.e. X_1 {} and X_2 {}.
2. Generate the random Velocity vectors for the two variables, i.e. V_1 {} and V_2 {}. (No. of elements in the vectors is equal to no. of particles i.e. 'P')
3. So at 0^{th} iteration

Random velocities generated by MATLAB

V_{1j}^0	V_{2j}^0
0.3111	0.5949
0.9234	0.2622
0.4302	0.6028
0.1848	0.7112
0.9049	0.2217
0.9797	0.1174
0.4389	0.2967
0.1111	0.3188
0.2581	0.4242
0.4087	0.5079

Random positions generated by MATLAB

X_{1j}^0	X_{2j}^0
1.2886	0.4155
0.7572	**0.6025**
1.6232	0.9418
1.0657	0.4610
0.7015	1.6886
1.8780	0.3895
1.7519	0.4518
1.1003	0.3414
1.2450	0.4553
1.1741	0.8714

Function values at random positions

f^0

155.1104
0.1437
286.9471
45.5177
143.2694
985.0782
685.5702
75.5734
119.8689
25.7442

4. Personal best values for each particle will be their own position in the first iteration.

$$Xpbest_1 = X_1^0 \text{ and } Xpbest_2 = X_2^0$$

5. Global best value of position will be positions of that particle, corresponding to which function value is minimum.

$$Xgbest_1 = 0.7572 \text{ and } Xgbest_2 = 0.6025$$

6. Now new velocity vectors can be determined for both the variables, i.e. $\begin{bmatrix} V_{1j}^{k+1} \\ V_{2j}^{k+1} \end{bmatrix}$ for all the particles using eq. (3.1c)

7. Than new position vectors, i.e. $\begin{bmatrix} X_{1j}^{k+1} \\ X_{2j}^{k+1} \end{bmatrix}$ for all the particles using eq. (3.1b).

8. Again the objective function value is calculated using the new position vectors of the two variables for all the particles.

9. At 1st iteration

The newly calculated velocity vectors

V_{1j}^1	V_{2j}^1
-0.2516	0.7221
0.8306	0.2359
-0.4790	0.2029
-0.1422	0.7813
0.8697	-0.8867
-0.2395	0.3186
-0.5999	0.4175
-0.2431	0.5478
-0.2556	0.5287
-0.0492	0.1879

The newly calculated positions of particles

X_{1j}^1	X_{2j}^1
1.0371	1.1376
1.5878	0.8384
1.1442	1.1448
0.9235	1.2422
1.5712	0.8020
1.6385	0.7081
1.1520	0.8694
0.8572	0.8892
0.9893	**0.9840**
1.1249	1.0593

Function values for positions

f^1
0.3872
283.5112
2.7242
15.1731
278.0776
391.0947
20.9730
2.4072
0.0028
4.2594

10. Global best value of position will be changed to positions of that particle, corresponding to which function value is now minimum as compared to the last Global best value .

$$\begin{bmatrix} Xgbest_1 \\ Xgbest_2 \end{bmatrix} = \begin{bmatrix} 0.9893 \\ 0.9840 \end{bmatrix}$$

11. New personal best values are decided by comparing the last two iteration's function values.

0th iteration positions				
Random Positions		Fitness Values	Personal Best	
X_1^0	X_2^0	f^0	X_{Pbest1}	X_{Pbest2}
1.2886	0.4155	155.1104	1.2886	0.4155
0.7572	0.6025	0.1437	0.7572	0.6025
1.6232	0.9418	286.9471	1.6232	0.9418
1.0657	0.4610	45.5177	1.0657	0.4610
0.7015	1.6886	143.2694	0.7015	1.6886
1.8780	0.3895	985.0782	1.8780	0.3895
1.7519	0.4518	685.5702	1.7519	0.4518
1.1003	0.3414	75.5734	1.1003	0.3414
1.2450	0.4553	119.8689	1.2450	0.4553
1.1741	0.8714	25.7442	1.1741	0.8714

1st iteration positions				
New Positions		Fitness Values	Personal Best	
X_1^1	X_2^1	f^1	X_{Pbest1}	X_{Pbest2}
1.0371	1.1376	0.3872	**1.0371**	**1.1376**
1.5878	0.8384	283.5112	0.7572	0.6025
1.1442	1.1448	2.7242	**1.1442**	**1.1448**
0.9235	1.2422	15.1731	**0.9235**	**1.2422**
1.5712	0.8020	278.0776	0.7015	1.6886
1.6385	0.7081	391.0947	**1.6385**	**0.7081**
1.1520	0.8694	20.9730	**1.1520**	**0.8694**
0.8572	0.8892	2.4072	**0.8572**	**0.8892**
0.9893	0.9840	0.0028	**0.9893**	**0.9840**
1.1249	1.0593	4.2594	**1.1249**	**1.0593**

Bold figures are the updated values of X_{Pbest1} and X_{Pbest2} , on the basis of comparison between the fitness value.

12. Check if the difference in function values between two consecutive iterations is less than a prescribed value. If not, repeat the procedure.

With Particle Swarm Optimization, the minimum values of the Rosenbrock function are:

Table 4.1: Result for Rosenbrock's Function

No. of particles	No. of iterations	x_1	x_2	$f(x_1,x_2)$
10	112	1.0006	1.0006	$4.782*10^{-7}$
20	130	1.0067	1.0136	$4.5792*10^{-5}$
30	154	1.0000	1.0000	$4.3841*10^{-11}$
40	172	1.0000	1.0000	$1.0951*10^{-17}$

Table 4.1 shows the results of Rosenbrock function. It is observed that as the no. of particles is increased the no. of iterations increase and the optimum value are obtained with greater accuracy.

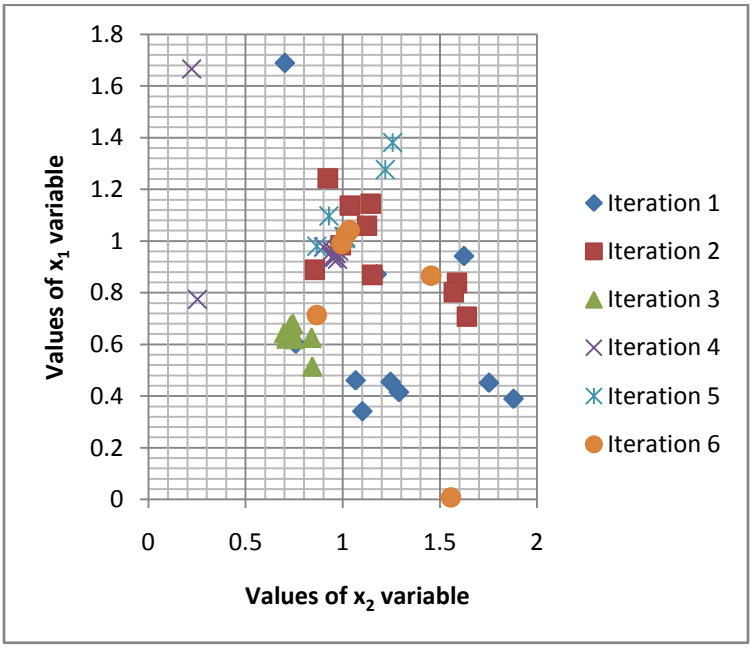


Fig 4.2(a): Mapping Of 10 Particle Position Values for 6 Iterations for Rosenbrock Function

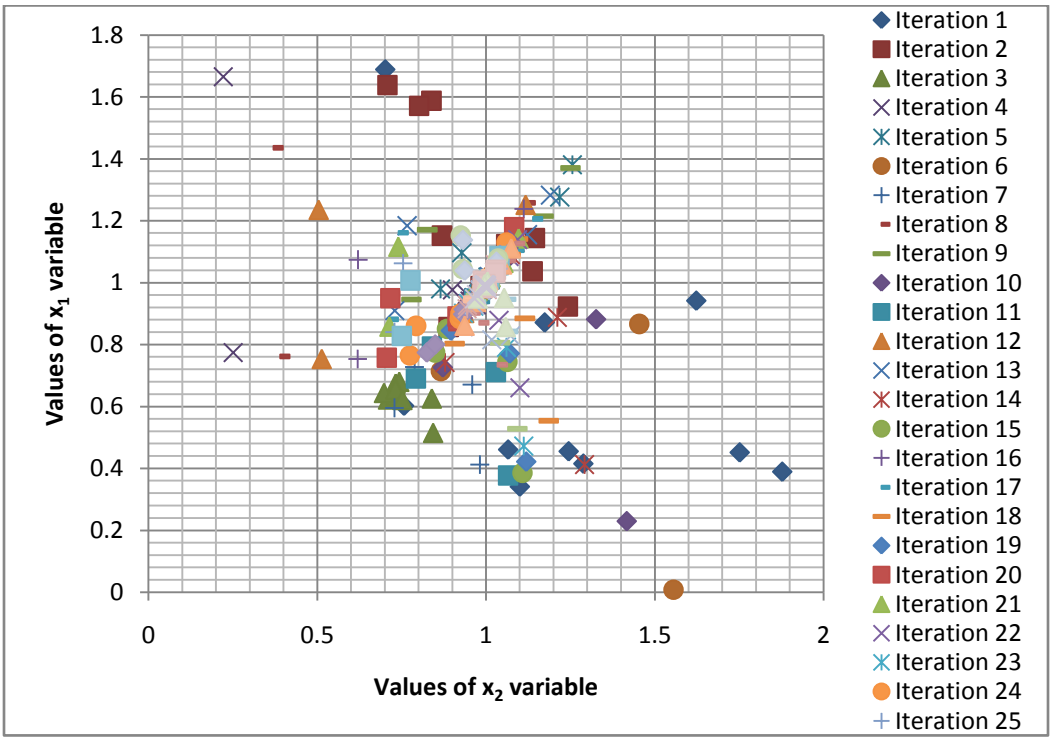


Fig 4.2(b): Mapping Of 10 Particle Position Values for 25 Iterations for Rosenbrock Function

As we can see from the fig 4.2(a) and fig 4.2(b), that as the iteration count increases the particles start converging at the (1, 1) and oscillates about it until value of all the particles converges to optimum value i.e. (1, 1) for each particle.

2. Beale function:

$$f(x,y) = (1.5 - x + xy)^2 + \left(2.25 - x + xy^2\right)^2 + \left(2.625 - x + xy^3\right)^2.$$

The minimum values for the Beale function are as follows:

Minimum = $f(3, 0.5) = 0$ for $-4.5 \leq x, y \leq 4.5$

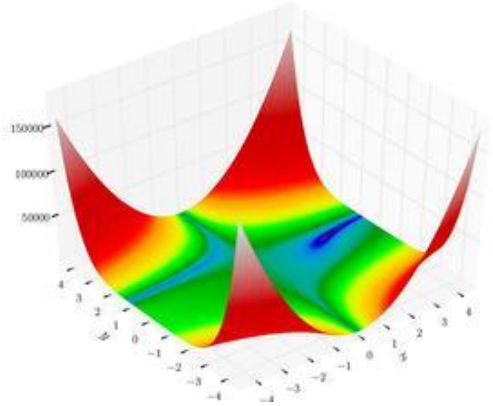


Fig 4.3: Beale's Function In 3d

Table 4.2 shows the results of Beale function.

Table 4.2: Result for Beale's Function

No. of Particles	No. of iterations	*x*	*y*	*f*(*x*, *y*)
10	308	3.1684	0.5642	$2.2979*10^{-2}$
20	310	3.0002	0.4948	$6.2522*10^{-4}$
30	316	3.0384	0.5127	$4.9643*10^{-4}$
40	320	3.0161	0.5048	$5.7103*10^{-5}$

It is observed that as the no. of particles is increased the no. of iterations increase and the optimum value are obtained with greater accuracy.

3. Sphere function:

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

The minimum values for the Sphere function are as shown below:

Minimum = $f(x_1, \ldots, x_n) = f(0, \ldots, 0) = 0$

for $-\infty \leq x_i \leq \infty, 1 \leq i \leq n$

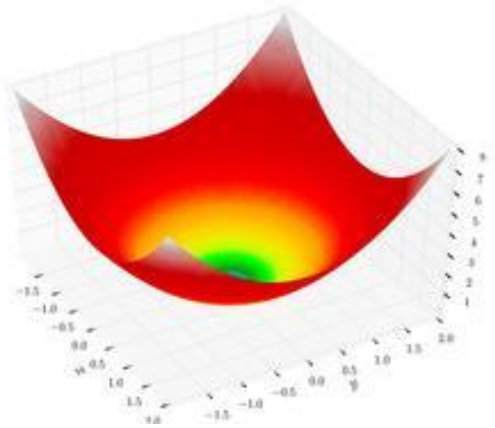


Fig 4.4: Sphere Function In 3d

Here I have considered n=3, so the function I have taken is:

$$f(x_1,x_2,x_3) = x_1^2 + x_2^2 + x_3^2$$

Table 4.3 shows minimum values of the sphere function using PSO technique are:

Table 4.3: Result for Sphere Function

No. of Particles	No. of iterations	x_1	x_2	x_3	$f(x_1,x_2,x_3)$
10	265	-0.0505	0.1034	0.0127	0.0197
20	273	-0.0123	-0.0242	-0.0252	0.0013
30	275	0.0063	0.0110	0.0141	0.0003
40	273	0.0204	-0.0028	0.0018	0.0004

It is observed that as the no. of particles is increased the no. of iterations increase and the optimum value are obtained with greater accuracy.

4. Booth's Function:

$$f(x, y) = (x + 2y - 7)^2 + (2x + y - 5)^2$$

The minimum values for the Sphere function are as shown below:

Minimum: $f(1,3) = 0$

for $-10 \leq x, y \leq 10$

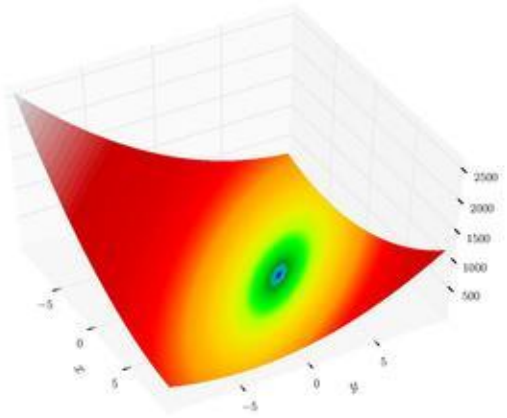


Fig 4.5: Booth's Function In 3d

Table 4.4: Result for Booth's Function

No. of Particles	No. of iterations	*x*	*y*	f(*x*, *y*)
10	278	1.0117	3.0051	0.0013
20	296	0.9959	2.9995	0.0001
30	294	1.0023	3.0016	0.00007
40	296	0.9916	3.0005	0.0003

In general it is observed that as the no. of particles is increased the no. of iterations increase and the optimum value are obtained with greater accuracy. But in this case it is observed that with the 30 particles no. of iterations reduced and the accuracy increased as compared when no. of particles was taken as 40.

CHAPTER5:

Multi-Objective Approach to Economic Load Dispatch

5.1 PROBLEM FORMULATION IN 2-D SPACE WITH EQUALITY CONSTRAINTS

The two aspects of Multi-objective Economic Load Dispatch (MELD) problem considered here are:

1) To minimise the cost of generation.
2) To minimise the system transmission loss.

The objective function to minimise the cost of generation is given as

$$F_C = \sum_{i=1}^{NG} F[C_i(P_{gi})] \quad (5.1a)$$

where:

$$C_i(P_{gi}) = \sum_{i=1}^{NG} (a_i * P_{gi}^2 + b_i * P_{gi} + c_i) \quad (5.1b)$$

where:

P_{gi}	is the active power generation at the i^{th} generator.
C_i	is the cost of generation for i^{th} generator.
NG	is the total number of generators in the system.
a_i, b_i, c_i	are fuel cost coefficients of i^{th} generator.

The objective function to minimise the system transmission loss is given as

$$F_L = \sum_{i=1}^{NG} \sum_{i=1}^{NG} (P_{gm} B_{mn} P_{gn}) + \sum_{i=1}^{NG} (P_{gm} B_{om}) + B_{oo} \quad (5.1c)$$

Where:

P_{gm}, P_{gn}	is the active power at the m^{th} and n^{th} generator.
NG	is the total number of generators in the system.
B_{mn}, B_{om}, B_{oo}	are loss coefficients.

Table 5.1: Values of Cost Coefficients

	Coefficients	G1	G2	G3
5- Bus	a	0.005	0.005	------
	b	3.51	3.89	------
	c	44.4	40.6	------

14- Bus	a	0.005	0.005	0.005
	b	3.51	3.89	2.45
	c	44.4	40.6	105
30- Bus	a	0.005	0.005	0.005
	b	3.51	3.89	2.45
	c	44.4	40.6	105

Table 5.2: Values of Loss Coefficients

	5-Bus	14-Bus	30-Bus
B11	0.000349	0.000349	0.000307
B12	0.000086	0.000068	0.000129
B13	---	-0.000039	-0.000002
B22	0.000371	0.000157	0.000152
B23	---	0.000015	-0.000011
B33	---	0.000275	0.000190
B01	---	0.000044	---
B02	---	0.000024	---
B03	---	0.000000	---
B00	---	0.000254	---

The method used in this dissertation for considering the transmission losses, has been developed by Kron and adopted by Kirchmayer, which is the loss coefficient method [44, 45].

The multiobjective function ($F(F_C,F_L)$) to be minimised using PGP technique is formulated as the weighted sum of the cost of generation (F_C) and transmission loss (F_L).

Mathematically, the problem is to minimise

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

where:

$$F = W_C*F_C + W_L*F_L \quad (5.1d)$$

subject to the constraints:

Equality constraint

$$\sum_{i=1}^{NG} P_{gi} = P_D + F_L \quad (5.1e)$$

Inequality constraint

$$P_{gimin} \leq P_{gi} \leq P_{gimax} \qquad i = 1, 2... NG \quad (5.1f)$$

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

where:

F:	multi-objective function to be optimized.
F_C:	cost of generation.
F_L:	system transmission losses.
W_C:	priority given to the cost of generation.
W_L:	priority given to transmission loss.
P_{g1}, P_{g2}... P_{gNG}:	are the generations at the generators.
P_D:	is the total load demand.
NG:	is the no. of generators.
P_{gi}:	generation from i^{th} generator.
P_{gimin}:	minimum generation possible from i^{th} generator.
P_{gimax}:	maximum generation possible from i^{th} generator.

5.2 COMPUTATIONAL PROCEDURE FOR APPLICATION OF PSO IN MELD

Although the above approach to the Multiobjective Economic Load Dispatch using Priority goal programming (PGP) technique seems easy to handle, there are two major tasks involved:

I. Selection of the best solution (i.e. minimum value of F_C and F_L).

II. Decision about the priorities assigned to F_C and F_L i.e. W_C and W_L.

The PGP formulation has been modified by inclusion of a parameter K to consider the equality constraint of the problem. The multi-objective function becomes as follows:

$$F = W_C*F_C + W_L*F_L + K (P_D + F_L - P_G) \quad (5.2a)$$

Where:

Parameter K is fixed at 1000 for all three IEEE 5, 14 and 30 bus systems. Different values of K were considered and it was observed that MELD problem converged when it was fixed to 1000 for all the systems, i.e. IEEE 5, 14 and 30- bus systems.

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

```
for i=1: NG
    for m=1: p
        if Pgi < Pgimin
           Pgi = Pgimin;
        end
        if Pgi > Pgimax
           Pgi = Pgimax
        end
    end
end
```

The optimum solution is obtained when the

I. Change in the value of Multi-objective Economic Load Dispatch function during successive iterations is less than the limit specified which is $\varepsilon=1*10^{-6}$ and

II. The equality constraint is satisfied such that the absolute value of difference between generation, demand and losses is less than $\varepsilon=1*10^{-6}$.

Population size of the swarm and the maximum number of iteration can be selected by the user of the program in run time. We have chosen ten (10) particles in the swarm and one thousand (1000) as the maximum number of iteration.

The initial position and velocity of particles have been generated randomly by using MATLAB. The limits are imposed on position of particles to increase the convergence. Here positions i.e. the generations are decision variables.

The maximum and minimum limits on the velocity have been assigned as follows $V_{min}= -P_{gimin}/2$ and $V_{max}= P_{gimax}/2$ respectively. The velocities are fixed to the values of corresponding limits if violated during the iterations [25].

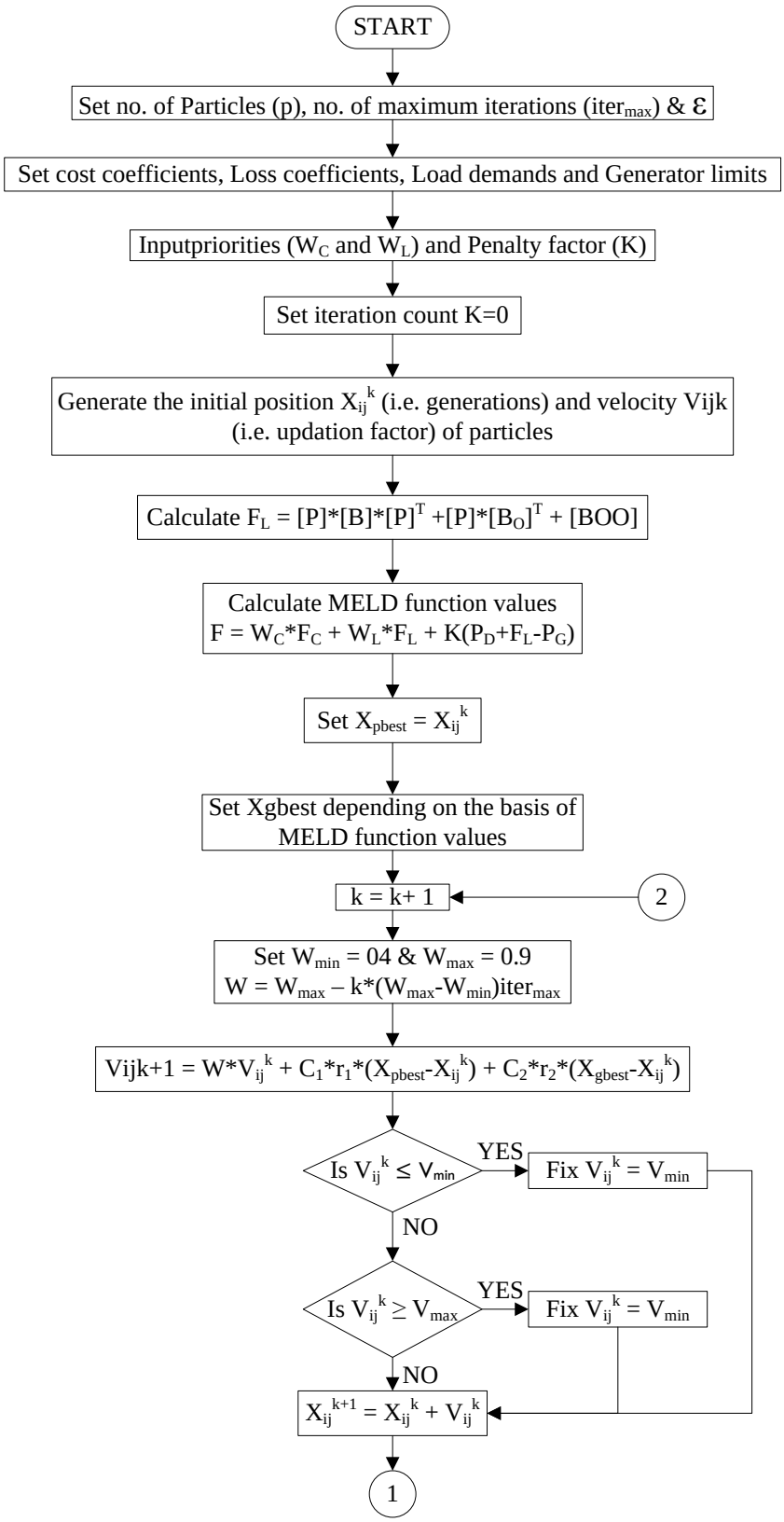
Initial values of personal best and global best have been taken as the initial value randomly generated by MATLAB.

The values of acceleration factors C_1 and C_2 are same, which implies the same priority given to Xpbest and Xgbest in the evolution processes. Both C_1 and C_2 are kept 2 [25]. r_1 and r_2 are the

random variable associated with the Xpbest and Xgbest respectively and they can have values between [0, 1]. Here r_1 and r_2 are kept 0.4 and 0.5 respectively [25]. Flowchart of solution of Multi-objective Economic Load Dispatch (MELD) problem using PSO is shown in Fig. 5.1.

The sequence for the solution of Multi-objective Economic Load Dispatch (MELD) problem using Particle Swarm Optimization technique is explained as follows:

1. Set the no. of particles 'p' in swarm and set the no. of maximum iterations $iter_{max}$.
2. Set the cost coefficients, loss coefficients, and load demand and generator limits of all the generators.
3. Input the priorities W_C and W_L assigned to cost and loss functions respectively.
4. Set iteration count k = 0.
5. Generate X_{ij}^k and V_{ij}^k, the initial random positions (i.e. generations) and velocity (i.e. updation factor) respectively.
6. Calculate the losses for each particle, using the eq. (5.1c).
7. Calculate the value of MELD function using eq. (5.2a).
8. At $zero^{th}$ iteration the personal and global best positions (i.e. generations) are same as the initial random positions i.e. generations).
9. Increase the iteration count k by 1 using k=k+1.
10. Calculate the particle's inertia weight factor W using W_{min}=0.4 and W_{max}=0.9 and the $iter_{max}$ in eq. (3.1d).
11. Calculate the velocity (i.e. positions updation factor) of each particle using eq. (3.1c).
12. Check if velocity is within the limits. If not fix the velocity to the limit violated.
13. Calculate the new positions (i.e. generations) of the particles by evaluating eq. (3.1b).
14. Check if generations (i.e. positions) of each particle are within the generator limits, if not fix the generation to the limit violated.
15. Calculate MELD function for the new positions (i.e. generations) generated.
16. Update Xpbest and Xgbest values by comparing MELD function values.
17. Check if both the stopping criteria (i.e. one is difference in the value of objective function for two consecutive iteration and another is the value of equality criteria) are satisfied (i.e. both should be less than 10^{-6}), if not then go to step 9.
18. Output the values of cost of generation and system transmission losses.


START
Set no. of Particles (p), no. of maximum iterations (iter$_{max}$) & ε
Set cost coefficients, Loss coefficients, Load demands and Generator limits
Inputpriorities (W_C and W_L) and Penalty factor (K)
Set iteration count K=0
Generate the initial position X_{ij}^k (i.e. generations) and velocity Vijk (i.e. updation factor) of particles
Calculate $F_L = [P]*[B]*[P]^T + [P]*[B_O]^T + [BOO]$
Calculate MELD function values $F = W_C*F_C + W_L*F_L + K(P_D+F_L-P_G)$
Set $X_{pbest} = X_{ij}^k$
Set Xgbest depending on the basis of MELD function values
k = k+ 1
2
Set W_{min} = 04 & W_{max} = 0.9
$W = W_{max} – k*(W_{max}-W_{min})iter_{max}$
$Vijk+1 = W*V_{ij}^k + C_1*r_1*(X_{pbest}-X_{ij}^k) + C_2*r_2*(X_{gbest}-X_{ij}^k)$
Is $V_{ij}^k \leq V_{min}$
YES
Fix $V_{ij}^k = V_{min}$
NO
Is $V_{ij}^k \geq V_{max}$
YES
Fix $V_{ij}^k = V_{min}$
NO
$X_{ij}^{k+1} = X_{ij}^k + V_{ij}^k$
1

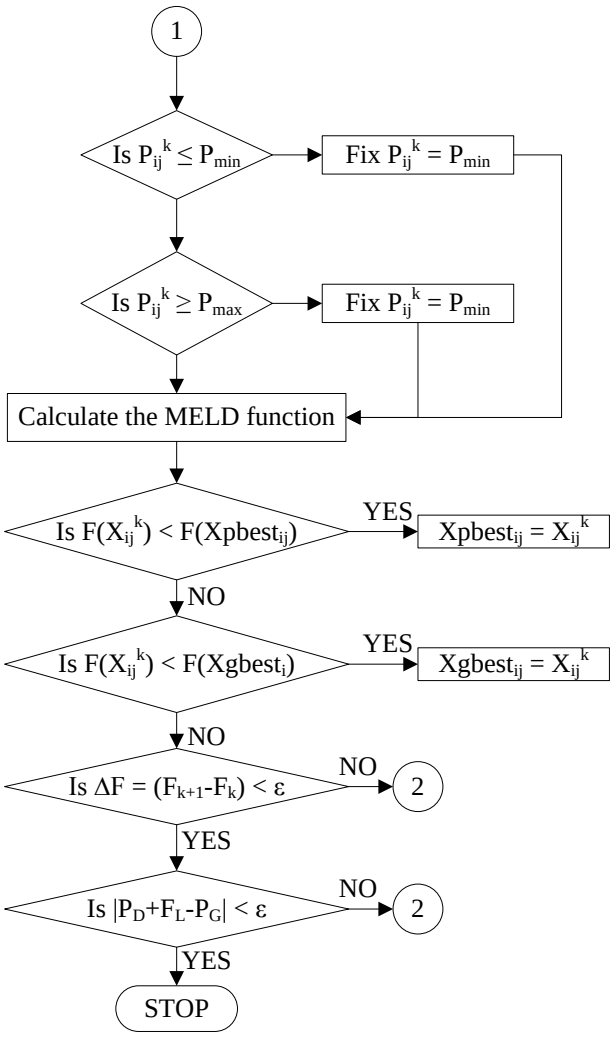


Fig. 5.1 FLOW CHART OF IMPLEMENTATION OF PSO ON MELD

5.3 COMPUTATIONAL RESULTS IN 2-D SPACE

Multi-objective Economic Load Dispatch (MELD) problem has been formulated as weighted sum of various objectives using PGP technique. The priority of cost of generation (i.e. W_C) has been fixed to one and priority of system transmission losses (i.e. W_L) has been varied. The Multi-objective Economic Load Dispatch (MELD) problem so formed has been solved by Particle Swarm Optimization technique. These solutions are the non-inferior sets of IEEE 5, 14 and 30 bus systems which are shown in Tables 5.3, 5.4 and 5.5 respectively.

Table 5.3: Non-inferior Set Of IEEE 5 Bus System

S. No.	W_C	W_L	F_C($/h)	F_L(MW)
1	1.0	0.0	761.13	5.2273
2	1.0	0.5	761.14	5.2259
3	1.0	1.0	761.16	5.2126
4	1.0	10.0	761.30	5.1698
5	1.0	19.0	761.67	5.1379
6	1.0	20.0	761.77	5.1351
7	1.0	30.0	762.68	5.1208
8	1.0	50.0	763.64	5.1150
9	1.0	100.0	763.70	5.1149
For W_C = 0.0 and W_L = 1.0, the cost of generation assumes a value which is maximum (F_{Cmax}) of 763.75$/h and transmission loss is minimum (F_{Lmin}) with a value of 5.1149MW.				

Table 5.4: Non-inferior Set Of IEEE14 Bus Sytem

S. No.	W_C	W_L	F_C($/h)	F_L(MW)
1	1.0	0.0	1162.32	7.4125
2	1.0	0.5	1164.02	7.3246
3	1.0	1.0	1176.83	6.9062
4	1.0	9.0	1183.28	6.7542
5	1.0	10.0	1184.47	6.7648
6	1.0	20.0	1199.09	6.5244
7	1.0	30.0	1204.96	6.4786
8	1.0	50.0	1210.44	6.4521
9	1.0	100.0	1222.62	6.4321
For W_C = 0.0 and W_L = 1.0, the cost of generation assumes a value which is maximum (F_{Cmax}) of 1223.11$/h and transmission loss is minimum (F_{Lmin}) with a value of 6.4135MW.				

Table 5.5: Non-inferior Set Of IEEE 30 Bus System

S. No.	W_C	W_L	F_C($/h)	F_L(MW)
1	1.0	0.0	1290.67	8.4740
2	1.0	0.5	1294.46	8. 3249
3	1.0	1.0	1302.35	8.0381
4	1.0	10.0	1311.35	7.7864
5	1.0	13.0	1318.05	7.5652
6	1.0	20.0	1328.04	7.3357
7	1.0	30.0	1346.92	7.0637
8	1.0	50.0	1354.27	6.9815
9	1.0	100.0	1358.01	6.9448
For W_C = 0.0 and W_L = 1.0, the cost of generation assumes a value which is maximum (F_{Cmax}) of 1358.02$/h and transmission loss is minimum (F_{Lmin}) with a value of 6.9447MW.				

Tables 5.3, 5.4 and 5.5 represent the variation of F_C and F_L with W_C = 1.0 and different values of W_L in the cost function for IEEE 5 bus, 14 bus and 30 bus systems [46] respectively. It is observed that, when W_C = 1.0 and W_L = 0.0, we achieve the minimum cost of generation and maximum transmission loss. Keeping W_C equal to 1.0 constant and as W_L is increased the cost of generation is increased and the transmission loss is reduced. When W_L is very large compared to W_C, essentially we approach minimum loss and maximum cost. This means that, if we give full priority to the transmission loss and no priority to cost in the objective function, i.e. if W_L = 1.00 and W_C = 0.0, then we attain the minimum system transmission loss (F_L) and maximum cost of generation (F_C).

5.4 ANALYSIS AND ACHIVEMENT OF TARGET POINT

Target point is achieved by using the percentage savings in the F_C (i.e. cost of generation) and the F_L (i.e. system transmission losses) i.e. PSC and PSL respectively. Target point is one where both these percentage savings become equal As both the objectives are conflicting in nature a trade off may be settled between the two by assuming equal percentage savings, which also means equal satisfaction for both the objectives.

For in depth analysis and clarity of the studies the following definitions are useful:

Saving in F_C (SFC) = value of F_C when only F_L is minimised – value of F_C when both F_C and F_L are minimised simultaneously

$= F_{Cmax} - F_C$

Saving in F_L (SFL) = value of F_L when only F_C is minimised – value of F_L when both F_L and F_C are minimised simultaneously

$= F_{Lmax} - F_L$

These savings are expressed as percentages of the maximum possible savings. It is known that minimum cost of generation is achieved when the objective function is minimised with $W_C = 1.0$ and $W_L = 0.0$. Similarly, the minimum transmission loss is achieved when the objective function is minimised considering $W_L = 1.0$ and $W_C = 0.0$. Thus, maximum possible saving in the cost of generation is defined as:

MPSC = value of F_C when $W_L = 1.0$ and $W_C = 0.0$ - value of F_C when $W_L = 0.0$ and $W_C = 1.0$

$= F_{Cmax} - F_{Cmin}$

Similarly, the maximum possible saving in transmission loss is defined as:

MPSL = value of F_L with $W_C = 1.0$ and $W_L = 0.0$ - value of F_L with $W_C = 0.0$ and $W_L = 1.0$

$= F_{Lmax} - F_{Lmin}$

From the data of Tables 5.3, 5.4 and 5.5, savings in F_C and F_L are computed as percentages of the respective maximum possible savings and shown in Tables 5.6, 5.7 and 5.8 for IEEE 5 bus, 14 bus and 30 bus systems, respectively.

$$\text{PSC} = \frac{\text{SFC}}{\text{MPSC}} * 100$$

$$\text{PSL} = \frac{\text{SFL}}{\text{MPSL}} * 100$$

Analysis of these Tables shows that, when $W_C = 1.0$ and $W_L = 0.0$, the percentage saving in F_C is 100% and in F_L is 0.00%. Keeping $W_C = 1.0$ as W_L is increased; percentage saving in F_C is reduced, whereas percentage saving in F_L is increased. Finally, when full weighting is given to

the transmission loss and no weighting to the cost of generation, i.e. when W_L = 1.0 and W_C = 0.0, 100% saving in F_L and 0.00% saving in F_C (see footnotes under Tables 5.6, 5.7 and 5.8) are achieved.

Table 5.6: Percentage Savings for IEEE 5 Bus System

S. No.	W_C	W_L	Savings		% Saving	
			SFC($/h)	SFL(MW)	PSC	PSL
1	1.0	0.0	2.62	0.0000	100.0	0.00
2	1.0	0.5	2.61	0.0014	99.61	1.24
3	1.0	1.0	2.59	0.0147	98.94	13.08
4	1.0	10.0	2.45	0.0575	93.51	51.15
5	**1.0**	**19.0**	**2.08**	**0.0894**	**79.38**	**79.55**
6	1.0	20.0	1.98	0.0922	75.57	82.03
7	1.0	30.0	1.07	0.1065	40.98	94.75
8	1.0	50.0	0.11	0.1123	04.20	99.91
9	1.0	100.0	0.05	0.1124	01.91	100.0
For W_C = 0.0 and W_L = 1.0, the percentage saving in F_C is 0.00% and percentage saving in F_L is 100%						

Table 5.7: Percentage Savings for IEEE 14 Bus System

S. No.	W_C	W_L	Savings		% Saving	
			SFC($/h)	SFL(MW)	PSC	PSL
1	1.0	0.0	60.79	0.0000	100.0	0.00
2	1.0	0.5	59.09	0.0879	97.20	8.80
3	1.0	1.0	46.28	0.5063	76.13	50.68
4	**1.0**	**9.0**	**39.83**	**0.6543**	**65.52**	**65.50**
5	1.0	10.0	38.64	0.6777	63.56	67.84
6	1.0	20.0	24.02	0.8881	39.51	88.89
7	1.0	30.0	18.15	0.9339	29.85	93.48
8	1.0	50.0	12.67	0.9604	20.84	96.13
9	1.0	100.0	00.49	0.9804	00.81	98.14
For W_C = 0.0 and W_L = 1.0, the percentage saving in F_C is 0.00% and percentage saving in F_L is 100%						

Table 5.8: Percentage Savings for IEEE 30 Bus System

S. No.	W_C	W_L	Savings		% Saving	
			SFC($/h)	SFL(MW)	PSC	PSL
1	1.0	0.0	67.35	0.0000	100.0	0.00
2	1.0	0.5	63.55	0.1491	94.36	9.75
3	1.0	1.0	55.66	0.4359	82.65	28.50
4	1.0	10.0	46.67	0.6876	69.29	44.96

5	**1.0**	**13.0**	**39.97**	**0.9088**	**59.35**	**59.43**
6	1.0	20.0	29.98	1.1383	44.51	74.43
7	1.0	30.0	11.10	1.4103	16.48	92.22
8	1.0	50.0	03.75	1.4925	5.57	97.59
9	1.0	100.0	00.01	1.5292	0.00	99.99
For W_C = 0.0 and W_L = 1.0, the percentage saving in F_C is 0.00% and percentage saving in F_L is 100%						

The results in Tables 5.6, 5.7 and 5.8 are represented by graphs in Fig. 5.2. All the curves for the three different systems initiate from the point of 100.00% saving in F_C and terminate at the point corresponding to 100% saving in F_L. For any system, Fig. 5.2 reveals that 100% saving in F_C is achieved when W_C = 1.0 and W_L = 0.0 and, as W_L is increased progressively, keeping W_C = 1.0, the percentage saving in F_C is reduced and the percentage saving in F_L is increased. Finally, when W_L becomes very large compared to W_C the above curve approaches 100% saving in F_L which is incidentally the point corresponding to W_L = 1.0 and W_C = 0.0. This represents the tradeoff between the two objectives i.e. the cost of generation, F_C and the system transmission losses F_L.

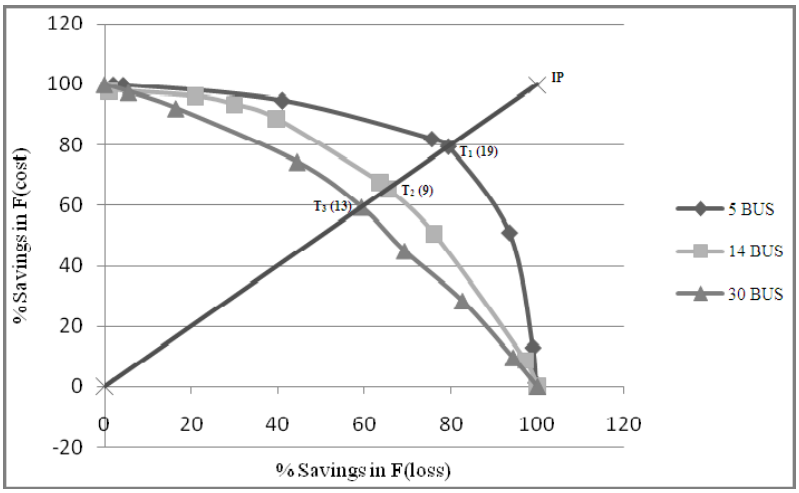


Fig. 5.2: Percentage Savings in F_C and F_L in IEEE 5bus, 14bus, and 30bus System

Table 5.9 summarizes the Target Point (TP) and the priorities (weights) at which they are achieved for IEEE 5, 14 and 30 bus systems.

Table 5.9: Target Points for the Various Test Systems

IEEE System	W_C	W_L	SFC	SFL
5 Bus System	1.0	19	761.67	5.1379
14 Bus System	1.0	9	1183.28	6.7582
30 Bus System	1.0	13	1318.05	7.5652

It may be observed that, ideally, one would like to operate at Ideal Point (IP) corresponding to minimum cost of generation and minimum loss, i.e. 100% saving in F_C and 100% saving in F_L. However, this is not feasible. When one optimizes an objective function consisting of multiple objectives the attempt should be to achieve an operating point as close as possible to the ideal point IP. Such operating points closest to IP, incidentally, lie on the intersection of the line joining origin and IP and percentage saving in F_L against percentage saving in F_C curves. In such a situation, depending on the requirement, the power system planner has to decide the required priorities to the multiple objectives to achieve a level of satisfaction. In case equal importance is apportioned to the two objectives in the problem under consideration, the optimum point of operation for the three systems will lie on the intersection of straight line joining the origin and IP and the corresponding percentage saving in F_C against percentage saving in F_L curves. These operating points are indicated by T_1 (W_C = 1.0, W_L = 19), T_2 (W_C = 1.0, W_L = 9) and T_3 (W_C = 1.0, W_L= 13) for 5 bus, 14 bus and 30 bus systems, respectively (Fig. 5.2). Interestingly, these points are closest to the ideal point IP, which provides equal satisfaction in percentage saving in both F_C and F_L. Deviation from these points along the curves, will lie at distances greater than the minimum distance of T1, T2 and T3 from IP and will correspond to unequal satisfaction levels in the achievement of the two objectives. The power system planner should appropriately choose the operating points on these curves meeting his own requirement, based on system considerations.

CHAPTER6:

Conclusions and future directions

6.1 CONCLUSIONS

The following are the significant contributions of this paper.

a. Priority goal programming technique has been used to formulate the MELD problem considering two objectives the cost of generation and the system transmission losses for IEEE 5, 14 and 30 bus systems. The technique has no limitation in handling more than two objectives.
b. Equality constraint of MELD problem has been considered using penalty parameter K whereas inequality constraints of the problem have been handled in the PSO programming itself.
c. An intelligent optimization technique – Particle Swarm Optimization has been used successfully to generate the non-inferior set for IEEE 5, 14, and 30 bus systems for MELD considering cost of generations and transmission losses for the first time.
d. Tradeoff analysis has been carried out by determining the percentage saving in cost of generation (PSC) and percentage saving in system transmission losses (PSL).
e. Target Point has been defined as the one for which percentage savings for both the objectives are same.

6.2 FUTURE DIRECTIONS

Enough future scope is there to work in the area of PSO, a few aspects of possible future research work are given below:

a. Here one can work on the selection criteria of various constants in the PSO algorithm like acceleration constants (i.e. C_1 and C_2) and random no. (i.e. r_1 and r_2).
b. Computation of W- inertia factor.
c. Consideration of other objectives of Power Systems like environmental pollution, multiple valve point effects, security etc.

APPENDIX I

1) IEEE 5 BUS SYSTEM:

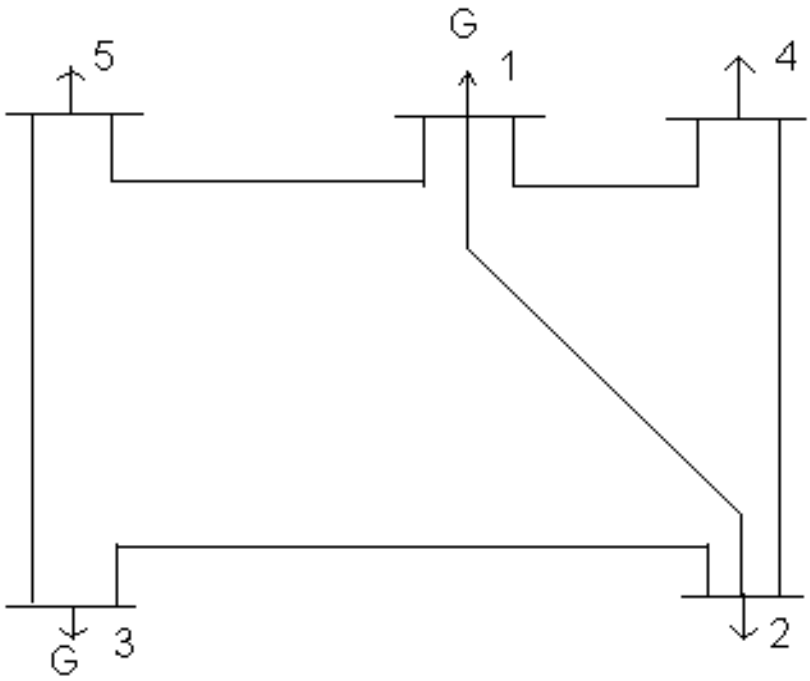


Fig. (I-A): BUS-CODE DIAGRAM 5 BUS SYSTEM

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 impedances are based on MVA as 100.

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

	GENERATION	GENERATION	LOAD	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	MINIMUM MVAR CAPABILITY	MAXIMUM MVAR CAPABILITY	MINIMUM MW CAPABILITY	MAXIMUM MW CAPABILITY
1	1.02	0.0	60	30	120
3	1.04	0.0	60	30	120

The nodal load voltage inequality constraints are $0.9 <= V_i <= 1.05$

Cost Characteristics

The cost characteristics of the IEEE 5 Bus System are as follows:

$C_1 = 50\ P_1\text{^}2 + 351\ P_1 + 44.4$ \$/hr

$C_3 = 50\ P_3\text{^}2 + 389\ P_3 + 40.6$ \$/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)
mwlimits=[30 120;30 120];
lfybus
lfnewton
busout
bloss
```

B- Coefficients Calculated are as:

B11 = 0.00035336
B12 = 0.0000103196
B21 = 0.0000103196
B22 = 0.000368992

2) IEEE 14 BUS SYSTEM:

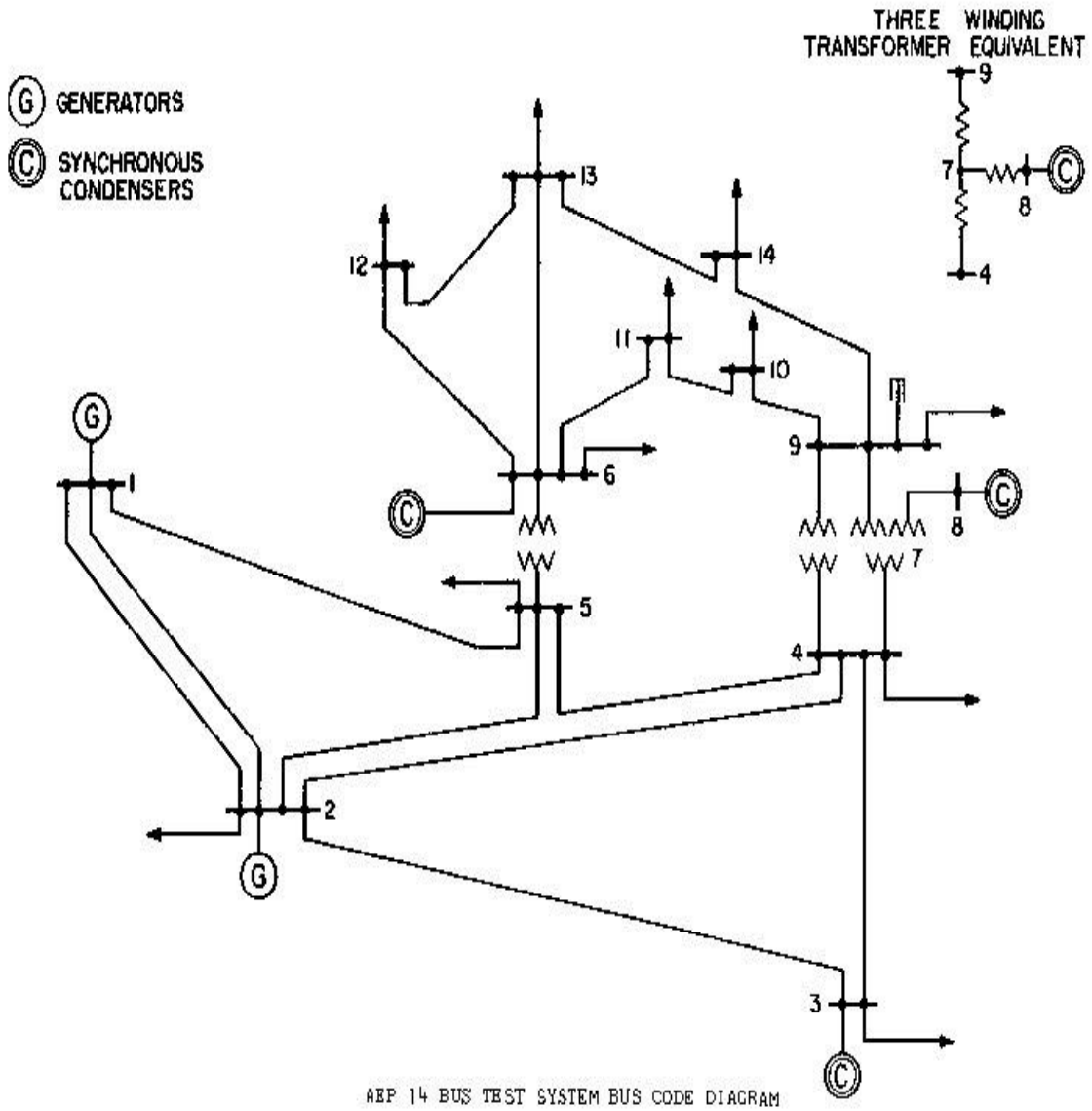


Fig. (I-B): BUS-CODE DIAGRAM 14 BUS SYSTEM

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

Line Designation	Resistance p.u.*	Reactance p.u.*	Line Charging	Tap Setting
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	0.0187	1
2-5	0.05695	0.17388	0.0170	1
3-4	0.06701	0.17103	0.0173	1
4-5	0.01335	0.04211	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. Line charging one-half of the total charging of line.

TABLE (I-E): BUS DATA OR OPERATING CONDITIONS (14 BUS SYSTEM)

Bus No.	Voltage		Generation	Generation	Load	Load
	Magnitude (p.u.)	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 p.u.	Minimum MVAR capability	Maximum MVAR capability
2	1.045	-40	50
3	1.010	0	40
6	1.070	-6	24
8	1.090	-6	24

Cost Characteristics:

$C_1 = 50\ P_1\text{^}2 + 245\ P_1 + 105$ \$/hr

$C_2 = 50\ P_2\text{^}2 + 351\ P_2 + 44.4$ \$/hr

$C_6 = 50\ P_6\text{^}2 + 389\ P_6 + 40.6$ \$/hr

Here the total load demand of the system is 259 MW. Maximum and minimum active power constraint on the generator bus for the given system is 150 MW and 50 MW respectively. Voltage magnitude constraint for generator 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 0.0187 1;2 5 0.0595 0.17388 0.0170 1;3 4 0.06701 0.17103
0.0173 1;4 5 0.01335 0.04211 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];
disp(busdata)
disp(linedata)
mwlimits=[50 150;50 150;50 150];
lfybus
lfnewton
busout
bloss
```

B-Coefficients Calculated are 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

3) IEEE 30 BUS SYSTEM:

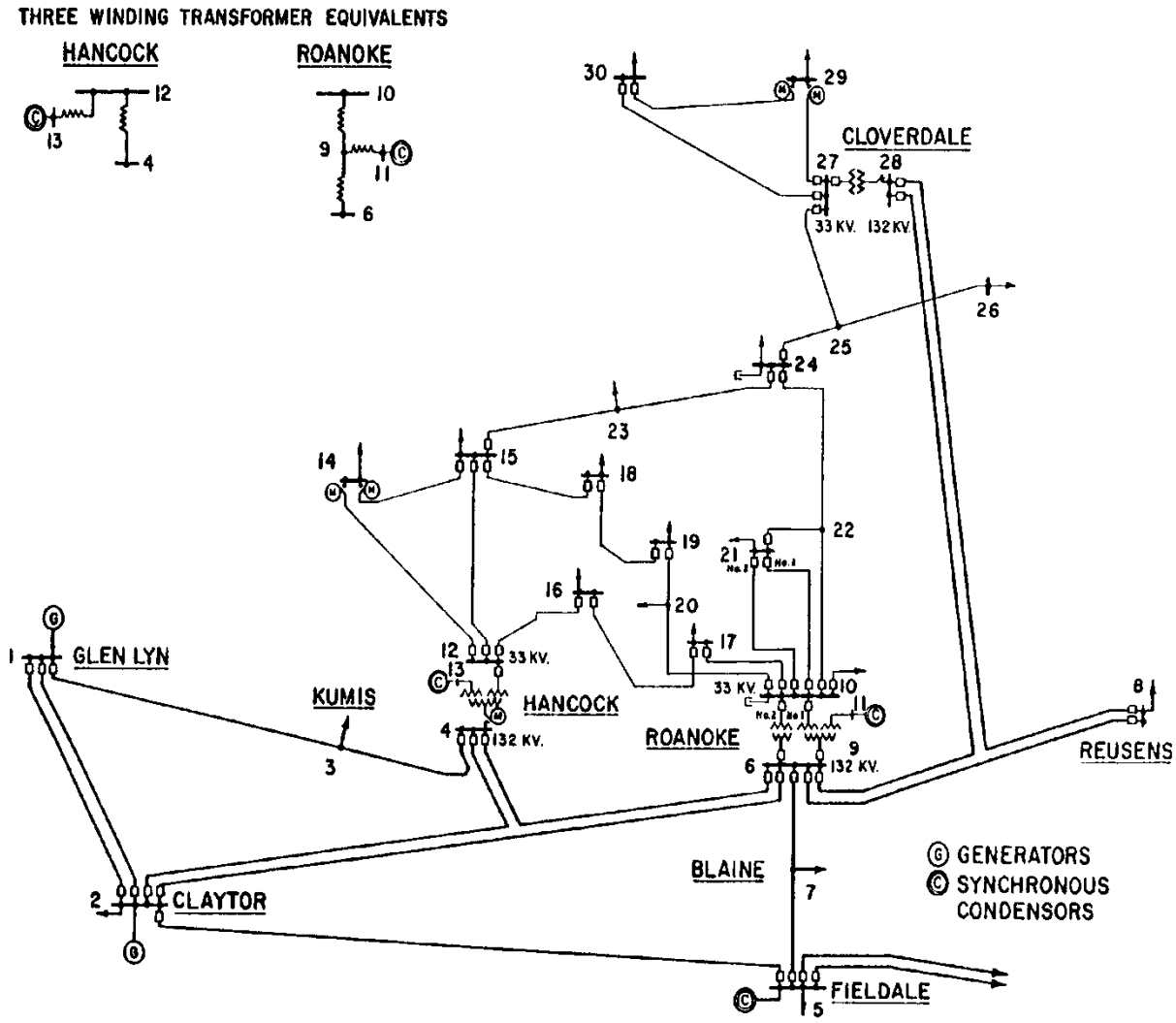


FIG (I-C): BUS-CODE DIAGRAM 30 BUS SYSTEM

TABLE (I-G): IMPEDANCE OR 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. Line charging one-half of total charging line.

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

Bus No.	Voltage		Generation	Generation	Load	Load
	Magnitude (p.u.)	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	1.2
4	1	0	0	0	7.6	1.6
5	1	0	0	0	94.2	19.0
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 Number	Voltage Magnitude p.u.	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 positions and the voltage bases. In the case of nominal turns ratio, this would equal 1.

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

Bus Number	Susceptance* p.u.
10	0.19
24	0.043

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

Cost Characteristics:

$C_1 = 50\ P_1\text{^}2 + 245\ P_1 + 105$ $/hr

$C_2 = 50\ P_2\text{^}2 + 351\ P_2 + 44.4$ $/hr

$C_8 = 50\ P_8\text{^}2 + 389\ P_8 + 40.6$ $/hr

Maximum and minimum active power constraint on the generator bus for the given system is 150 MW and 50 MW respectively. Voltage magnitude constraint for generator 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 1.01 0 30 30 150 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 0 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 0.1737 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];
disp(busdata)
disp(linedata)
lfybus
lfnewton
busout
bloss

B-Coefficients Calculated are 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

MATHEMATICAL STATEMENT OF NON-INFERIORTY

Single objective problems are characterized by complete ordering of their feasible solutions. Any two feasible solutions X_1 and X_2 are comparable in terms of the objective function; i.e. either

$$Z(X^1) = Z(X^2),\ \ Z(X^1) > Z(X^2),\ \ Z(X^1) < Z(X^2).$$

This comparison can be made for all the feasible solutions, and the solution X^* for which there exists no other solution X such that $Z(X) < Z(X^*)$ is called optimal solution for a minimization problem. But, in multiobjective problems, it is not possible to compare all the feasible solutions because the comparison on the basis of one objective function may contradict the comparison based on another objective function.

Suppose there are two objective functions,

$$Z(X) = [(Z_1(X), Z_2(X)] \qquad (1)$$

and two solutions $X^{1,}$ X^2 . Then,

$$Z(X^1) = [\ Z_1(X^1), Z_2(X^1)] \qquad (2)$$

$$Z(X^2) = [\ Z_1(X^2), Z_2(X^2)] \qquad (3)$$

X^1 is better than X^2 if

$$Z_1(X^1) < Z_1(X^2) \qquad \text{and } Z_2(X^1) \leq Z_2(X^2)$$

or

$$Z_1(X^1) \leq Z_1(X^2) \text{ and } Z_2(X^1) < Z_2(X^2)$$

but if $Z_1(X^1) < Z_1(X^2)$ AND $Z_2(X^1) > Z_2(X^2)$, then nothing can be said about the two solutions – X^1 , X^2 , i.e. they are incomparable. This is what is meant by partial ordering. All solutions are

not comparable on the basis of the values objective functions only. Since a complete order is not available, the notion of optimality must be dropped.

The partial ordering in multiobjective problems does not allow some feasible solutions to be eliminated. Inferior solutions, which are dominated by at least one feasible solution, may be dropped. Non-inferior solutions are the alternatives of interest.

Mathematically, a solution X is non-inferior for a minimization problem if there exist no fesible Y such that

$$Z_K(Y) \leq Z_K(X) \qquad VK= 1,2\ldots\ldots H \qquad (4)$$

and

$$Z_K(Y) < Z_K(X) \qquad \text{for at least one } K = 1,2\ldots\ldots h \qquad (5)$$

The non-inferior set generally includes many alternatives, all of which obviously cannot be selected. The objectives must be traded off against each other in moving from one non-inferior alternative to another and a strategy has to be adopted by the analyser to achieve optimum values as per his satisfaction level and requirements. The preferred alternative is called ***Target Point*** or the best compromise solution.

APPENDIX III

(i) MATLAB Programs for optimization of benchmark functions using PSO.

Code for the Optimization of Rosenbrock's Function

```
clc
disp(' we have to minimize f = 100(x1^2-x2)^2+(1-x1)^2 i.e. rosenbrock
function')
p=input('Enter the no. of particles in a swarm');           %no. of particles
it=input('Enter the no. of iterations');
x1=zeros(p,it);
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));
rp=0.4;
rg=0.5;
cp=2;
cg=2;
T=input('Enter the tolerance value');
x1(:,1)=unifrnd(0,2,1,p);
x2(:,1)=unifrnd(0,2,1,p);
v1(:,1)=rand(1,p);
v2(:,1)=rand(1,p);
for j=1:p
    f(j,1)= 100*( (x1(j,1))^2 - x2(j,1) )^2 + (1- x1(j,1))^2 ;
end
%Initial personal besst values
x1p=x1(:,1);
x2p=x2(:,1);
%for Initial Global best values updation
fmin=min(f(:,1));
for k=1:p
    if f(k,1)==fmin
        gb=k;
    else
    end
end
%Initial global best value
x1g=zeros(p);
x2g=zeros(p);
for k=1:p
x1g(k) = x1(gb,1);
x2g(k) = x2(gb,1);
end
fgm = min(f(:,1));
for i=1:it
    disp(sprintf('This is %d no. of iteration',i))
%for inertia weight W
    wmax=0.9;
    wmin=0.4;
    w = wmax-i*((wmax-wmin)/it);
```

```
for j=1:p
   v1(j,(i+1)) = w*v1(j,i) + rp*cp*(x1p(j)-x1(j,i)) + rg*cg*(x1g(j)-x1(j,i));
   v2(j,(i+1)) = w*v2(j,i) + rp*cp*(x2p(j)-x2(j,i)) + rg*cg*(x2g(j)-x2(j,i));
   x1(j,(i+1)) = x1(j,i) + v1(j,(i+1));
   x2(j,(i+1)) = x2(j,i) + v2(j,(i+1));
   f(j,(i+1))= 100*( (x1(j,(i+1)))^2 - x2(j,(i+1)) )^2 + (1- x1(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)= 100*( (x1p(j))^2 - x2p(j) )^2 + (1- x1p(j))^2 ;
    end
    for k=1:p
        if f(k,i)< fp(k)
            x1p(k)=x1(k,i);
            x2p(k)=x2(k,i);
        else
        end
    end
%for Global best values updation
    if min(f(:,(i+1)))<fgm
        fgm=min(f(:,(i+1)));
    else
    end
    for j=1:i
        for k=1:p
        if f(k,i)==fgm
            for l=1:p
                x1g(l) = x1(k,i);     %global best values
                x2g(l) = x2(k,i);
            end
        else
        end
        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
[r,c]=find(f==fgm);
minf=100*( (x1g(j))^2 - x2g(1) )^2 + (1- x1g(1))^2;
disp(sprintf('min value of function is %d and at values of x1=%d and x2=%d ',
minf,x1g(1),x2g(1)))
```

Code for the Optimization of Beale's Function

```
clc
disp(' we have to 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 function')
p=input('Enter the no. of particles in a swarm');           %no. of particles
it=input('Enter the no. of iterations');
x1=zeros(p,it);
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));
rp=0.4;
rg=0.5;
cp=2;
cg=2;
T=input('Enter the tolerance value');
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);
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;
end
%Initial personal besst values
x1p=x1(:,1);
x2p=x2(:,1);
%for Initial Global best values updation
fmin=min(f(:,1));
for k=1:p
    if f(k,1)==fmin
        gb=k;
    else
    end
end
%Initial global best value
x1g=zeros(p);
x2g=zeros(p);
for k=1:p
x1g(k) = x1(gb,1);
x2g(k) = x2(gb,1);
end
fgm = min(f(:,1));
for i=1:it
    disp(sprintf('This is %d no. of iteration',i))
%for inertia weight W
    wmax=1.0;
    wmin=0.2;
    w = wmax-i*((wmax-wmin)/it);
for j=1:p
   v1(j,(i+1)) = w*v1(j,i) + rp*cp*(x1p(j)-x1(j,i)) + rg*cg*(x1g(j)-x1(j,i));
   v2(j,(i+1)) = w*v2(j,i) + rp*cp*(x2p(j)-x2(j,i)) + rg*cg*(x2g(j)-x2(j,i));
   x1(j,(i+1)) = x1(j,i) + v1(j,(i+1));
   x2(j,(i+1)) = x2(j,i) + v2(j,(i+1));
```

```
   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;
    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
    f(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;
    end
    for k=1:p
        if f(k,i)< fp(k)
            x1p(k)=x1(k,i);
            x2p(k)=x2(k,i);
        else
        end
    end
%for Global best values updation
    if min(f(:,(i+1)))<fgm
        fgm=min(f(:,(i+1)));
    else
    end
        for j=1:i
        for k=1:p
        if f(k,i)==fgm
            for l=1:p
                x1g(l) = x1(k,i);     %global best values
                x2g(l) = x2(k,i);
            end
        else
        end
        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
[r,c]=find(f==fgm);
disp(sprintf('min value of function is %d and at values of x1=%d and x2=%d
',fgm,x1(r,c),x2(r,c)))
```

Code for the Optimization of Sphere Function

```
clc
disp(' we have to minimize f = 100(x1^2-x2)^2+(1-x1)^2 i.e. rosenbrock
function')
p=input('Enter the no. of particles in a swarm');           %no. of particles
it=input('Enter the no. of iterations');
x1=zeros(p,it);x2=zeros(p,it);x3=zeros(p,it);
v1=zeros(p,it);v2=zeros(p,it);v3=zeros(p,it);
f=zeros(p,it);
x1g=zeros(p);x2g=zeros(p);x3g=zeros(p);
fp=zeros(1,p);
df=zeros(1,(it-1));
rp=0.4;rg=0.5;
cp=2;cg=2;
T=input('Enter the tolerance value');
x1(:,1)=unifrnd(-1,1,1,p);x2(:,1)=unifrnd(-1,1,1,p);
x3(:,1)=unifrnd(-1,1,1,p);
v1(:,1)=rand(1,p);v2(:,1)=rand(1,p);v3(:,1)=rand(1,p);
for j=1:p
      f(j,1)= x1(j,1)^2 + x2(j,1)^2 + x3(j,1)^2;
end
%Initial personal besst values
x1p=x1(:,1);
x2p=x2(:,1);
x3p=x2(:,1);
%for Initial Global best values updation
fmin=min(f(:,1));
for k=1:p
    if f(k,1)==fmin
        gb=k;
    else
    end
end
%Initial global best value
for k=1:p
x1g(k) = x1(gb,1);
x2g(k) = x2(gb,1);
x3g(k) = x3(gb,1);
end
fgm = min(f(:,1));

for i=1:it
    disp(sprintf('This is %d no. of iteration',i))
%for inertia weight W
    wmax=1;
    wmin=0.3;
    w = wmax-i*((wmax-wmin)/it);
for j=1:p
   v1(j,(i+1)) = w*v1(j,i) + rp*cp*(x1p(j)-x1(j,i)) + rg*cg*(x1g(j)-x1(j,i));
   v2(j,(i+1)) = w*v2(j,i) + rp*cp*(x2p(j)-x2(j,i)) + rg*cg*(x2g(j)-x2(j,i));
   v3(j,(i+1)) = w*v3(j,i) + rp*cp*(x3p(j)-x3(j,i)) + rg*cg*(x3(j)-x3(j,i));
   x1(j,(i+1)) = x1(j,i) + v1(j,(i+1));
   x2(j,(i+1)) = x2(j,i) + v2(j,(i+1));
   x3(j,(i+1)) = x3(j,i) + v3(j,(i+1));
   f(j,(i+1))= x1(j,(i+1))^2 + x2(j,(i+1))^2 + x3(j,(i+1))^2;
```

```
 end
 %To find change in the values of f
    for j=1:p
        df(j,i)= abs(f(j,(i+1))-f(j,i)) ;
    end
 %personal best values updation
    for j=1:p
    fp(j)= x1p(j)^2 + x2p(j)^2;
    end
    for k=1:p
        if f(k,i)< fp(k)
            x1p(k)=x1(k,i);
            x2p(k)=x2(k,i);
            x3p(k)=x3(k,i);
        else
        end
    end
%for Global best values updation
    if min(f(:,(i+1)))<fgm
        fgm=min(f(:,(i+1)));
    else
    end

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

print = [x1(:,i)      x2(:,i)     x3(:,i)     v1(:,i)     v2(:,i)
f(:,i)];
    disp('     x1           x2         x3         v1         v2          f')
    disp(print)
%Stoping criterion
    ki=0;
    for j=1:p
        if (df(j,i)<=10^(-T))
            ki=ki+1;
        end
    end
    if ki >= p
        break
    end
end

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

Code for the Optimization of Sphere Function

```
clc
disp(' we have to 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 function')
p=input('Enter the no. of particles in a swarm');           %no. of particles
it=input('Enter the no. of iterations');
x1=zeros(p,it);
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));
rp=0.4;
rg=0.5;
cp=2;
cg=2;
T=input('Enter the tolerance value');
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);
for j=1:p
    f(j,1)=(x1(j,1)+2*x2(j,1)-7)^2+(2*x1(j,1)+x2(j,1)-5)^2;
end
%Initial personal besst values
x1p=x1(:,1);
x2p=x2(:,1);
%for Initial Global best values updation
fmin=min(f(:,1));
for k=1:p
    if f(k,1)==fmin
        gb=k;
    else
    end
end
%Initial global best value
x1g=zeros(p);
x2g=zeros(p);
for k=1:p
x1g(k) = x1(gb,1);
x2g(k) = x2(gb,1);
end
fgm = min(f(:,1));
for i=1:it
    disp(sprintf('This is %d no. of iteration',i))
%for inertia weight W
    wmax=1.0;
    wmin=0.2;
    w = wmax-i*((wmax-wmin)/it);

for j=1:p
 v1(j,(i+1)) = w*v1(j,i) + rp*cp*(x1p(j)-x1(j,i)) + rg*cg*(x1g(j)-x1(j,i));
 v2(j,(i+1)) = w*v2(j,i) + rp*cp*(x2p(j)-x2(j,i)) + rg*cg*(x2g(j)-x2(j,i));
 x1(j,(i+1)) = x1(j,i) + v1(j,(i+1));
 x2(j,(i+1)) = x2(j,i) + v2(j,(i+1));
```

```
 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;
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
        f(j)=(x1p(j)+2*x2p(j)-7)^2+(2*x1p(j)+x2p(j)-5)^2;
    end
    for k=1:p
        if f(k,i)< fp(k)
            x1p(k)=x1(k,i);
            x2p(k)=x2(k,i);
        else
        end
    end
%for Global best values updation
    if min(f(:,(i+1)))<fgm
        fgm=min(f(:,(i+1)));
    else
    end
    for j=1:i
        for k=1:p
        if f(k,i)==fgm
            for l=1:p
                x1g(l) = x1(k,i);     %global best values
                x2g(l) = x2(k,i);
            end
        else
        end
        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
[r,c]=find(f==fgm);
disp(sprintf('min value of function is %d and at values of x1=%d and x2=%d
',fgm,x1(r,c),x2(r,c)))
```

(ii) MATLAB Programs for MELD in IEEE 5, 14 and 30 bus systems using PSO.

Code for the MELD in IEEE 5 bus system

```
clear all
clc
disp(' we have to minimize the cost function of a 3 machine system')
p=input('Enter the no. of particles in a swarm');           %no. of particles
it=input('Enter the no. of iterations');
a=10^(-4)*[50 50];
b=10^(-2)*[351 389];
c=[44.4 40.6];
B=10^(-2)*[0.0349 0.0086; -0.0055 0.0371];
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);
rp=0.4;
rg=0.5;
cp=2;
cg=2;
pd=160;
p1g=zeros(p);
p2g=zeros(p);
fp=zeros(1,p);
plp=zeros(1,p);
k=1000;
w1=1;
w2=18;
% Initial values i.e. 0th iteration
n=1;
while n==1
    for j=1:p
    p1(j,1)=unifrnd(50,150,1);
    p2(j,1)=pd-p1(j,1);
    if p2(j,1)<50&&p2(j,1)>150
        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)))...
            + w2*pl(j,1) + 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;
    w = wmax-i*((wmax-wmin)/it);
%For calculatiing velocities for updation
for j=1:p
  v1(j,(i+1)) = w*v1(j,i) + rp*cp*(p1p(j)-p1(j,i)) + rg*cg*(p1g(j)-p1(j,i));
  v2(j,(i+1)) = w*v2(j,i) + rp*cp*(p2p(j)-p2(j,i)) + rg*cg*(p2g(j)-p2(j,i));
end
%V(min) and V(max) constraint
    for j=1:p
        if v1(j,(i+1))< -15
            v1(j,(i+1))= -15;
        end
        if v2(j,(i+1))< -15
            v2(j,(i+1))= -15;
        end
        if 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
        p1(j,(i+1)) = p1(j,i) + v1(j,(i+1));
        p2(j,(i+1)) = p2(j,i) + 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)))+...
            w2*pl(j,(i+1)) + 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))) + w2*plp(j) +  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)
%% Plotting the swarm
    clf
    plot(p1(:, i), p2(:, i),'o')   % drawing swarm movements
    axis([70 110 70 110]);
pause(.2)
%Stoping criterion (df(j,i)<=10^(-6))&& &&(csp(j,i)<=10^(-6))
    ki=0;
    for j=1:p
        if ((df(j,i)<=10^(-2))&&(sp(j,i)<=10^(-2)))
            ki=ki+1;
        end
    end
    if ki >= p
        break
    end
end
disp(' we have to minimize the cost function of a 2 machine 5 bus system')
disp(sprintf('No. of particles used in a swarm = %d',p))
disp(sprintf('Max. no. of iterations entered = %d\n',it))
disp(sprintf('Total demand of power Pd = %d \n',pd))
disp('Initial values of generations of 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('About this run')
disp('1. The Constraints has been included as absolute value.')
disp('2. Random values between the limits of generation have been taken for
each generator as the different starting point.')
disp('3. Correct values of B coefficients have been fed.')
disp(sprintf('4. K taken = %d',k))
disp(sprintf('5. w1 and w2 taken = %d and %d',w1,w2))
```

Code for the MELD in IEEE 14 bus system

```
clear all
clc
disp(' we have to minimize the cost function of a 3 machine system')
p=input('Enter the no. of particles in a swarm');           %no. of particles
it=input('Enter the no. of iterations');
a=10^(-4)*[50 50 50];
b=10^(-2)*[245 351 389];
c=[105 44.4 40.6];
B=10^(-2)*[0.0349 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);
p1=zeros(p,it);
p2=zeros(p,it);
p3=zeros(p,it);
v1=zeros(p,it);
v2=zeros(p,it);
v3=zeros(p,it);
f=zeros(p,it);
df=zeros(p,it);
sp=zeros(p,it);
csp=zeros(p,it);
pl=zeros(p,it);
c1=zeros(p,it);
c2=zeros(p,it);
c3=zeros(p,it);
C=zeros(p,it);
rp=0.4;
rg=0.5;
cp=2;
cg=2;
pd=259;
p1g=zeros(p);
p2g=zeros(p);
p3g=zeros(p);
fp=zeros(1,p);
plp=zeros(1,p);
k=5;
w1=1;
w2=0;
```

```
n=1;
while n==1
    for j=1:p
    p1(j,1)=unifrnd(30,120,1);
    p2(j,1)=unifrnd(30,120,1);
    p3(j,1)=pd-p1(j,1)-p2(j,1);
    if p3(j,1)<30&&p3(j,1)>120
        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))) + w2*pl(j,1) +
k*abs(pd+pl(j,1)-p1(j,1)-p2(j,1)-p3(j,1));
end
%0th iteration data display
disp('this is the 0th iteration')
print0 = [p1(:,1) p2(:,1) p3(:,1) v1(:,1) v2(:,1) v3(:,1) f(:,1) c1(:,1)
          c2(:,1) c3(:,1) C(:,1)];
disp('    P1     P2    P3    V1     v2    V3   f    c1     c2     c3     C ')
disp(print0)
%Initial personal besst values
p1p=p1(:,1);
p2p=p2(:,1);
p3p=p3(:,1);
%for Initial Global best values updation
fmin=min(f(:,1));
for m=1:p
    if f(m,1)==fmin
        gb=m;
    else
    end
end
%Initial global best value
for m=1:p
```

```
p1g(m) = p1(gb,1);
p2g(m) = p2(gb,1);
p3g(m) = p3(gb,1);
end
fgm = min(f(:,1));
%Main iterations starts from here
for i=1:it
    disp(sprintf('This is iteration no.= %d',i))
%for inertia weight W
    wmax=0.9;
    wmin=0.4;
    w = wmax-i*((wmax-wmin)/it);
%For calculatiing velocities for updation
for j=1:p
v1(j,(i+1)) = w*v1(j,i) + rp*cp*(p1p(j)-p1(j,i)) + rg*cg*(p1g(j)-p1(j,i));
v2(j,(i+1)) = w*v2(j,i) + rp*cp*(p2p(j)-p2(j,i)) + rg*cg*(p2g(j)-p2(j,i));
v3(j,(i+1)) = w*v3(j,i) + rp*cp*(p3p(j)-p3(j,i)) + rg*cg*(p3g(j)-p3(j,i));
end
%V(min) and V(max) constraint
    for j=1:p
        if v1(j,(i+1))< -15
            v1(j,(i+1))= -15;
        end
        if v2(j,(i+1))< -15
            v2(j,(i+1))= -15;
        end
        if v3(j,(i+1))< -15
            v3(j,(i+1))= -15;
        end
        if v1(j,(i+1))> 60
            v1(j,(i+1))= 60;
        end
        if v2(j,(i+1))> 60
            v2(j,(i+1))= 60;
        end
        if v3(j,(i+1))> 60
            v3(j,(i+1))= 60;
        end
    end
%Updation of p values
    for j=1:p
        p1(j,(i+1)) = p1(j,i) + v1(j,(i+1));
        p2(j,(i+1)) = p2(j,i) + v2(j,(i+1));
        p3(j,(i+1)) = p3(j,i) + v3(j,(i+1));
    end
%Pmin and Pmax constraint
    for j=1:p
        if p1(j,(i+1))< 30
            p1(j,(i+1))= 30;
        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))> 120
            p1(j,(i+1))= 120;
        end
        if p2(j,(i+1))> 120
            p2(j,(i+1))= 120;
        end
        if p3(j,(i+1))> 120
            p3(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)) 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))) +...
          w2*pl(j,(i+1)) + k*abs(pd+pl(j,(i+1))-p1(j,(i+1))-p2(j,(i+1))-
p3(j,(i+1)));
    end
%personal best values updation
    %For losses formulation (PL)
    for j=1:p
        plp(j)= [p1p(j) p2p(j) p3p(j)]*B*[p1p(j) p2p(j) p3p(j)]'+[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))) + w2*plp(j) +
k*abs(pd+plp(j)-p1p(j)-p2p(j)-p3p(j));
    end
    for m=1:p
        if f(m,i)< fp(m)
            p1p(m)=p1(m,(i+1));
            p2p(m)=p2(m,(i+1));
            p3p(m)=p3(m,(i+1));
        else
        end
    end

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

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

```
                p3g(l) = p3(m,j);
            end
        else
        end
        end
    end

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

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

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

%Stoping criterion &&(csp(j,i)<=10^(-6))  (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

disp(' we have to minimize the cost function of a 3 machine 14 Bus System')
disp(sprintf('No. of particles used in a swarm = %d',p))
disp(sprintf('Max. no. of iterations entered = %d\n',it))

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

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

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

```
disp(sprintf('No. of total Iterations took place = %d \n',i))
disp(sprintf('Total loses in the lines Pl = %d \n',pl(1,i)))
disp(sprintf('Minimum cost incured = %d \n',C(1,i)))
disp('\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('About this run')
disp('1. The Constraints has been included as absolute value.')
disp('2. Random values between the limits of generation have been taken for
each generator as the different starting point.')
disp('3. Correct values of B coefficients have been fed.')
disp(sprintf('4. K taken = %d',k))
disp(sprintf('5. w1 and w2 taken = %d and %d',w1,w2))
```

Code for the MELD in IEEE 30 bus system

```
clear all
clc
disp(' we have to minimize the cost function of a 3 machine system')
p=input('Enter the no. of particles in a swarm');           %no. of particles
it=input('Enter the no. of iterations');
a=10^(-4)*[50 50 50];
b=10^(-2)*[245 351 389];
c=[105 44.4 40.6];
B=10^(-2)*[0.0307 0.0129 -0.0002; 0.0129 0.0152 -0.0011; -0.0002 -0.0011
0.0190];
p1=zeros(p,it);
p2=zeros(p,it);
p3=zeros(p,it);
v1=zeros(p,it);
v2=zeros(p,it);
v3=zeros(p,it);
f=zeros(p,it);
df=zeros(p,it);
sp=zeros(p,it);
csp=zeros(p,it);
pl=zeros(p,it);
c1=zeros(p,it);
c2=zeros(p,it);
c3=zeros(p,it);
C=zeros(p,it);
rp=0.4;
rg=0.5;
cp=2;
cg=2;
pd=283.4;
p1g=zeros(p);
p2g=zeros(p);
p3g=zeros(p);
fp=zeros(1,p);
plp=zeros(1,p);
k=1000;
w1=1;
```

```
w2=13;
% Initial values i.e. 0th iteration
n=1;
while n==1
    for j=1:p
    p1(j,1)=unifrnd(30,120,1);
    p2(j,1)=unifrnd(30,120,1);
    p3(j,1)=283.4-p1(j,1)-p2(j,1);
    if p3(j,1)<30&&p3(j,1)>120
        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))) + w2*pl(j,1) + k*abs(pd+pl(j,1)-p1(j,1)-p2(j,1)-p3(j,1));
end
%0th iteration data display
disp('this is the 0th iteration')
print0 = [p1(:,1) p2(:,1) p3(:,1) v1(:,1) v2(:,1) v3(:,1) f(:,1) c1(:,1)
          c2(:,1) c3(:,1) C(:,1)];
disp('  P1     P2    P3    V1     v2    V3   f    c1    c2      c3      C ')
disp(print0)
%Initial personal besst values
p1p=p1(:,1);
p2p=p2(:,1);
p3p=p3(:,1);
%for Initial Global best values updation
fmin=min(f(:,1));
for m=1:p
    if f(m,1)==fmin
        gb=m;
    else
    end
end
%Initial global best value
for m=1:p
```

```
p1g(m) = p1(gb,1);
p2g(m) = p2(gb,1);
p3g(m) = p3(gb,1);
end
fgm = min(f(:,1));
%Main iterations starts from here
for i=1:it
    disp(sprintf('This is iteration no.= %d',i))
%for inertia weight W
    wmax=0.9;
    wmin=0.4;
    w = wmax-i*((wmax-wmin)/it);
%For calculatiing velocities for updation
for j=1:p
v1(j,(i+1)) = w*v1(j,i) + rp*cp*(p1p(j)-p1(j,i)) + rg*cg*(p1g(j)-p1(j,i));
v2(j,(i+1)) = w*v2(j,i) + rp*cp*(p2p(j)-p2(j,i)) + rg*cg*(p2g(j)-p2(j,i));
v3(j,(i+1)) = w*v3(j,i) + rp*cp*(p3p(j)-p3(j,i)) + rg*cg*(p3g(j)-p3(j,i));
end
%V(min) and V(max) constraint
    for j=1:p
        if v1(j,(i+1))< -15
            v1(j,(i+1))= -15;
        end
        if v2(j,(i+1))< -15
            v2(j,(i+1))= -15;
        end
        if v3(j,(i+1))< -15
            v3(j,(i+1))= -15;
        end
        if v1(j,(i+1))> 60
            v1(j,(i+1))= 60;
        end
        if v2(j,(i+1))> 60
            v2(j,(i+1))= 60;
        end
        if v3(j,(i+1))> 60
            v3(j,(i+1))= 60;
        end
    end
%Updation of p values
    for j=1:p
        p1(j,(i+1)) = p1(j,i) + v1(j,(i+1));
        p2(j,(i+1)) = p2(j,i) + v2(j,(i+1));
        p3(j,(i+1)) = p3(j,i) + v3(j,(i+1));
    end
%Pmin and Pmax constraint
    for j=1:p
        if p1(j,(i+1))< 30
            p1(j,(i+1))= 30;
        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))> 120
            p1(j,(i+1))= 120;
        end
        if p2(j,(i+1))> 120
            p2(j,(i+1))= 120;
        end
        if p3(j,(i+1))> 120
            p3(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)) p3(j,(i+1))]*B*[p1(j,(i+1))
p2(j,(i+1)) p3(j,(i+1))]';
    end
%Main objective function
for j=1:p
f(j,(i+1))= w1*((a(1)*(p1(j,(i+1)))^2 + b(1)*p1(j,(i+1)) + c(1)) +...
            (a(2)*(p2(j,(i+1)))^2 + b(2)*p2(j,(i+1)) + c(2))+...
            (a(3)*(p3(j,(i+1)))^2 + b(3)*p3(j,(i+1)) + c(3))) +...
             w2*pl(j,(i+1)) + k*abs(pd+pl(j,(i+1))-p1(j,(i+1))-p2(j,(i+1))-
             p3(j,(i+1)));
end
%personal best values updation
%For losses formulation (PL)
    for j=1:p
        plp(j)= [p1p(j) p2p(j) p3p(j)]*B*[p1p(j) p2p(j) p3p(j)]';
    end
    for j=1:p
    fp(j)= w1*((a(1)*(p1p(j))^2 + b(1)*p1p(j) + c(1)) + (a(2)*(p2p(j))^2 +
           b(2)*p2p(j) + c(2)) + (a(3)*(p3p(j))^2 + b(3)*p3p(j) + c(3))) +
           w2*plp(j) +  k*abs(pd+plp(j)-p1p(j)-p2p(j)-p3p(j));
    end
    for m=1:p
        if f(m,i)< fp(m)
            p1p(m)=p1(m,(i+1));
            p2p(m)=p2(m,(i+1));
            p3p(m)=p3(m,(i+1));
        else
        end
    end
%for Global best values updation
    if min(f(:,(i+1)))<fgm
        fgm=min(f(:,(i+1)));
    else
    end
    for j=1:(i+1)
        for m=1:p
        if f(m,j)==fgm
            for l=1:p
                p1g(l) = p1(m,j);      %global best values
                p2g(l) = p2(m,j);
                p3g(l) = p3(m,j);
            end
        else
        end
```

```
        end
    end
%For cost calculation
    for j=1:p
         c1(j,(i+1)) = a(1)*(p1(j,(i+1)))^2 + b(1)*p1(j,(i+1)) + c(1);
         c2(j,(i+1)) = a(2)*(p2(j,(i+1)))^2 + b(2)*p2(j,(i+1)) + c(2);
         c3(j,(i+1)) = a(3)*(p3(j,(i+1)))^2 + b(3)*p3(j,(i+1)) + c(3);
         C(j,(i+1)) = c1(j,(i+1)) + c2(j,(i+1)) + c3(j,(i+1));
    end
%To find change in the values of f, equality constraint and change in cost
    for j=1:p
        df(j,i)= abs(f(j,(i+1))-f(j,i)) ;
        sp(j,i)= abs(pd+pl(j,(i+1))-p1(j,(i+1))-p2(j,(i+1))-p3(j,(i+1)));
        csp(j,i)= abs(C(j,(i+1))-C(j,i));
    end
print = [p1(:,(i+1)) p2(:,(i+1)) p3(:,(i+1)) v1(:,(i+1)) v2(:,(i+1)
         v3(:,(i+1)) f(:,(i+1)) c1(:,(i+1)) c2(:,(i+1)) c3(:,(i+1))
         C(:,(i+1))];
disp('   P1    P2     P3     V1    v2     V3     f     c1     c2    c3   C ')
disp(print)
%Stoping criterion  &&(csp(j,i)<=10^(-6))
    ki=0;
    for j=1:p
        if ((df(j,i)<=10^(-6))&&(sp(j,i)<=10^(-6)))
            ki=ki+1;
        end
    end
    if ki >= p
        break
    end
end
disp(' we have to minimize the cost function of a 3 machine 30 Bus System')
disp(sprintf('No. of particles used in a swarm = %d',p))
disp(sprintf('Max. no. of iterations entered = %d\n',it))
disp(sprintf('Total demand of power Pd = %d \n',pd))
disp('Initial values of generations of 3 generators')
initial=[p1(:,1) p2(:,1) p3(:,1)];
disp('    P1        P2        P3   ')
disp(initial)
disp(sprintf('\nPD+Pl = %d',pd+pl(1,i)))
disp(sprintf('\nP1+P2+P3=%d\n',p1(1,i)+p2(1,i)+p3(1,i)))
disp(sprintf('No. of total Iterations took place = %d \n',i))
disp(sprintf('Total loses in the lines Pl = %d \n',pl(1,i)))
disp(sprintf('Minimum cost incured = %d \n',C(1,i)))
disp('Final values of generations of the three generators')
disp(sprintf('P1=%d',p1(1,i)))
disp(sprintf('P2=%d',p2(1,i)))
disp(sprintf('P3=%d',p3(1,i)))
disp('About this run')
disp('1. The Constraints has been included as absolute value.')
disp('2. Random values between the limits of generation have been taken for
each generator as the different starting point.')
disp('3. Correct values of B coefficients have been fed.')
disp(sprintf('4. K taken = %d',k))
disp(sprintf('5. w1 and w2 taken = %d and %d',w1,w2))
```

REFRENCES

[1] Kothari D. P. and Dhillon J. S., 'Power System Optimization', Prentice Hall of India, 2006.

[2] Chowdhury B. H. and Rahman S., 'A review of recent advances in economic dispatch', IEEE transaction on power system, Vol. 5, No. 4, Nov. 1990, pp. 1248-1259.

[3] Deschamps D., 'Optimization in power system planning', In El-Abiad, AH. Ed. Power system analysis and planning, London, Hemishpere Publishing Corporation, 1981, pp. 201-203.

[4] Deschamps D., 'Optimization in power system planning', In El-Abiad, AH. Ed. Power system analysis and planning, London, Hemishpere Publishing Corporation, 1981, pp. 203-205.

[5] Deschamps D., 'Optimization in power system planning', In El-Abiad, AH. Ed. Power system analysis and planning, London, Hemishpere Publishing Corporation, 1981, pp. 205-208.

[6] Lee K. Y. et. al, 'Adaptive hopfield neural network for economic dispatch', IEEE transactions on power systems, Vol. 18, No. 2, Feb. 2003, pp. 519-529.

[7] Ongsakul W. and Ruangpayoongsak R., 'Constrained economic dispatch by simulated annealing genetic algorithms', IEEE power engineering society international conference on power industry computer application, 2001, pp. 207-212.

[8] Hong Y. Y. and Li C. L., 'Genetic algorithms based economic dispatch for cogeneration units considering multiplant multibuyer wheeling', IEEE Transactions on Power Systems, Feb 2002, Vol. 17, Issue. 1, pp. 134-140.

[9] Eberhart R. C. and Shi Y., 'Particle Swarm Optimization: Developments, Applications and Resources', IEEE Transactions on Evolutionary Computation, 2001, Vol. 1, pp. 81-86.

[10] Sum-Im, T., 'Economic dispatch by ant colony search algorithm', 'IEEE Conference on Cybernetics and Intelligent Systems', Dec. 2004, Vol. 1, pp. 416-421.

[11] Sasson A. M., 'Non-linear programming solutions for the load FLow, minimum loss and economic dispatching problem', IEEE Transaction on power apparatus and systems,1969, pp. 399–409.

[12] Cohon J. L., 'Multiple objective programming and planning', Academic press, New York, 1978.

[13] Nanda J., Kothari D. P. and Lingamurthy K. S., 'Economic emission load dispatch through goal programming techniques', IEEE Winter Meeting, New Orleans, Louisiana, 1988, Vol. 3, pp. 26-32.

[14] Wadhwa C. L. and Jain N. K., 'Multiple objective optimal load flow: a new perspective', IEE Proceedings, 1990, Vol. 137, pp. 59-64.

[15] Nangia Uma, Jain N. K. and Wadhwa C. L., 'Investigations of multiobjective optimal load flow study by sequential goal programming', Journal of IE, 1996, Vol. 77, pp. 99-103.

[16] Nangia Uma, Jain N. K. and Wadhwa C. L., 'Multiobjective load flow studies through maximization of minimum relative attainments', Journal of IE, 1996, Vol. 77, pp. 154-159.

[17] Nangia Uma, Jain N. K. and Wadhwa C. L., 'Surrogate worth tradeoff technique for multiobjective optimal power flows', IEE Proc- Generation, Transmission, Distribution, 1997, Vol. 144, pp.547-553.

[18] Nangia Uma, Jain N. K. and Wadhwa C. L., 'A new interactive step method for multiobjective optimal load flow study', Journal of IE, Vol. 78, March 1998, pp. 225-228.

[19] Nangia Uma, Jain N. K. and Wadhwa C. L., 'Non-inferior set estimation for multiobjective optimal load flow study', Journal of IE, 1998, Vol. 78, pp. 229-235.

[20] Nangia Uma, Jain N. K. and Wadhwa C. L., 'Optimal weight assessment based on range of objectives in multiobjective optimal load flow study', IEE Proc.- Generation, Transmission and Distribution , 1998, Vol. 145, pp. 65-69.

[21] Nangia Uma, Jain N. K. and Wadhwa C. L., 'Multiobjective optimal load flow based on ideal distance minimization in 3d space', Electrical power and energy systems, 2001, Issue 23, pp. 847-855.

[22] Nangia Uma, Jain N. K. and Wadhwa C. L., 'Comprehensive comparison of various multiobjective techniques', Engineering Intelligent Systems Journal, 2003, Vol. 11, pp. 123-132.

[23] Abido M. A., 'Multiobjective evolutionary algorithms for electric power dispatch problem', IEEE Transactions on Evolutionary Computation, 2006, Vol. 10, pp. 315–329.

[24] Basu K., Bhattacharya A., Chowdhury S. and Chowdhury S. P., 'Planned scheduling for economic power sharing in a chp-based micro-grid', IEEE Trans. on Power Sys., 2012, Vol. 27, pp. 30–38.

[25] Park J. B., Lee K.S., Shin J. R. and Lee K. Y., 'A particle swarm optimization for economic dispatch with nonsmooth cost functions', IEEE Trans. Power Sys., 2005, Vol. 20, pp. 34–42.

[26] Selvakumar A. I. and Thanushkodi K., 'A new particle swarm optimization solution to nonconvex economic dispatch problems', IEEE Trans. Power Sys., 2007, Vol. 22, pp. 42–51.

[27] Mahor A., Prasad V. and Rangnekar S., 'Economic dispatch using particle swarm optimization: A review', Renewable and Sustainable Energy Reviews, 2009, Issue 13, pp. 2134–2141.

[28] Meng K., Wang H. G., Dong Z. Y. and Wong K. P., 'Quantum-inspired particle swarm optimization for valve-point economic load dispatch', IEEE Trans. Power Sys., 2010, Vol. 25, pp. 215–221.

[29] Chakraborty S., Senjyu T., Yona A., Saber A. Y. and Funabashi T., '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, pp. 1042–1052.

[30] Niknam T. and Golestaneh F., 'Enhanced adaptive particle swarm optimization algorithm for dynamic economic dispatch of units considering valve-point effects and ramp rates', IET Gener. Transm. Distrib., 2012, Vol. 6, pp. 424–435.

[31] Huang C. M. and Wang F. L., 'An RBF network with OLS and EPSO algorithms for real-time power dispatch', IEEE Trans. Power Sys., 2007, Vol. 22, pp. 215–221.

[32] Niknam T. and Doagou-Mojarrad H., 'Multiobjective economic emission dispatch by multiobjective Θ-particle swarm optimization', IET Gener. Transm. Distrib., 2012, Vol. 6, pp. 363–377.

[33] Nikman T., Narimani M. R., Aghaei J. and Azizipanah-Abarghooee R., 'Improved particle swarm optimization for multi-objective optimal power considering the cost, loss, emission and voltage stability index', IET Gener. Transm. Distrib., 2012, Vol. 6, pp. 515–527.

[34] Steveson W. D., 'Elements of Power System Analysis', McGraw Hill Book Co., 1962.

[35] Jared L. C., 'Multi-objective Programming and Planning', Academic Press New York, 1978.

[36] Kennedy J. and Eberhart R., 'Particle swarm optimization', IEEE Proc. Int. Conf. Neural Networks, 1995, pp. 1942–1948.

[37] Sudhakaran M., Ajay-D-Vimal Raj P. and Palanivelu T. G., 'Application of particle swarm optimization for economic load dispatch problems', IEEE International Conference, 2007, pp. 1–7.

[38] Shi Y. and Eberhart R., 'A modified Particle Swarm Optimizer', IEEE proc. int. Conf. on Evolutionary Computation, 1998, pp. 69–73.

[39] Shi Y. and Eberhart R., 'Parameter selection in particle swarm optimization', In portovw, Saravanan N, waagen D and Eiben AE(eds) Evolutionary Programming VII, 1998, PP 611- 616

[40] Shi Y. amd Eberhrart R., 'Emperical study of Particle swarm optimization', In proc. congress on evolutionary computation, 1999, vol. 3, pp. 1945-1950.

[41] Eberhart R. C. and Shi Y., 'Comparison between genetic algorithms and particle swarm optimization', IEEE Proc. on Int. Conf. Evol. Compu., May 1998, pp. 611-616.

[42] Niknam T. and Golestaneh F., 'Enhanced adaptive particle swarm optimization algorithm for dynamic economic dispatch of units considering valve-point effects and ramp rates', IET Gener. Transm. Distrib., 2012, Vol. 6, pp. 424–435.

[43] Gaing Z. L., 'Particle swarm optimization to solving the economic dispatch considering the generator constraints', IEEE Trans. on Power Systems, Aug. 2003, Vol. 18, No. 3, pp. 1187-1195.

[44] Woods J. and Wollenberg B. F, 'Power generation, operation, and control', John Wiley & sons, 1996.

[45] Saadat H., 'Power System Analysis', McGraw-hill companies, Inc, 1999.

[46] Stevenson W.D., JR., 'Elements of power system analysis', McGraw Hill Book Co., 2nd Ed., 1962, p. 219.