MAULANA ABUL KALAM AZAD UNIVERSITY OF TECHNOLOGY, WEST BENGAL

Economic Integration of Renewable Energy Sources in Smart grid.

Submitted in partial fulfillment of the requirements for the degree of

Bachelor of Technology

In

Electrical Engineering

By

TANMOY PAUL

UNIVERSITY ROLL NO: 35501618001

REGISTRATION NO: 183550110035 OF 2018-2022

Under the guidance of

Dr. Sandip Chanda

Electrical Engineering Department

GHANI KHAN CHOUDHURY INSTITUTE OF ENGINEERING AND TECHNOLOGY

June 2022

GHANI KHAN CHOUDHURY INSTITUTE OF ENGINEERING AND TECHNOLOGY

Malda

FOREWARD

We forward the thesis entitled "ECONOMIC INTEGRATION OF RENEWABLE ENERGY SOURCES IN SMART GRID" submitted by MR. TANMOY PAUL (UNIVERSITY ROLL NO: 35501618001, REGISTRATION NO: 183550110035 OF 2018-2022)as a bona fide record of the project work carried out by him under the guidance and supervision of the undersigned in a partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Electrical Engineering from the institute under MAULANA ABUL KALAM AZAD UNIVERSITY OF TECHNOLOGY,WEST BENGAL.

Dr. Sandip Chanda

(Professor and Head of the Department) Electrical Engineering,

Ghani Khan Choudhury Institute of Engineering and Technology

Pin code- 732141

GHANI KHAN CHOUDHURY INSTITUTE OF ENGINEERING AND TECHNOLOGY

Malda

CERTIFICATE OF APPROVAL

We forward the thesis entitled "ECONOMIC INTEGRATION OF RENEWABLE ENERGY SOURCES IN SMART GRID" submitted by MR. TANMOY PAUL (UNIVERSITY ROLL NO: 35501618001, REGISTRATION NO: 183550110035 OF 2018-2022)as a bonafide record of the project work carried out by him under the guidance and supervision of Dr. Sandip Chanda, in a partial fulfillment of the requirements for the award of the degree of Master of Technology in Electrical Engineering from the institute under MAULANA ABUL KALAM AZAD UNIVERSITY OF TECHNOLOGY,WEST BENGAL.

BOARD OF EXAMINERS

_____ _____

_____ _____

_____

GHANI KHAN CHOUDHURY INSTITUTE OF ENGINEERING AND TECHNOLOGY

Malda

DECLARATION

I declare that this written submission represents my ideas in my own words and others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be a cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Tanmoy Paul	Pranta Debnath	Sajal Gayen	Subhendu Mondal
----------	----------	----------	----------
University Roll No. 35501618001	University Roll No. 35501618011	University Roll No. 35501618008	University Roll No. 35501619002

Acknowledgement

My deepest gratitude goes to Dr. Sandip Chanda of the department of Electrical Engineering of the Institution for his invaluable direction and supervision work, his wise advice, his patience, and for allowing me the opportunity to develop this work. From very first day of my project work, he has been kind and firm in showing me the right direction, being there for me in all my queries and quandary, and destining me to my objectives. Sir, without your constant encouragement and tremendous support I would have been “lost”. I highly appreciate the co-operation of my fellow researchers, all laboratory and non-teaching staff of this Department, for helping me directly or indirectly in framing this research in its form. Though it is beyond the scope of any acknowledgement what I have received from my family members in form of inspiration and encouragement, yet I make a humble effort to express my profound gratitude towards them.

June 2022 **Tanmoy Paul**

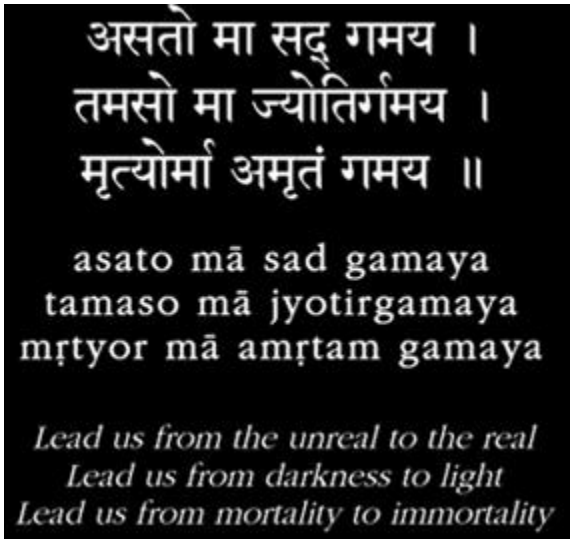

असतो मा सद् गमय ।
तमसो मा ज्योतिर्गमय ।
मृत्योर्मा अमृतं गमय ॥
asato mā sad gamaya
tamaso mā jyotirgamaya
mṛtyor mā amṛtam gamaya
Lead us from the unreal to the real
Lead us from darkness to light
Lead us from mortality to immortality

Table of contents

List of abbreviation

RE - Renewable Energy

PV - Photo Voltaic

MCP - Marketing Clearing Price

TL - Transmission Loss

RTO - Regional Transmission Organization

ATC - Available Transmission Capacity

ISO - Independent System Operator

OPF - Optimal Power Flow

PSO - Particle Swarm Optimization

IEEE - Institute of Electrical and Electronics Engineers

List of Figures

List of Tables

Chapter 1

INTRODUCTION

Smart Grid is a concept and vision that captures a range of advanced information, sensing, communications, control, and energy technologies. Taken together, these result in an electric power system that can intelligently integrate the actions of all connected users—from power generators to electricity consumers to those that both produce and consume electricity ("prosumers")—to efficiently deliver sustainable, economic, and secure electricity supplies. (Source: Definition adapted from the European Technology Platform Smart Grid [ETPSG]).

Researchers have been working around the world to examine and develop the nature of variable Renewable Energy grid integration challenges that have arisen. The major challenges faced during integration of variable RE grid are:

1. Technical Challenges: Ensuring power system reliability as uncertainty and variability increase.
2. Economic, Policy, and Regulatory Challenges: Effectively managing the cost of RE integration and the grid investments that support it, designing policies to harness maximum value from RE, and ensuring that appropriate incentives are in place to encourage appropriate grid investments.

Technical Challenges

Two dominant technical challenges can be identified with a higher penetration of RE generation:

1) managing variability and uncertainty during the continuous balancing of the system, and

2) balancing supply and demand during generation scarcity and surplus situations.

Managing variability and uncertainty during the continuous balancing of the system

Variable RE sources are both more uncertain and more variable than conventional generators. Wind farms provide a useful illustration of uncertainty: while the farm may reliably produce power for 40% of the hours in a year, it is not easy to predict far in advance when generation will occur.

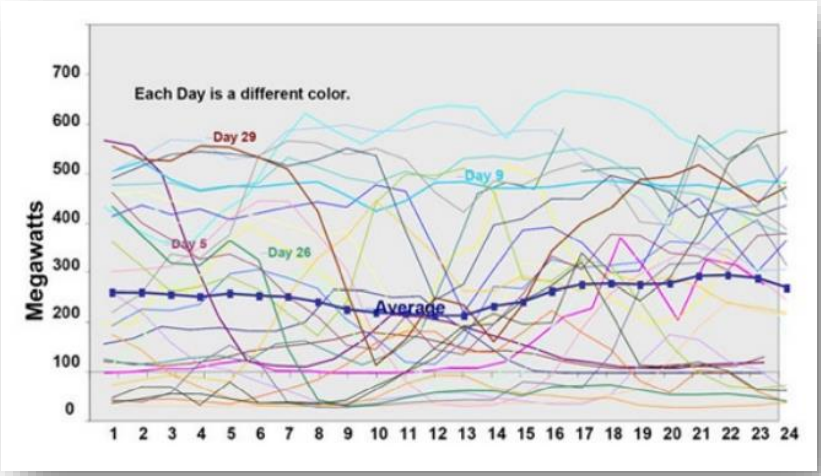


Figure (1.a) Tehachapi wind farm output, 30 successive days (Source: California ISO 2007)

Figure (1.a) illustrates hourly output from a single wind farm on 30 successive days. A rooftop solar installation provides a useful illustration of variability: transient events such as cloud passage can reduce output quite rapidly (see Figure 1.b).

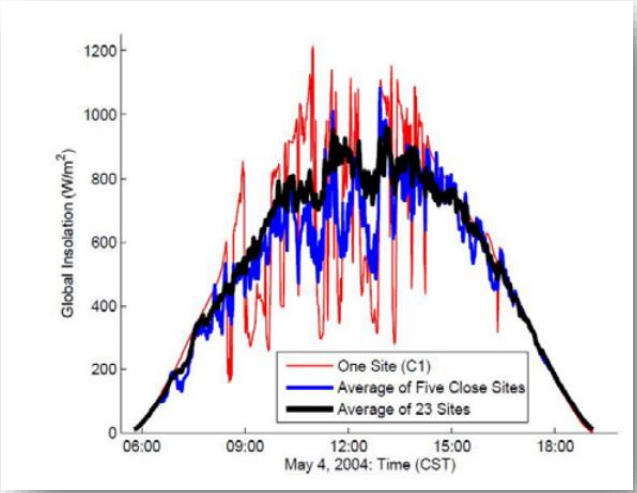


Figure (1.b). Example of 1-min global insolation on a partly cloudy day from one site, the average of five close sites, and the average of 23 sites (Source: Mills and Wiser 2010).

Figure (1.b) also illustrates how spreading solar photovoltaics (PV) over a larger geographic area tends to reduce aggregate variability. Generally, solar PV output is more variable than wind (changing faster on a minute-to-minute basis), but it is less uncertain. With both wind and solar power, the ability of system and generation operators to predict generation levels is improving (Bullis 2014).

Balancing supply and demand during generation scarcity and surplus situations

Related to but distinct from the previous challenge is system operators' need to balance supply and demand in situations of high RE production and low demand or low RE production and high demand. Supply of wind and solar may not coincide closely with demand, introducing challenges at the bulk power system level and—if there is significant distributed PV generation—at the local distribution system level. An illustration of the distribution system. challenge is 'reverse power flow' that can occur during the middle of the day from areas with large amounts of distributed PV. When most people are out of their homes at work, residential electricity demand is low, so more of the generated electricity feeds back up through the transformer to the medium-voltage network. Distribution systems have not traditionally needed to anticipate this situation because most generation has come from large-scale systems located on the transmission grid, with power flowing predictably one way down to lower voltage systems.

Nighttime wind provides an illustration of the supply-demand challenge on the bulk power system. When nighttime demand is very low and wind is strong (for example on a windy springtime night), there may be insufficient demand to utilize all of the wind power, and conventional generators may not be able or willing to reduce their generation further to make space for the wind power. In this case, insufficient power system flexibility could result in 'curtailed' wind power.

Challenges related to high peak load during periods of low variable RE production are less technically demanding and are primarily economic challenges related to market solutions chosen to remunerate reserve capacity or demand response activities.

Smart grid solutions emerging to manage continuous balancing of the system include:

• Better forecasting. Widespread instrumentation and advanced computer models allow system operators to better predict and manage RE variability and uncertainty.

• Smart inverters. Inverters and other power electronics can provide control to system operators, as well as to automatically provide some level of grid support.

• Demand response. Smart meters, coupled with intelligent appliances and even industrial-scale loads, can allow demand-side contributions to balancing.

• Integrated storage. Storage can help to smooth short-term variations in RE output, as well as to manage mismatches in supply and demand.

• *Real-time system awareness and management. Instrumentation and control equipment across transmission and distributions networks allows system operators to have real-time awareness of system conditions, and increasingly, the ability to actively manage grid behavior.*

Economic, Policy, and Regulatory Challenges

In addition to technical challenges, institutional challenges also arise with increasing shares of variable RE. Broadly these relate to the unique economics of variable RE, which give rise to various policy and regulatory issues. Two specific challenges are identified here: capital intensive grid upgrades, and uncertain project costs and cash flows.

Capital-intensive grid upgrades

Grid upgrades may be required to accommodate wind and solar power. For example, to the extent high quality wind and solar resources are located far from demand centers, new transmission lines or upgrades to existing lines may be required. At the distribution level, rooftop PV may accelerate the fatigue of distribution components, such as low-voltage transformers, moving forward the need for grid upgrades. Minimizing the cost of upgrades, while ensuring system reliability, translates to greater value from RE investments.

Uncertain RE project costs and cash flows

Smart grid solutions are emerging to two specific issues that historically have negatively impacted RE project economics: grid upgrade costs allocated to RE project developers, and energy curtailment when full RE production cannot be readily integrated into the power system. Both issues may cause cash flows of the project to diverge further from expectation. To the extent upgrades become costly or curtailments increase, the investment landscape for variable RE becomes more uncertain and can slow overall deployment. In cases where policy measures and subsidies insulate project investors from these risks—for example to further enhance the investment environment for RE—costs and risks may be socialized. Smart grid investments can also play an important role in reducing those costs and risks. Cost-effective methods of reducing curtailment and minimizing new transmission or grid upgrades can therefore capture more value from RE sources, improve the viability of individual RE projects, and maximize value to the system, enhancing the overall investment climate.

Smart grid solutions emerging to address the economic, policy, and regulatory challenges of variable RE include:

• Dynamic line rating. Real-time information about transmission line capacity can allow grid operators to extract more value from existing lines, reducing the need for costly upgrades.

• Demand response. Enabled by smart meters and intelligent loads, customer demand response solutions can help absorb excess RE generation, reducing the need for distribution upgrades.

• Smart inverters and other advanced power controls. Smart inverters and other power controls can reduce the need for significant grid transmission and distribution upgrades, thus reducing costs that may otherwise be levied on RE projects or socialized.

• Grid-scale storage. Large-scale storage of various types can help to reduce the need for additional transmission capacity.

• Behind-the-meter storage. Customer storage solutions can help absorb excess PV generation, reducing the need for distribution upgrades.

• Advanced energy management systems. Advanced energy management systems that provide real-time, high-resolution visibility and control of power systems, can allow grid operators to defer more costly capital expenditures.

• Better forecasting. System-level forecasting can help system operators operate their grids more flexibly, allowing more production to be accepted.

1.1 Literature review

This paper proposes an optimization model to maximize social welfare by standardizing the operating conditions with an overall improvement of dynamic stability of power markets endowed with Smart Grid communication technology. For optimum utilization of smart metering facility, the model effectively involves resources like demand response, generation surplus and an efficient methodology to optimize the Market Clearing Price (MCP) as well as profit of the market participants by effective categorization. The power market dynamic price equilibrium has been estimated by forming Jacobian of the sensitivity matrix to regulate the state variables for the standardization of the quality of solution. A novel load curtailment strategy has also been proposed to amalgam stability restoring shedding with profit retentive load cut. The model has been tested in IEEE 30 bus system and IEEE 118 bus System in comparison with standard curtailment based optimization technique to produce encouraging results.

In [1] market structures and functioning of smart grid is carried out. From load and demand response graph the generation cost optimization formula: $\sum Ci = a(i)p^2+b(i)p + c(i)$ is used. The cost coefficients a, b, and c are calculated with the help of data collected from different renewable energy power systems. The data is implemented in cost optimization formula to get the best results for cost coefficients a, b, and c.

In [2] PSO load flow model using OPF2 function in IEEE 118 buses and IEEE 30 buses System cost optimization is carried out for the particular values of cost coefficients a, b, and c. The Total cost of generation(F1) is noted for a healthy system, single line contingency, double line contingency and triple line contingency.

In [3] PSO load flow model using OPF2 function in IEEE 118 buses and IEEE 30 buses System loss optimization is carried out for the particular values of cost coefficients a, b, and c. The Total generation and transmission loss (TL) is noted for a healthy system, single line contingency, double line contingency and triple line contingency.

In [4] PSO load flow model using OPF2 function in IEEE 118 buses and IEEE 30 buses System penalty based optimization is carried out for the particular values of cost coefficients a, b, and c. Penalty based method is used to estimate the penalty in both the systems. The best values for total cost of generation(F1) and total transmission and generation loss (TL) is extracted using the line convergence method.

In [5] The convergence line graph is plotted and all the data is presented with the data values using the penalty based optimization technique to obtain the best result.

1.2 Motivation behind the work

From the above literature survey it is quite clear that research worldwide has been going on in the area of economic integration of smart grid using the demand response scheme the proper modeling of Renewable energy power system is of utmost importance and power engineers have been working on developing new scheme with every new abnormality faced by the power system. Absence of proper and standardize scheme for assuring reliability of operation of smart grid even in case of cost optimization, has been the underlining motivation behind this project work.

1.3 Thesis Organization

Chapter 1: Introduces the objective of the project, literature review and a brief description about different types of renewable energy sources for integration of Smart grid.

Chapter 2: This chapter starts with the market structure and functioning in Smart Grid.

Chapter 3: Says about the cost optimization function. The cost optimization function is used to extract the cost coefficients a, b, and c.

Chapter 4: Determines the cost optimization results using IEEE 30 buses system and IEEE 118 buses system for healthy system, single line contingency, double line contingency and triple line contingency.

Chapter 5: Determines the loss optimization results using IEEE 30 buses system and IEEE 118 buses system for healthy system, single line contingency, double line contingency and triple line contingency.

Chapter 6: Determines the penalty based optimization results using IEEE 30 buses system and IEEE 118 buses system for healthy system, single line contingency, double line contingency and triple line contingency.

Chapter 7: The convergence graph for penalty based method is plotted with generation and transmission losses in x axis and generation cost in y axis. The convergence formula is calculated from the convergence graph.

Chapter 2

MARKET STRUCTURE AND FUNCTIONING IN SMART GRID

The Smart Grid is an eco-friendly optimization of the present grid endeavored to achieve operational excellence with high degree of reliability. The functional architecture proposed and implemented are based on some basic modification of the present grid organization for proper management of distributed generation with renewable energy sources, improvement of sustainability with self-healing activities like congestion, power quality management and encouragement of price responsive demand reduction. The profusion of these activities employs extensive bidirectional communication between wholesale markets/transmission operation and retail markets/distribution operations. Fig. 1 depicts one such architecture enabling the system operator to not only utilize the generator information, but is also to make itself capable of incorporating the demand response of consumers aggregated by local entities like Regional Transmission Organization (RTO), historical and forecasted data, Available Transmission Capacity (ATC) margins etc. Efficient employment with these new resources as wholesale market product, Independent System Operator (ISO) will be able to reach the furthest corners of

the network from generation to load end to maintain profound operating conditions under the worst possible states of the system. In this context ISO will be able to identify the state variable creating imbalance in the power market to de-standardize its operations. The price responsiveness of the state variables of modern power markets has been elaborated in the following section.

The proposed price responsive OPF model

Price sensitivity of demand

The price responsiveness of demand has become an incredible input to the OPF algorithms for their load peak shaving capability in high price conditions. Effective deployment of this resource may lead to the solutions of modern day power network tribulations like network congestion, voltage instability and perturbations in dynamics in power market. The demand elasticity of price is, hence as depicted in an important parameter to be considered for Optimal Power Flow. With the assistance of smart metering this product can be incorporated in optimization to achieve distinction in operating conditions and welfare of the market participants. As shown in Fig. 2, a

demand with a marginal benefit above the marginal price will lead to an expansion in consumption until the equilibrium is reached. In Fig. 2a A–B–C represents the bid curve at a particular hour, while X–Y its tangent. The price responsive demand curve (Fig. 2b) shows the nature of the consumer towards price volatility corresponding to the bids. From the bid curve the willingness to pay of the consumers can be determined as

willingness to pay = tan Θ

$$\tan\theta = \frac{f(d_1) - f(d_2)}{d_1 - d_2} = \frac{b_1 - b_2}{d_1 - d_2}$$

where d1, d2...etc. are the power demands and the bid curve is expressed as a function of demand such as f(di). The Market Clearing Price (MCP) corresponding to each point of the bid curve have been plotted in Fig. 2a and b. In this figure it has been assumed that b1 = d1. MCP1, b2 = d2 . MCP2, b3 = d3 . MCP3. Let us assume that the two curves are fitted with two different polynomials. k1x^2 + l1x + m1 represent the bid curve while k2x^2 + l2x + m2 represent the price responsive curve where k1, l1, m1 are the coefficients of bid curve and k2, l2, m2 are the coefficients of price responsive curve. Now mapping willingness to pay into price responsiveness of demand

$$\begin{aligned}\tan\theta &= \frac{b_2 - b_1}{d_2 - d_1} = \frac{d_2 \cdot MCP_2 - d_1 \cdot MCP_1}{d_2 - d_1} \\ &= \frac{d_2(k_2 d_2^2 + l_2 d_2 + m_2) - d_1\left(k_2 d_1^2 + l_2 d_2 + m_2\right)}{d_2 - d_1} \\ &= k_2\left(d_2^2 + d_2 \cdot d_1 + d_1^2\right) + l_2(d_2 + d_1) + m_2\end{aligned}$$

The willingness to pay is as sensitive to bid curve as is to price responsive curve. Hence, inclusion of demand response or price responsive characteristics into OPF not only incorporates the price dependent consumption characteristics but also involves the willingness to pay of the consumers for a particular alteration in price. In the present work load demand is also scheduled like generation, the load curtailment becomes willingness to pay dependent and involves the consumer more into OPF. In every hour (or a specified period) the consumer with the assistance of smart metering, will be able to modify his stand in the power market enabling Independent System Operator (ISO) to regulate curtailment depending on ‘‘willingness to pay’’ of the consumers.

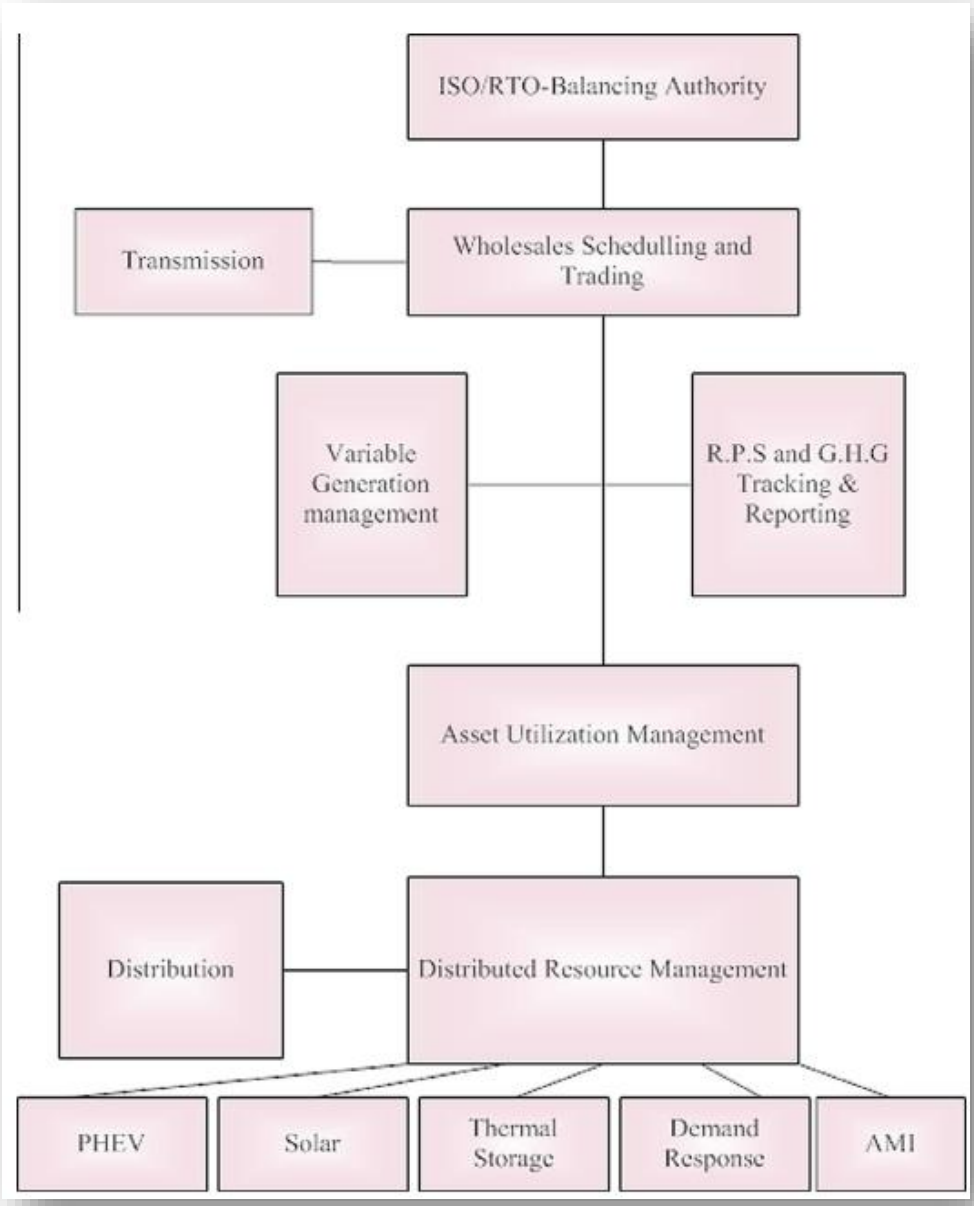


Fig: (2.a) Interaction of power market entities of Smart Grid. Note: Conceptualized architecture of modern power grid or Smart Grid to ensure optimum utilization of generation, transmission and distribution resources.

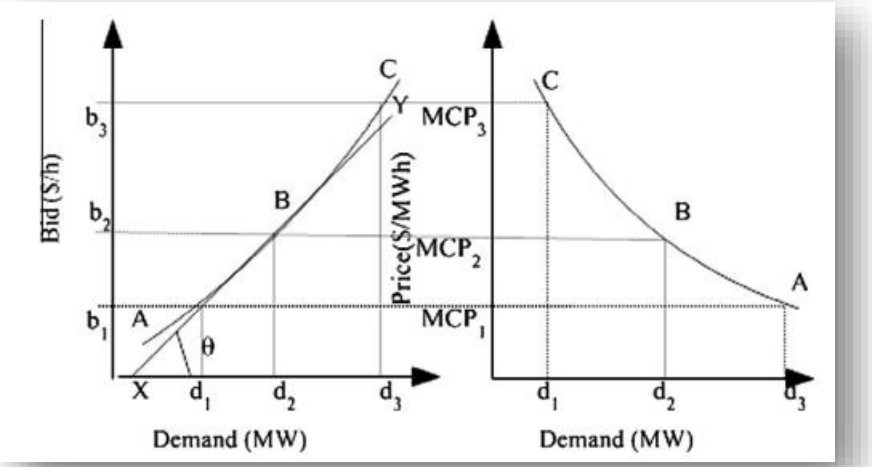


Fig: (2.b) - A and B: Mapping the price responsiveness of demand from Bid curve. Note: Incorporation of this price responsive demand curve into optimal power flow ensures load curtailment of consumers in accordance with their ‘‘willingness to pay’’.

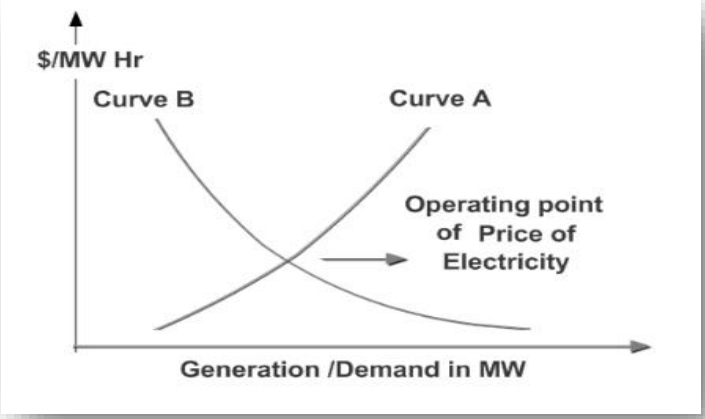


Fig: (2.c) generation/ demand curves A and B cuts at a particular point which is known as the operating point of Price of Electricity.

Single Line Diagram for IEEE 30 bus System

The feasibility and effectiveness of the proposed algorithm has been demonstrated in the modified IEEE 30 bus system shown . The summary of relevant data from the modified IEEE 30 bus system is presented in tables. All simulations have been carried out in MATLAB.

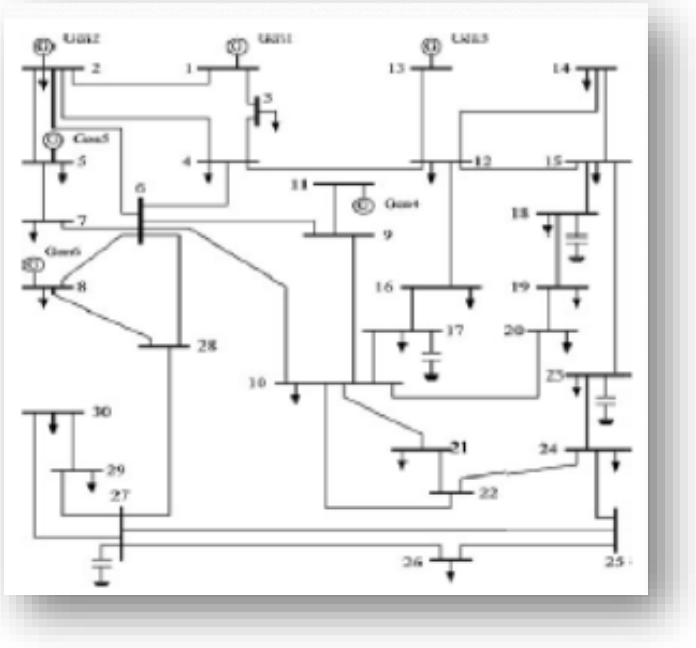


Fig:(2.d) Single Line Diagram of IEEE 30 bus System

Table (2.a): **Description of the IEEE 30 Buses Test System**

1. Number of buses:	30
2. Number of lines:	40
3. Number of generator buses:	6
4. Number of load buses:	24
5. Total demand in MW:	283.4 MW
6. Maximum line resistance:	0.3202 ohm
7. Maximum line reactance:	0.6027 ohm
8. Maximum active power load in a bus:	94.2 MW
9. Maximum reactive power load in a bus:	30 MVAR

Single Line Diagram for IEEE 118 bus System

The feasibility and effectiveness of the proposed algorithm has been demonstrated in the modified IEEE 118 bus system shown . The summary of relevant data from the modified IEEE 118 bus system is presented in tables. All simulations have been carried out in MATLAB.

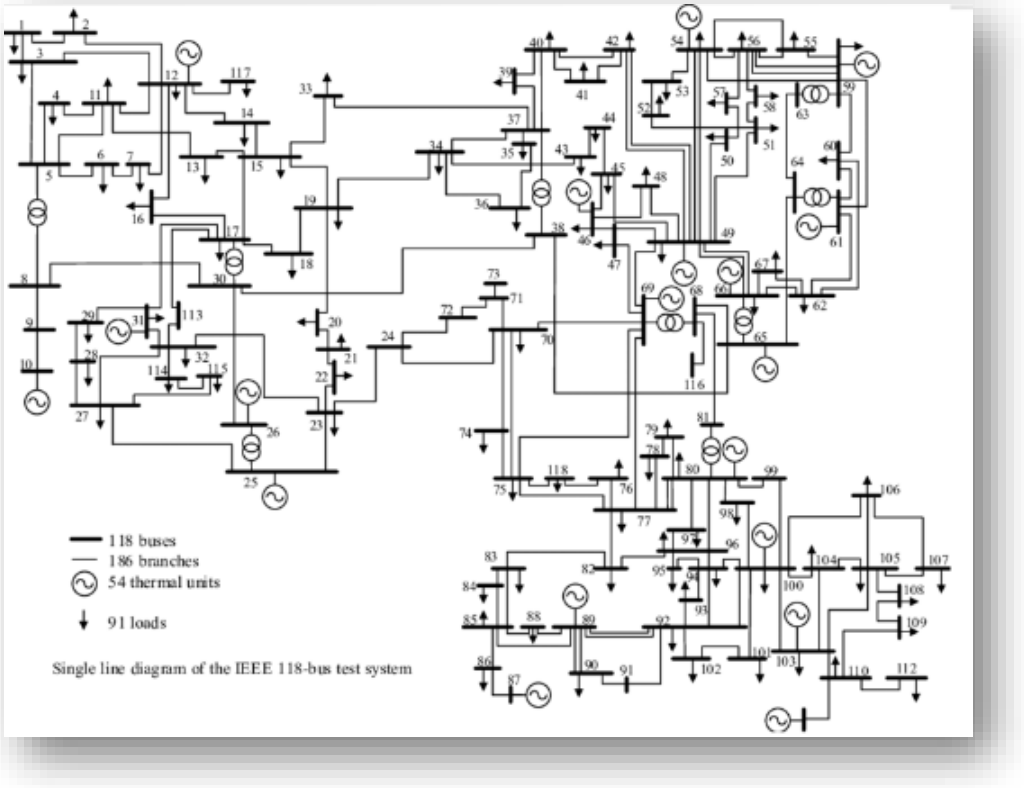


Fig:(2.e) Single Line Diagram of IEEE 118 bus System

Table (2.b): **Description of the 118 Buses Test System**

1. Number of buses:	118
2. Number of lines:	185
3. Number of generator buses:	53
4. Number of load buses:	65
5. Total demand in MW:	3668.0 MW
6. Maximum line resistance:	0.0985 ohm
7. Maximum line reactance:	0.41150 ohm
8. Maximum active power load in a bus:	277.0 MW
9. Maximum reactive power load in a bus:	113.0 MVAR

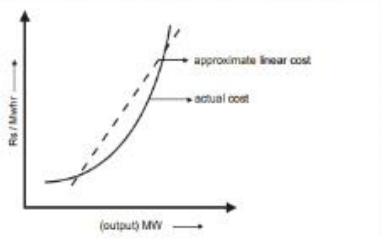
Chapter 3

OPTIMAL POWER FLOW

Objective function in conventional cost optimization :

Minimize $F = \sum_{n=1}^{N_G} C_i$ \$/hr

Where $C_i = AP_{gi}^2 + BP_{gi} + C$



In the present work, the objective function is suitably modified to incorporate the proposed voltage, line loss and congestion penalties. The modified multi-objective OPF can be described as:

Minimize $F = \sum_{n=1}^{N_G} C_T + p1.V_{\min} + p2.P_{l\max} + p3.P_{ij\max}$ \$/hr

Constraints of Optimization

Equality or power balance constraints:

$$P_{Gi} - P_{Di} - V_i \sum_{i=1}^{n} V_j (G_{ij}\cos\theta_{ij} + B_{ij}\sin\theta_{ij}) = 0 \qquad Q_{Gi} - Q_{Di} - V_i \sum_{i=1}^{n} V_j (G_{ij}\cos\theta_{ij} - B_{ij}\sin\theta_{ij}) = 0$$

Inequality or generator output constraints:

$$P_{gi}^{\min} \le P_{gi} \le P_{gi}^{\max} \qquad Q_{gi}^{\min} \le Q_{gi} \le Q_{gi}^{\max}$$

Voltage constraint:

$$V_i^{\min} \le V_i \le V_i^{\max}$$

Transmission constraint:

$$P_{ij\min} \le P_{ij} \le P_{ij\max}$$

By using the cost optimization function :

$\sum Ci = a(i)p^2 + b(i)p + c(i)$

The cost coefficients a, b, and c are calculated for three renewable energy based power systems.

Renewable Energy Power Sources

1. Solar Park by NHPC in Odisha – Power generation 100 MW.

The total cost of generation is converted into US dollars($).

For 10 MW power generation using the cost optimization function:

1100$= (10 × 10)a + 10×b + c ….(i)

For 50 MW power generation using the cost optimization function:

4100$= (50×50)a + 50×b + c…..(ii)

For 100 MW power generation using the cost optimization function:

12000$= (100×100)a + 100×b + c…..(iii)

From the equations (i), (ii), and (iii) we get the values of a, b, c as

a = 0.422

b = 74.66

c = 311.11

2. Loktak Power Station under NHPC in Manipur - Power generation 100-105 MW.

The total cost of generation is converted into US dollars($).

For 10 MW power generation using the cost optimization function:

900$= (10×10)a + 10×b + c ….(i)

For 50 MW power generation using the cost optimization function:

4000$= (50×50)a + 50×b + c…..(ii)

For 100 MW power generation using the cost optimization function:

8000$= (100×100)a + 100×b + c…..(iii)

From the equations (i), (ii), and (iii) we get the values of a, b, c as

a = 0.027

b = 75.83

c = 138.88

Waste Management Source

1. Tata Power Waste Gas Plant in Odisha – Power Generation 67.5 MW

The total cost of generation is converted into US dollars($).

For 10 MW power generation using the cost optimization function:

1000$= (10×10)a + 10×b + c ….(i)

For 50 MW power generation using the cost optimization function:

4000$= (50×50)a + 50×b + c…..(ii)

For 100 MW power generation using the cost optimization function:

9000$= (100×100)a + 100×b + c…..(iii)

From the equations (i), (ii), and (iii) we get the values of a, b, c as

a = 0.277

b = 58.33

c = 388.88

The cost coefficients for these three systems is used in IEEE 118 and IEEE 30 buses system to get the best results for cost optimization, loss optimization, and penalty based optimization.

Chapter 4

PARTICLES SWARM OPTIMIZATION

Particle Swarm Optimization (PSO) is an optimization approach based on social behavior of animals such that the movement of organisms in a bird flock or a fish school, which is initially proposed in Eberhart and Kennedy. The PSO has been largely applied to solve optimization problems. Its main idea is to optimize a problem by iteratively moving a group of particles towards the best position in a given search space. Inspired by the social behavior of animals, five main principles of the

PSO algorithm are outlined in the works of van den Bergh and Wang et al.:

• Proximity: it is able to perform simple calculations in time and space.

• Stability: the swarm does not change the behavior regarding every environment change.

• Quality: it is able to detect the quality change in the environment and respond to it.

• Diverse response: it has no limitation in the response to environment change.

• Adaptability: it is able to know if the change is worthy.

There are two main operators in the PSO structure: position update and velocity update. Figure 4a illustrates the basic algorithm of PSO which consists of four mains steps at each iteration of the process:

(1) For each particle of the population, the best position that the particle has reached thus far, called pBest, is evaluated. If the current position is better than the previous position, then the particle position is updated; otherwise, the previous position kept.

(2) Evaluate gBest, which is the best position of the particles in the entire population.

(3) Update the velocity using pBest and gBest. The new velocity is computed by:

$$v_i^{t+1} = v_i^t + \alpha\varepsilon_1\left[pBest_i^t - x_i^t\right] + \beta\varepsilon_2\left[gBest^t - x_i^t\right]$$

where i is particle index, t is time index, ε1 and ε2 are two random vectors in range [0, 1] and α and β are positive constants.

(4) Update the velocity of the particle. The new position of particles is calculated by:

$$x_i^{t+1} = x_i^t + v_i^{t+1}$$

These four steps are repeated to satisfy a stopping criterion, which means that the particles in the population are in the best-desired positions.

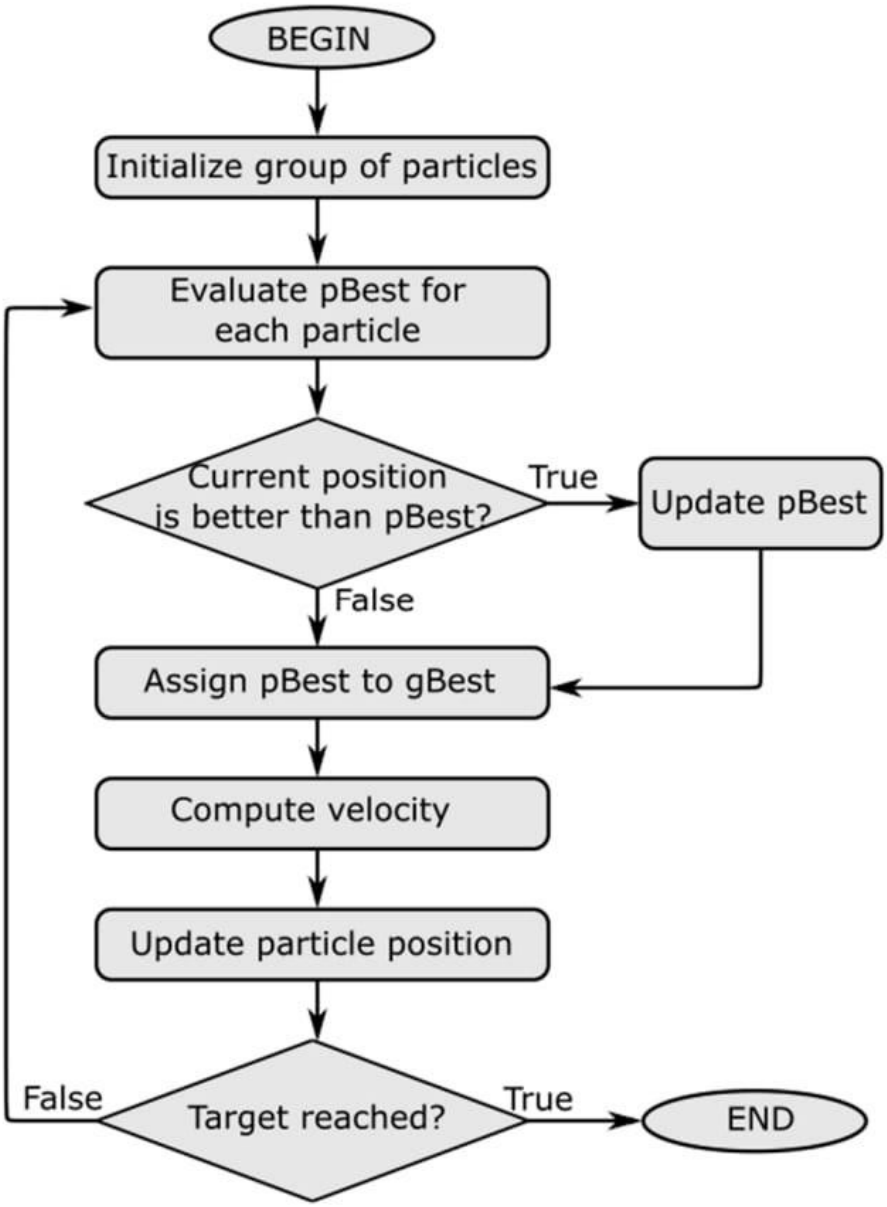


Figure 4a. The Particle Swarm Optimization (PSO) algorithm.

Chapter 5

COST OPTIMIZATION

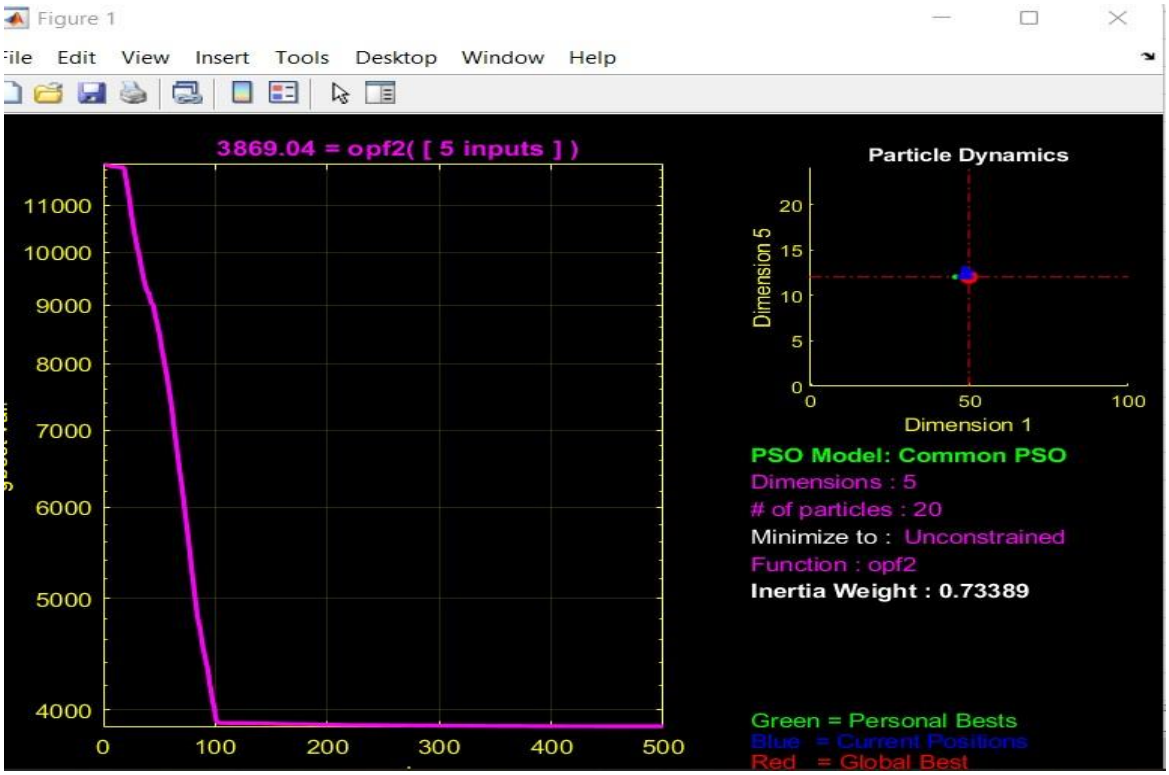
Cost Optimization of IEEE30BUS System

For Healthy System

Table no.(5.a)

	P1	P2	P3	P4	P5	P6	Vmin	Vmax	TL	Total Cost (F1)
Cost Optimization	189.2763	49.72952	22.82808	10	10	12	0.995647	1.06116	10.4339	3.8690x10^3

Output Figure. (5.a)



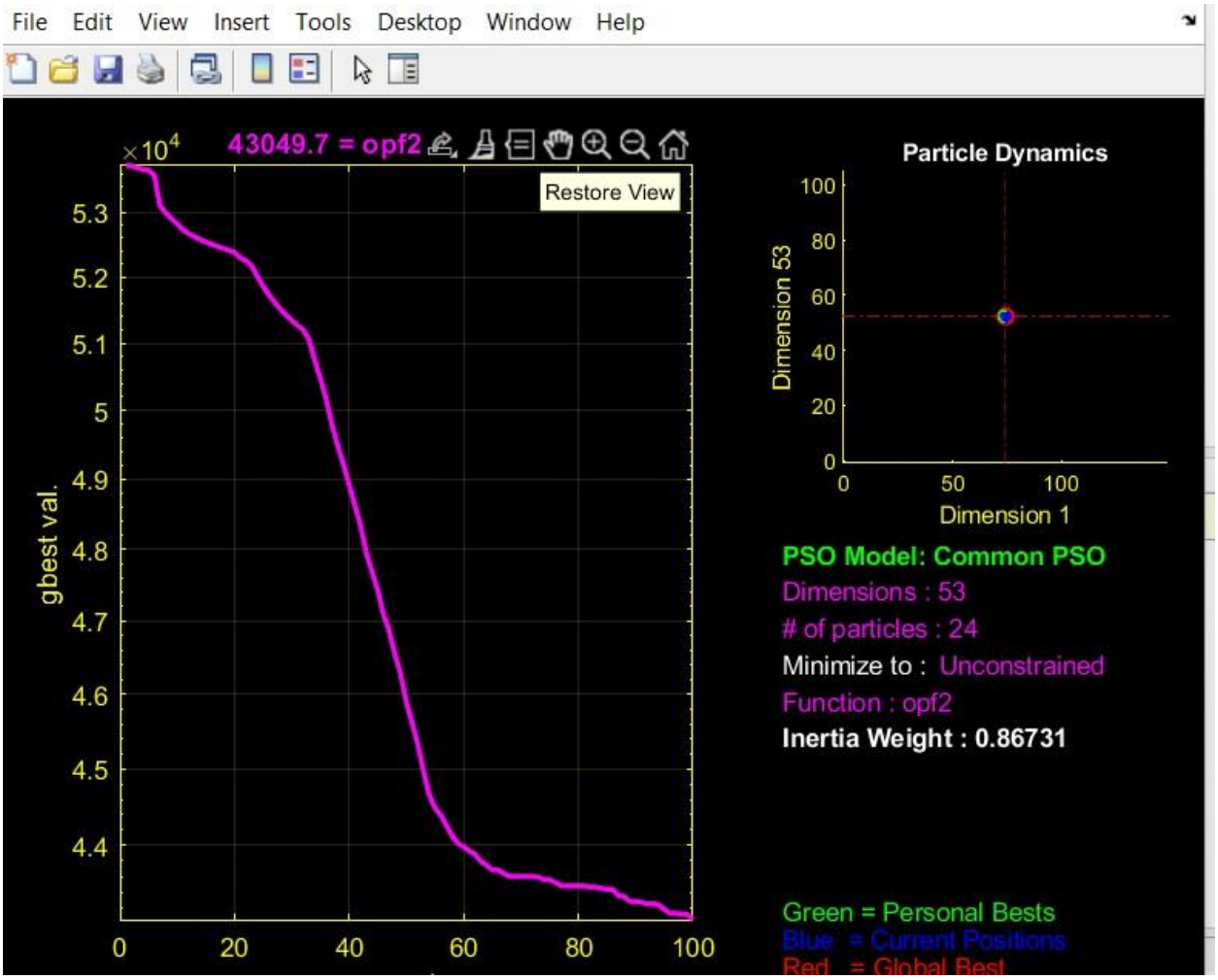
Cost Optimization of IEEE118BUS System

For Healthy System

Table No.(5.b)

	Vmin	Vmax	TL	Total Cost (F1)
Cost Optimization	0.9568	1.1087	2.4256x 10^3	43049.6831

Output Figure. .(5.b)



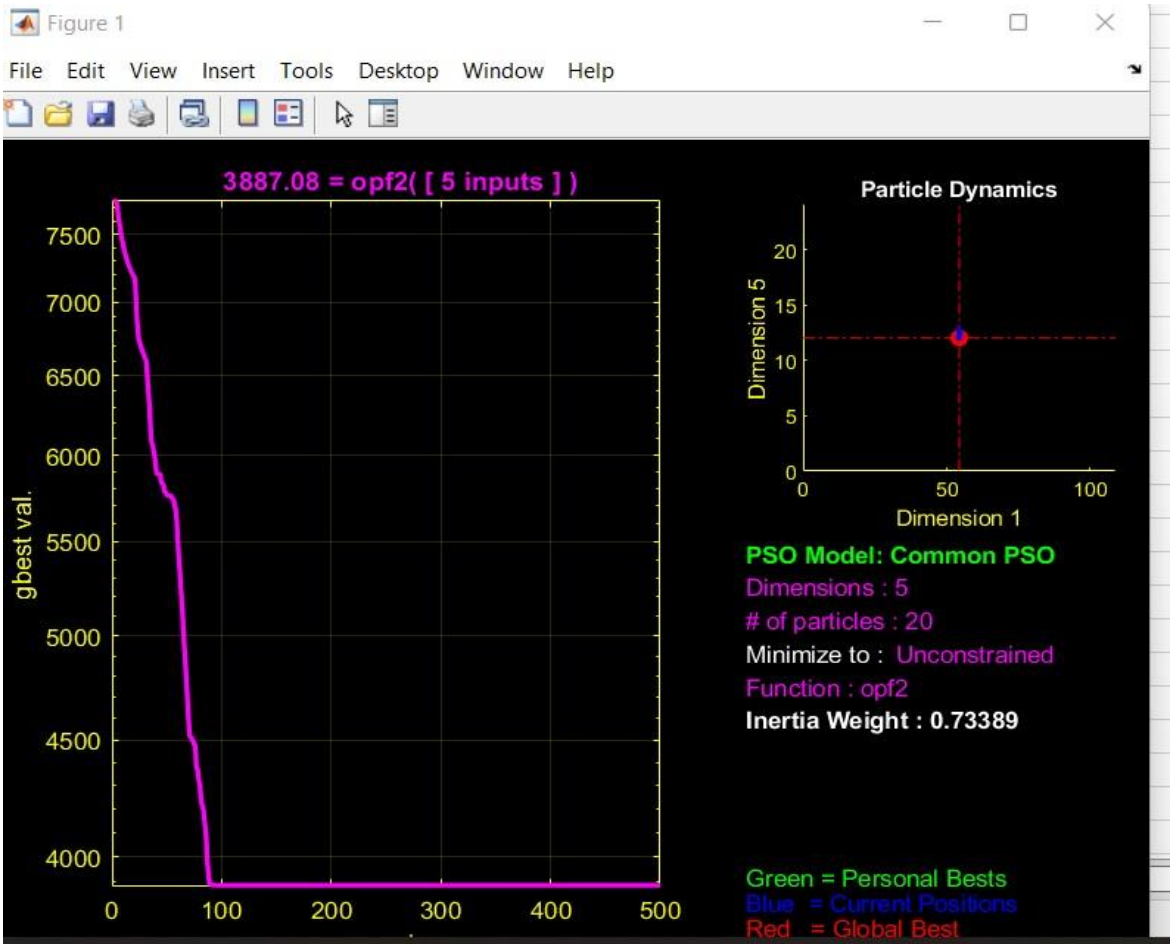
Cost Optimization of IEEE30BUS System

Single line contingency of ieee30bus System (n-1),1-3

Table no. .(5.c)

	P1	P2	P3	P4	P5	P6	Vmin	Vmax	TL	Total Cost (F1)
Cost Optimization	188.8037	54.36674	23.62134	10	10	12	0.992773	1.06	15.3917	3.8871x10^3

Output Figure. .(5.c)



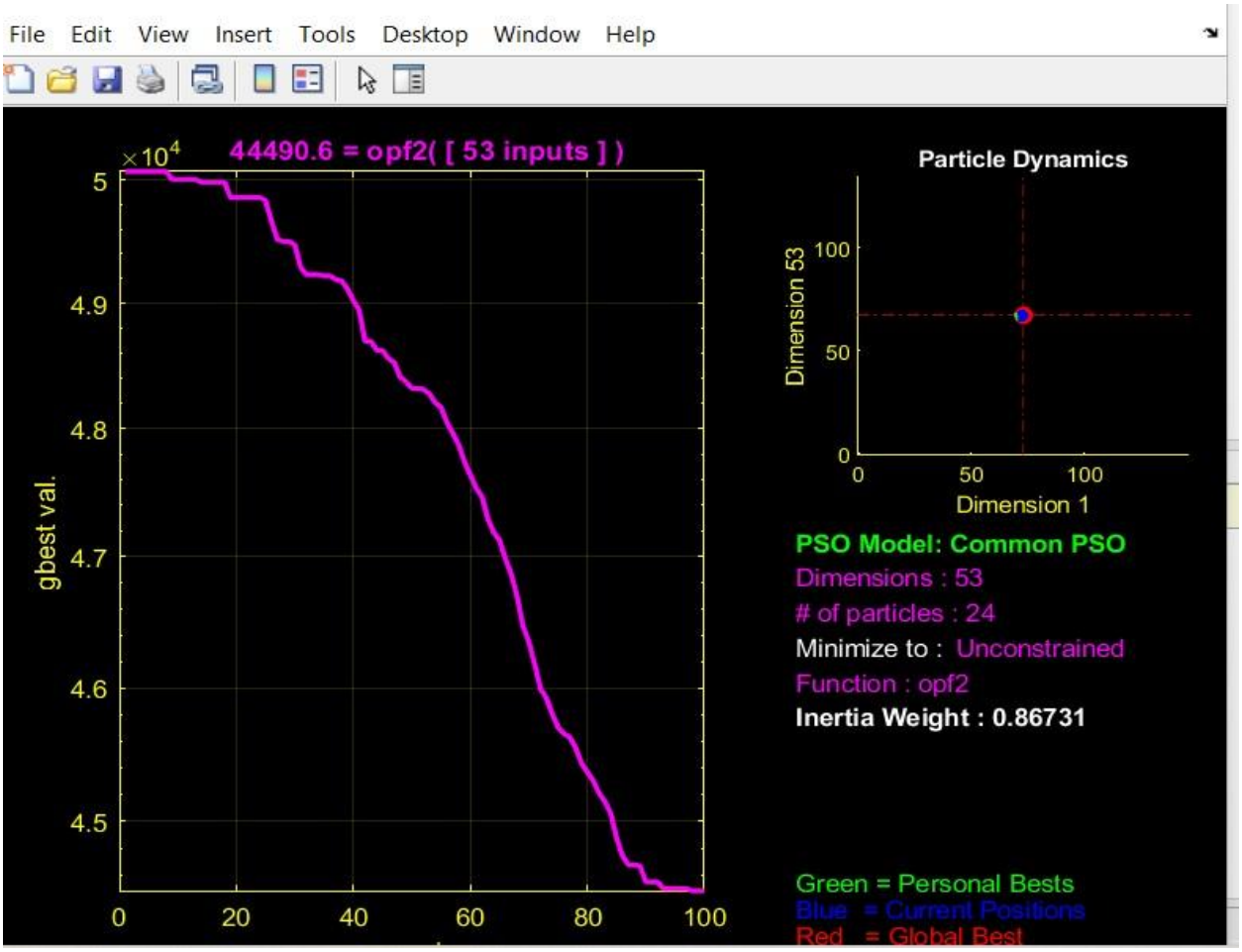
Cost Optimization of IEEE118BUS System

Single line contingency of ieee118bus System (n-1),1-3

Table no. .(5.d)

	Vmin	Vmax	TL	Total Cost (F1)
Cost Optimization	0.9577	1.1086	2.5762 x 10^3	44490.5574

Output Figure. .(5.d)



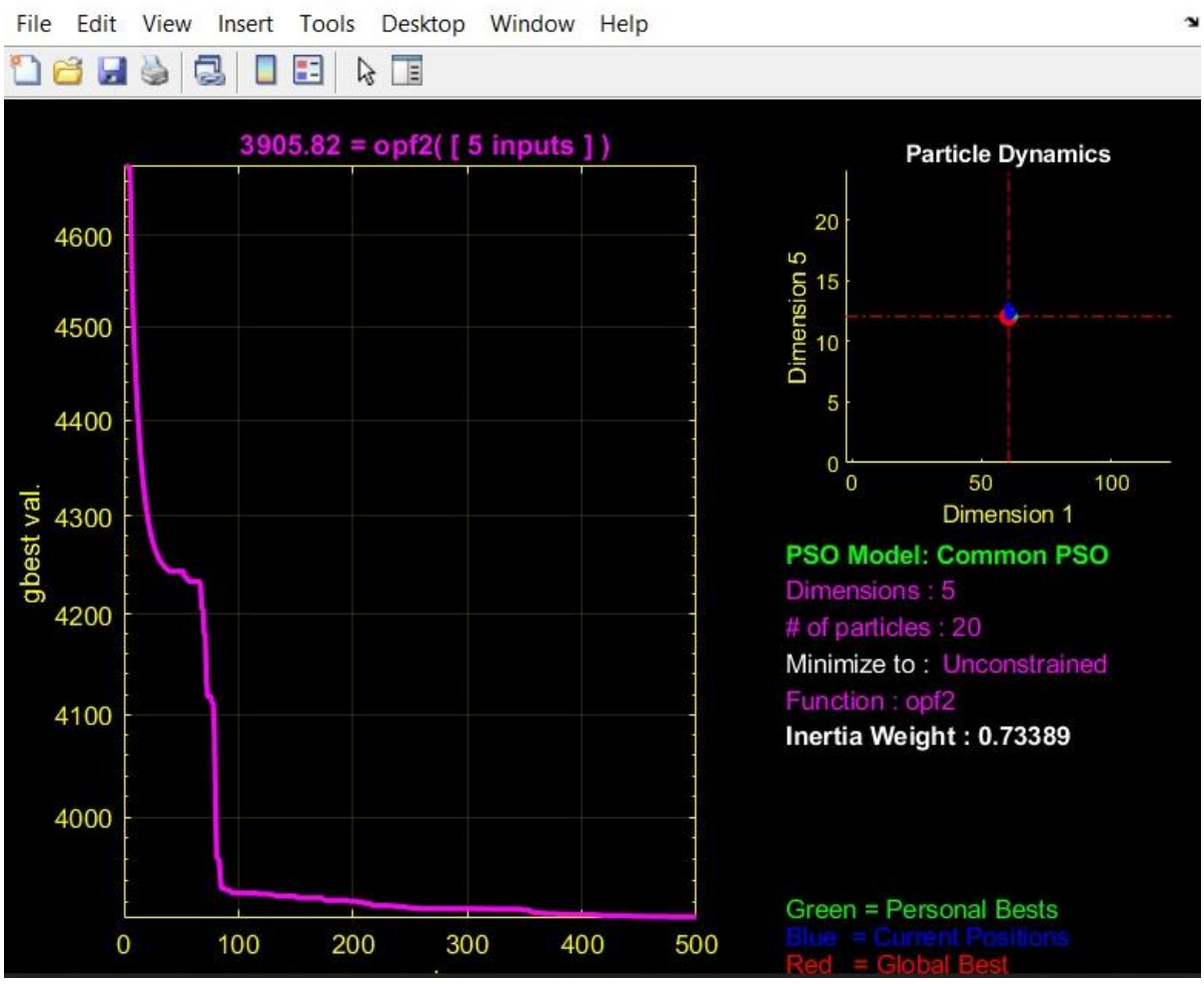
Cost Optimization of IEEE30BUS System

Double line contingency of ieee30bus System (n-2),1-3,2-6

Table no. .(5.e)

	P1	P2	P3	P4	P5	P6	Vmin	Vmax	TL	Total Cost (F1)
Cost Optimization	186.2172	60.2651	24.94597	10	10	12	0.988343	1.06	20.0283	3.9058x10^3

Output Figure. .(5.e)



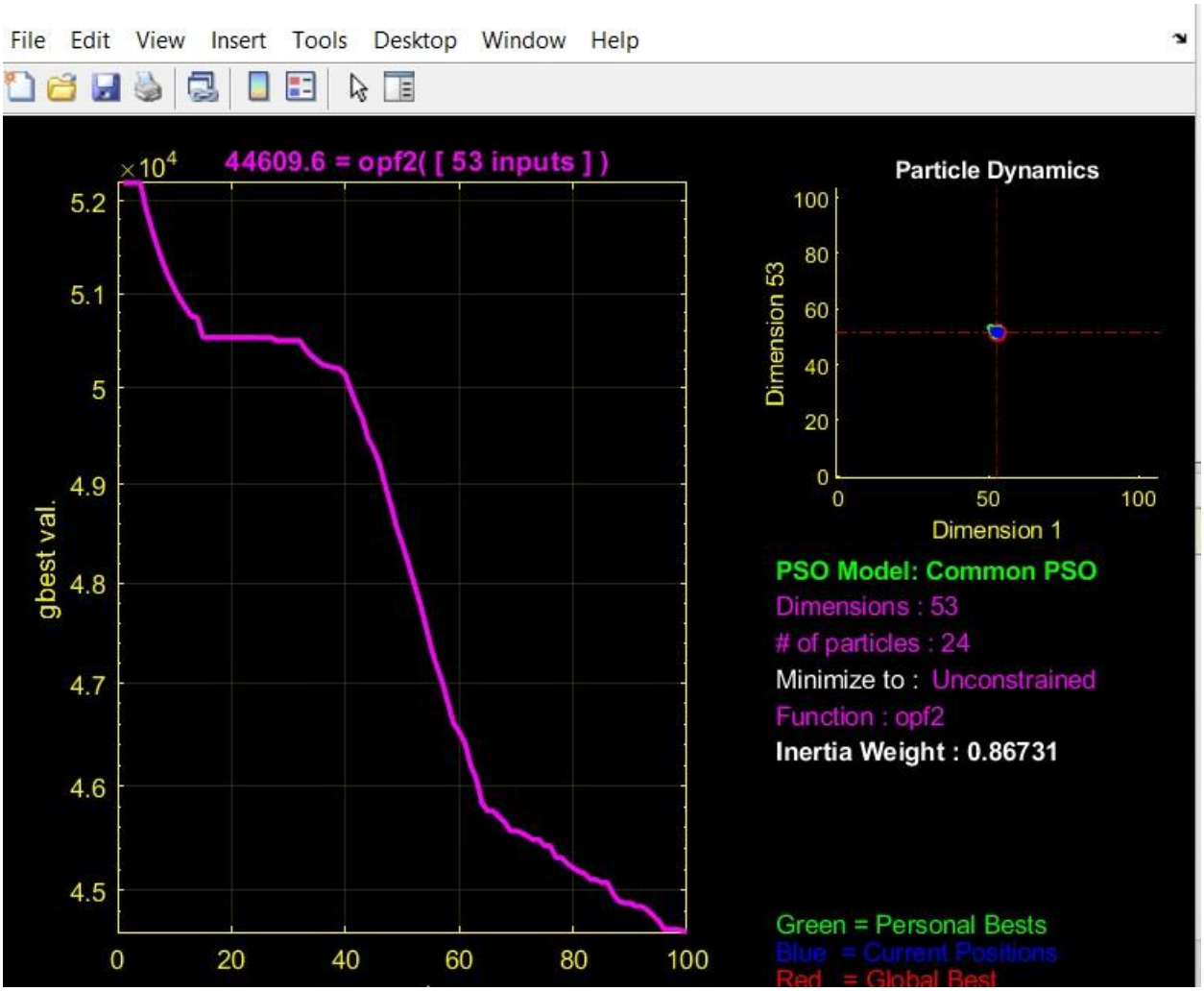
Cost Optimization of IEEE118BUS System

Double line contingency of ieee118bus System (n-2),1-3,5-6

Table no. (5.f)

	Vmin	Vmax	TL	Total Cost (F1)
Cost Optimization	0.9577	1.1086	2386.5	44909.6

Output Figure.(5.f)



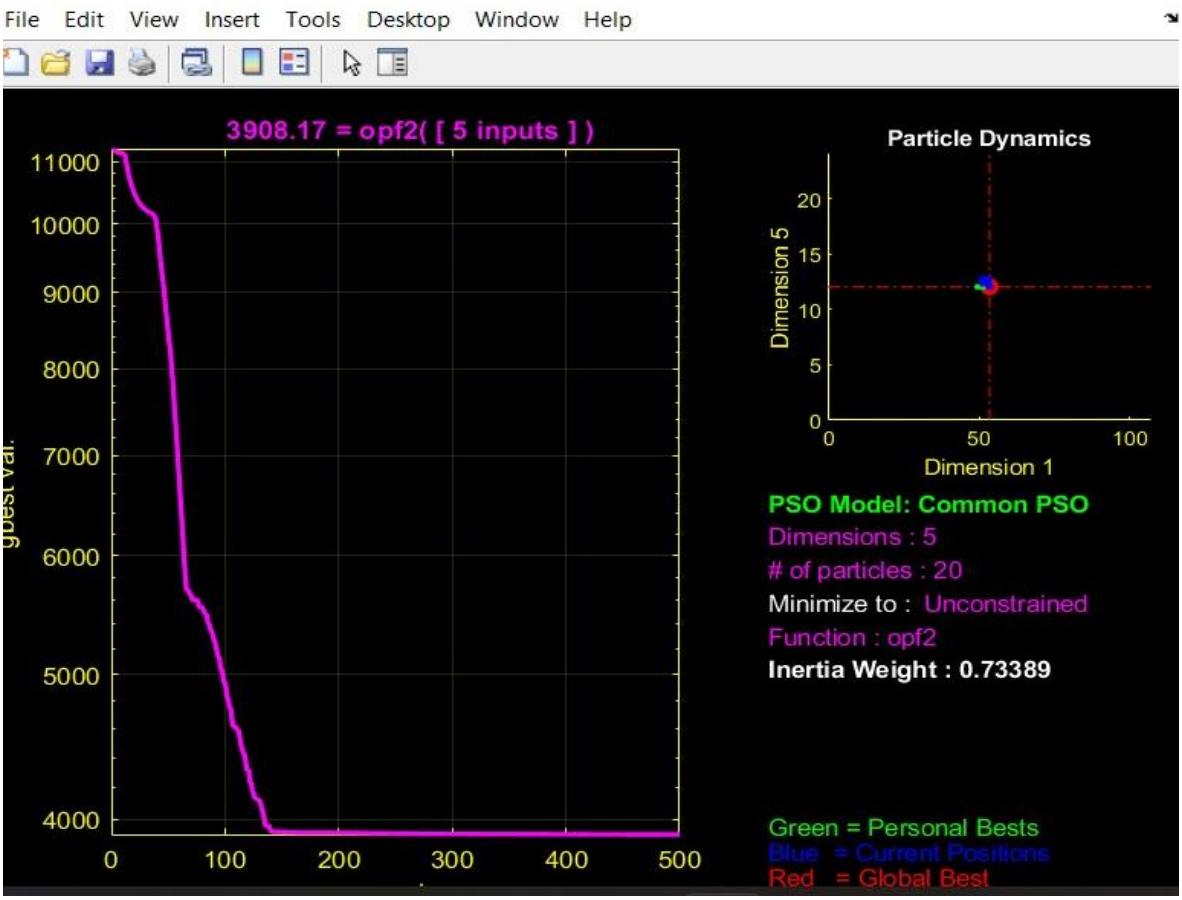
Cost Optimization of IEEE30BUS System

Triple line contingency of ieee30bus System (n-3),1-3,2-6,6-8

Table no. (5.g)

	P1	P2	P3	P4	P5	P6	Vmin	Vmax	TL	Total Cost (F1)
Cost Optimization	194.3319	53.37169	25.03352	10	10	12	0.9899	1.082	21.3371	3.9082x10^3

Output Figure.(5.g)



Chapter 6

LOSS OPTIMIZATION

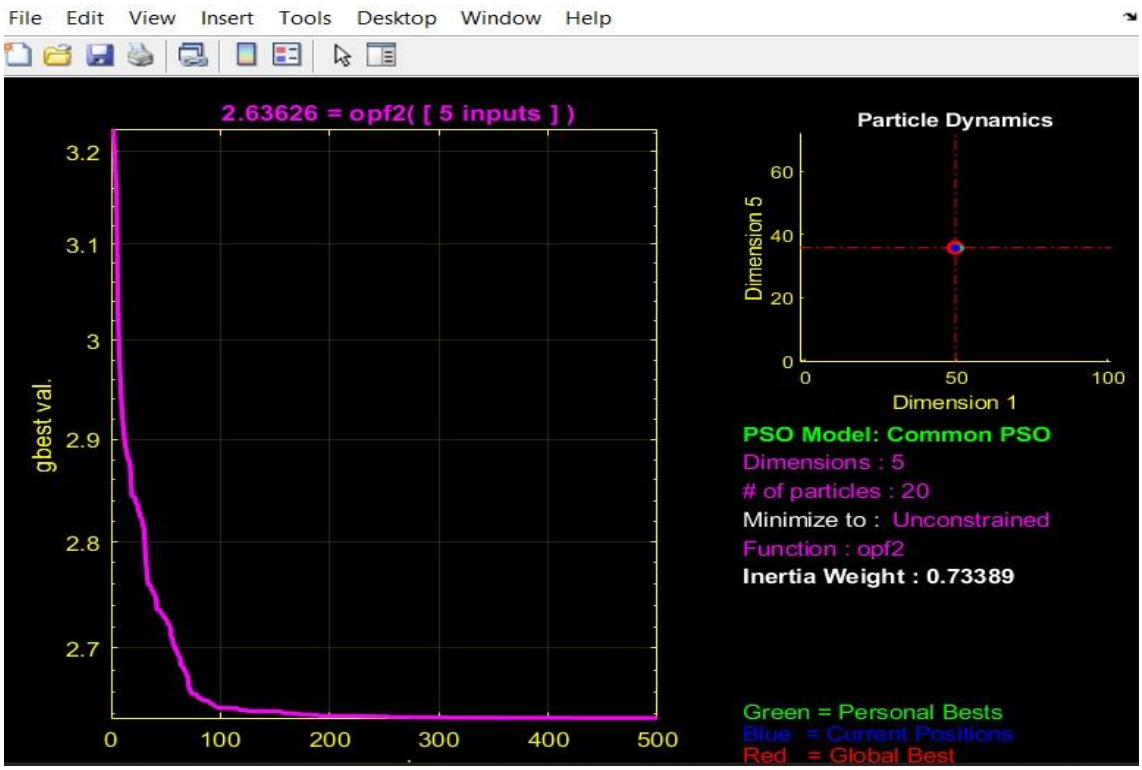
Loss Optimization of IEEE30BUS System

For Healthy System

Table no. (6.a)

	P1	P2	P3	P4	P5	P6	Vmin	Vmax	TL	Total Cost (F1)
Loss Optimization	14.98052	49.67285	50	65.8464917	69.73145429	35.80495	0.9929	1.082	2.6722	2.6363

Output Figure.(6.a)



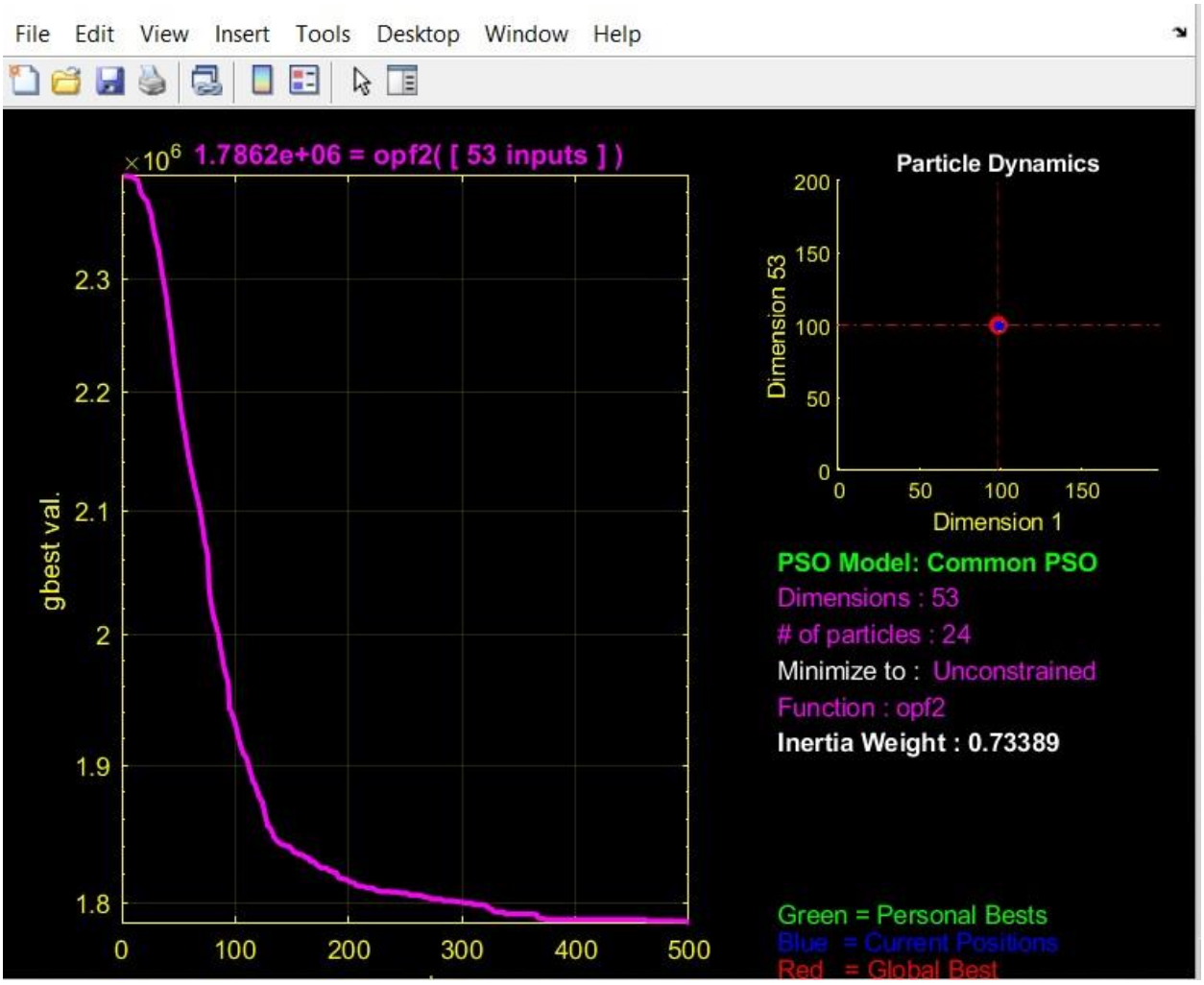
Loss Optimization of IEEE118BUS System

For Healthy System

Table no. (6.b)

	Vmin	Vmax	TL	Total Cost (F1)
Loss Optimization	0.9577	1.1086	4733.0	1786198.0640

Output Figure. (6.b)



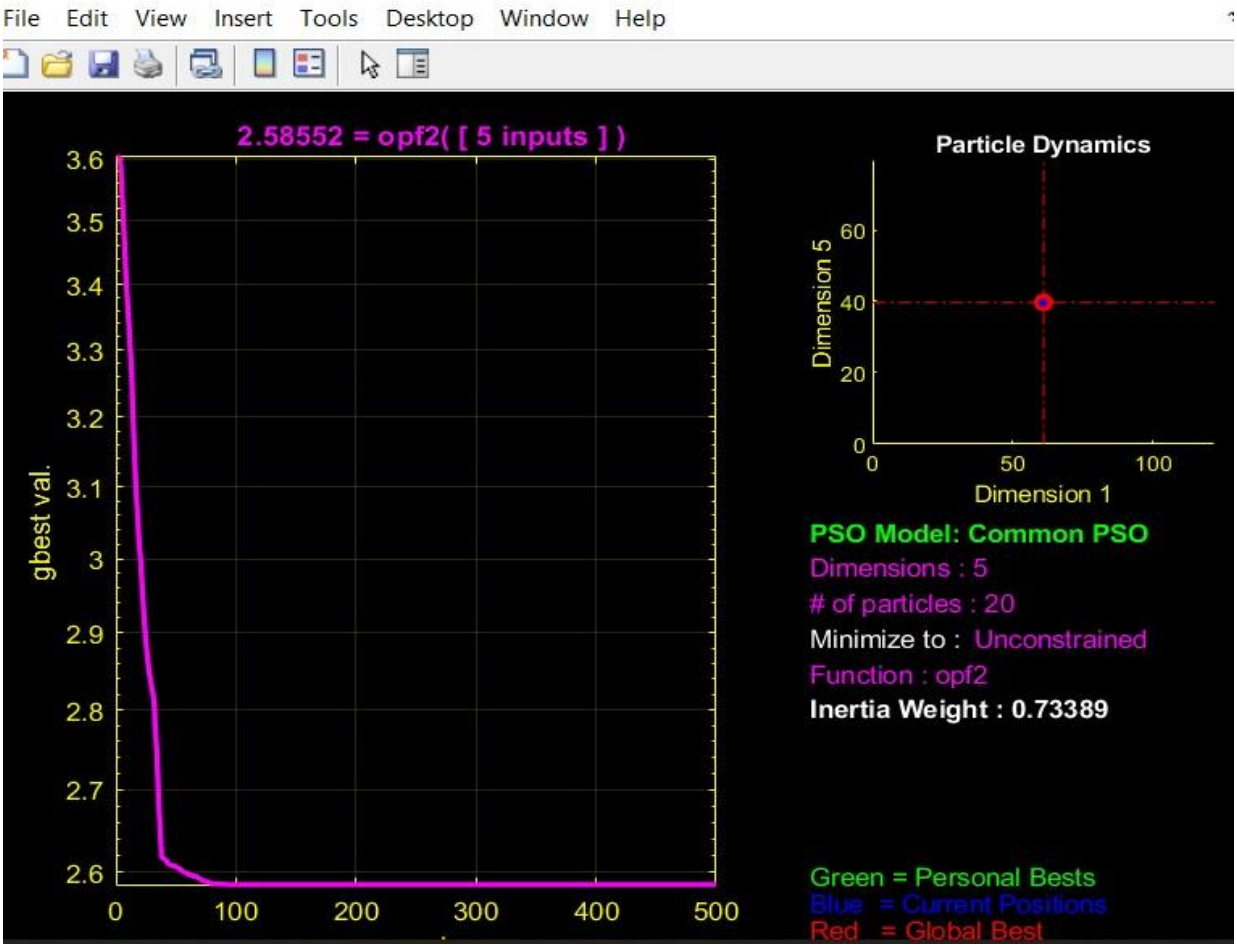
Loss Optimization of IEEE30BUS System

For Single line contingency System (n-1),1-3

Table no. (6.c)

	P1	P2	P3	P4	P5	P6	Vmin	Vmax	TL	Total Cost (F1)
Loss Optimization	9.441058	60.96439	50	40.7663837	85.16189961	39.65179	0.9903	1.082	2.5855	2.5855

Output Figure.(6.c)



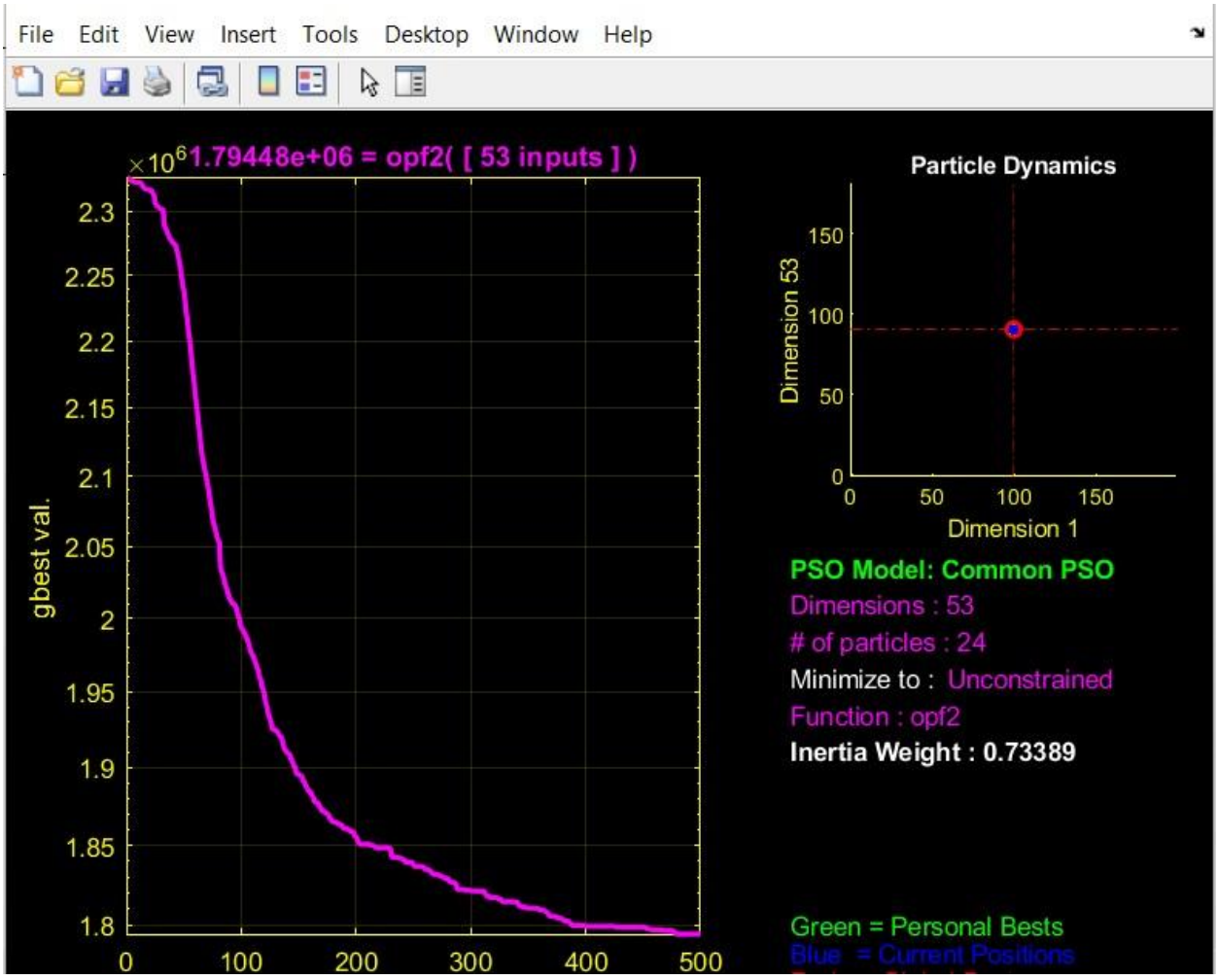
Loss Optimization of IEEE118BUS System

For Single line contingency System (n-1),1-3

Table no. (6.d)

	Vmin	Vmax	TL	Total Cost (F1)
Loss Optimization	0.9577	1.1086	4714.5	1794479.7147

Output Figure.(6.d)



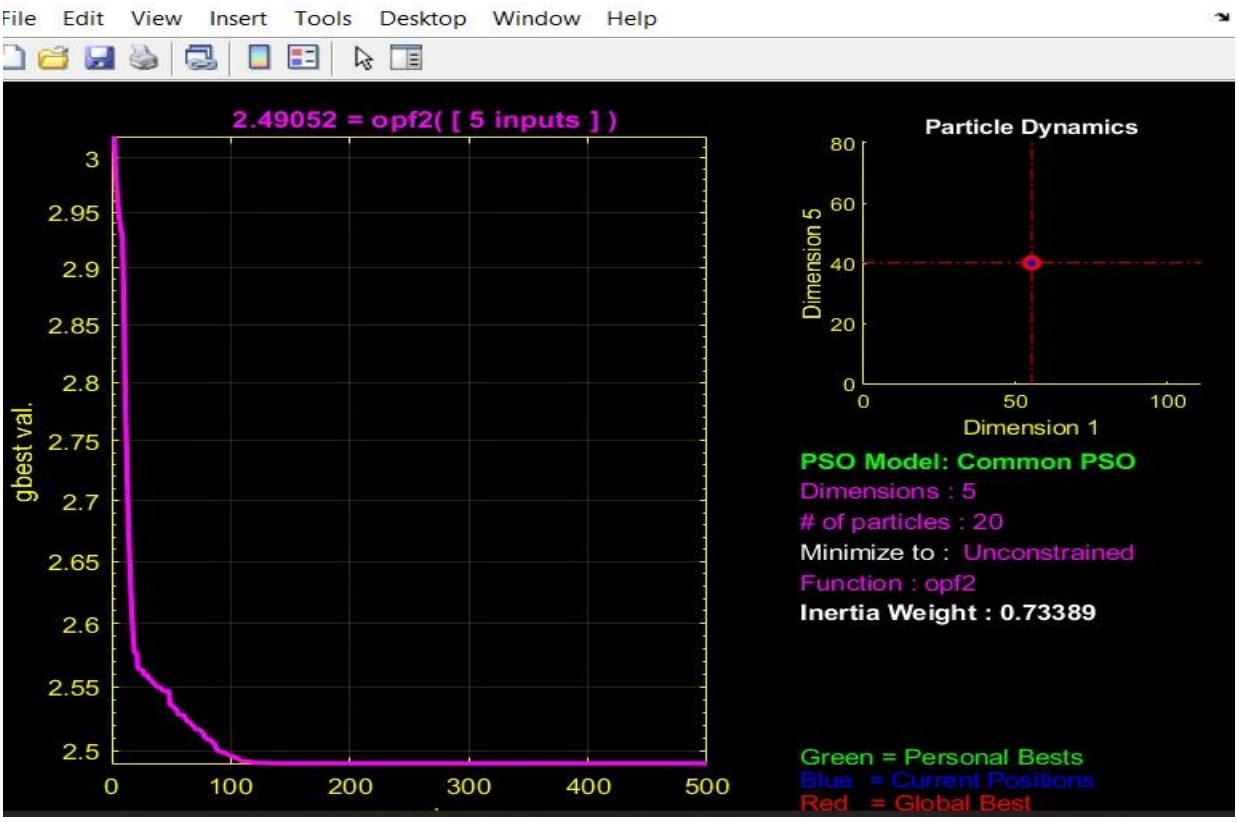
Loss Optimization of IEEE30BUS System

For Double line contingency System (n-2),1-3,2-6

Table no. (6.e)

	P1	P2	P3	P4	P5	P6	Vmin	Vmax	TL	Total Cost (F1)
Loss Optimization	9.6370	55.309	50	43.9731674	86.7387	40.23189	0.987019	1.082	2.4905	2.4905

Output Figure.(6.e)



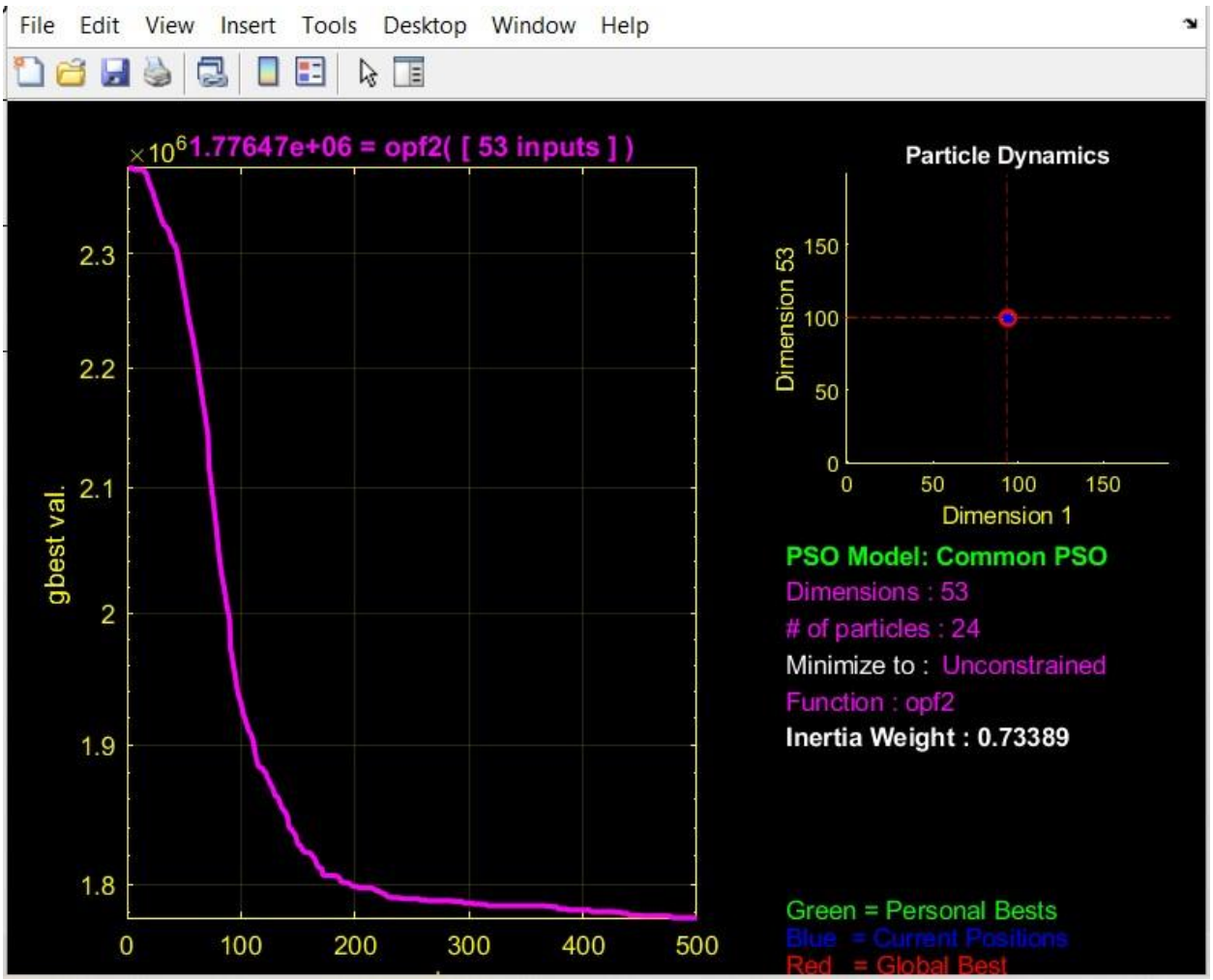
Loss Optimization of IEEE118BUS System

For Double line contingency System (n-1),1-3,2-6

Table no. (6.f)

	Vmin	Vmax	TL	Total Cost (F1)
Loss Optimization	0.9577	1.1086	4789.0	1776471.1916

Output Figure.(6.f)



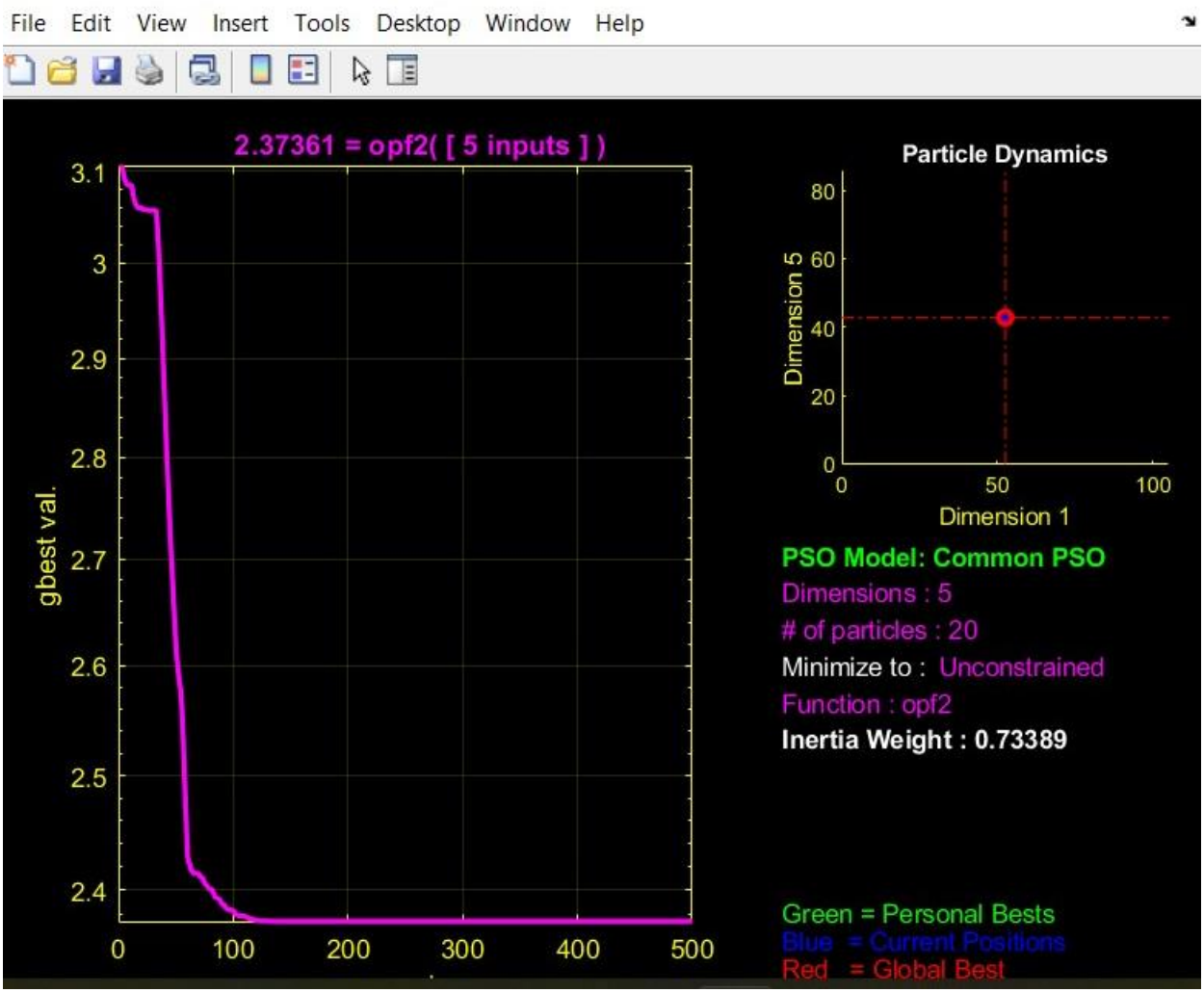
Loss Optimization of IEEE30BUS System

For Triple line contingency System (n-3),1-3,2-6,6-8

Table no. (6.g)

	P1	P2	P3	P4	P5	P6	Vmin	Vmax	TL	Total Cost (F1)
Loss Optimization	9.436787	52.57229	50	34.29605	96.75978	42.70869	0.9899	1.082	2.3736	2.3736

Output Figure.(6.g)



Chapter 7

PENALTY BASED OPTIMIZATION

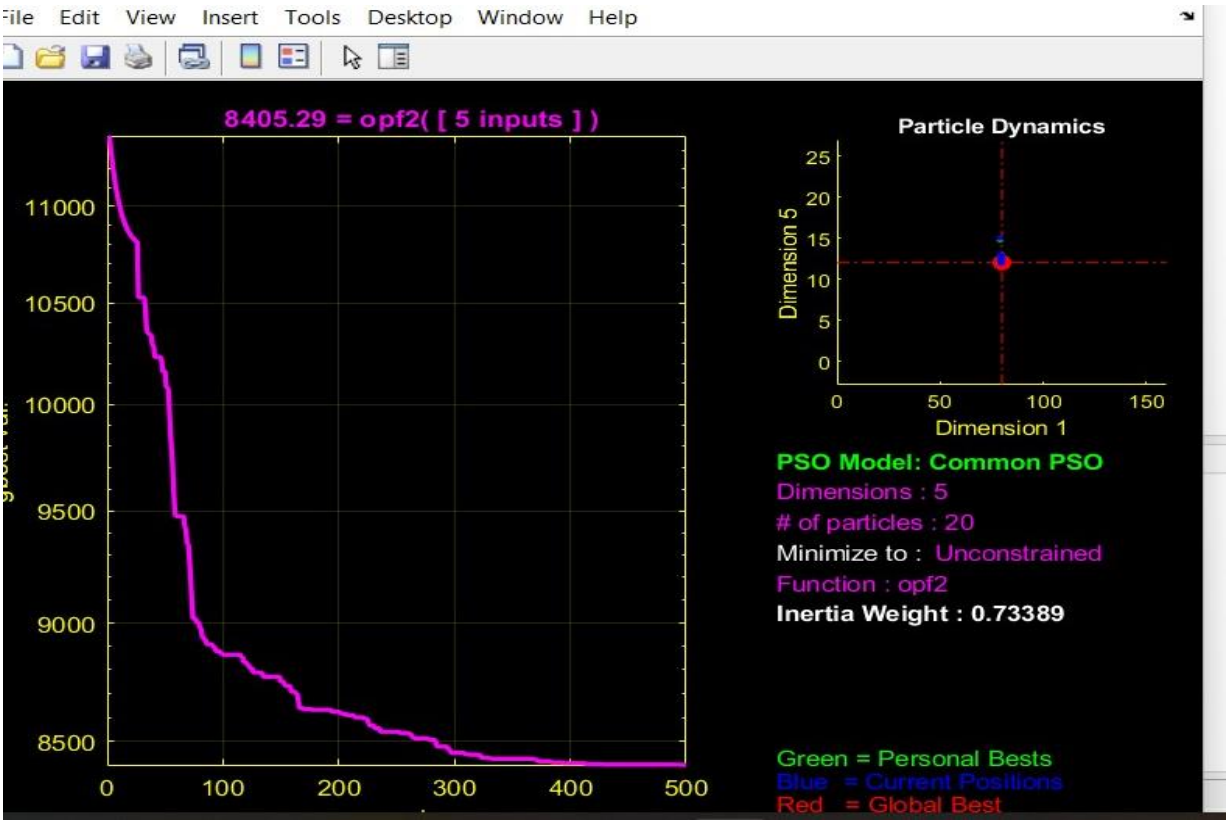
Penalty Based Cost Optimization of IEEE30BUS System

For Healthy System

Table no. (7.a)

	P1	P2	P3	P4	P5	P6	Vmin	Vmax	TL	Total Cost (F1)
Penalty based Cost Optimization	62.97857	80	50	35.0326034	46.98368298	12.02333	0.9958	1.082	3.6182	8.4053x10^3

Output Figure.(7.a)



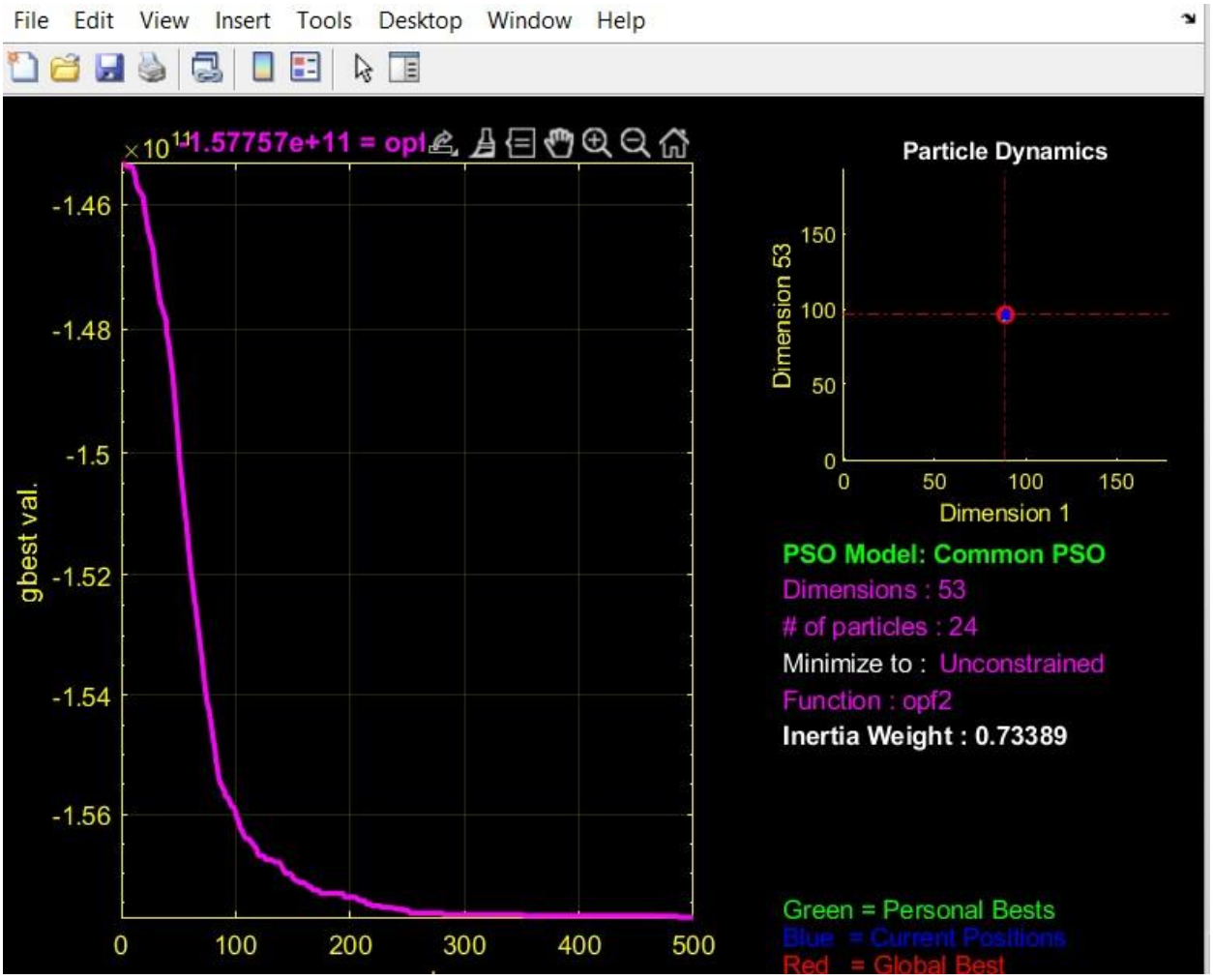
Penalty Based Cost Optimization of IEEE118BUS System

For Healthy System

Table no. (7.b)

	Vmin	Vmax	TL	Total Cost (F1)
Loss Optimization	0.9577	1.1086	2949.7	-157757466909.1

Output Figure.(7.b)



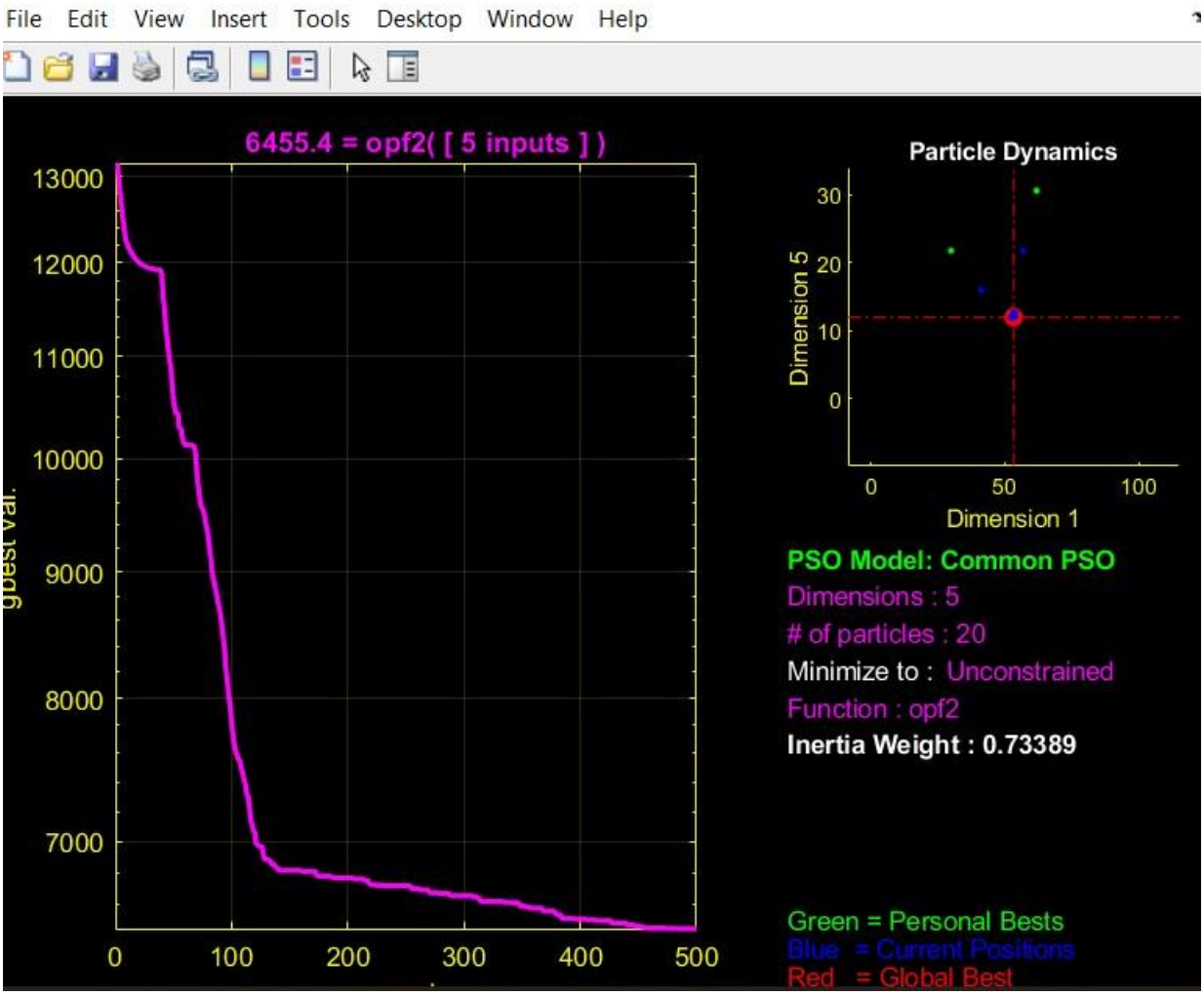
Penalty Based Cost Optimization of IEEE30BUS System

For Single line contingency System (n-1),1-3

Table no. (7.c)

	P1	P2	P3	P4	P5	P6	Vmin	Vmax	TL	Total Cost (F1)
Penalty based Cost Optimization	199.9803	53.29597	15	10	10	12	0.992	1.082	16.8762	6.4554x10^3

Output Figure.(7.c)



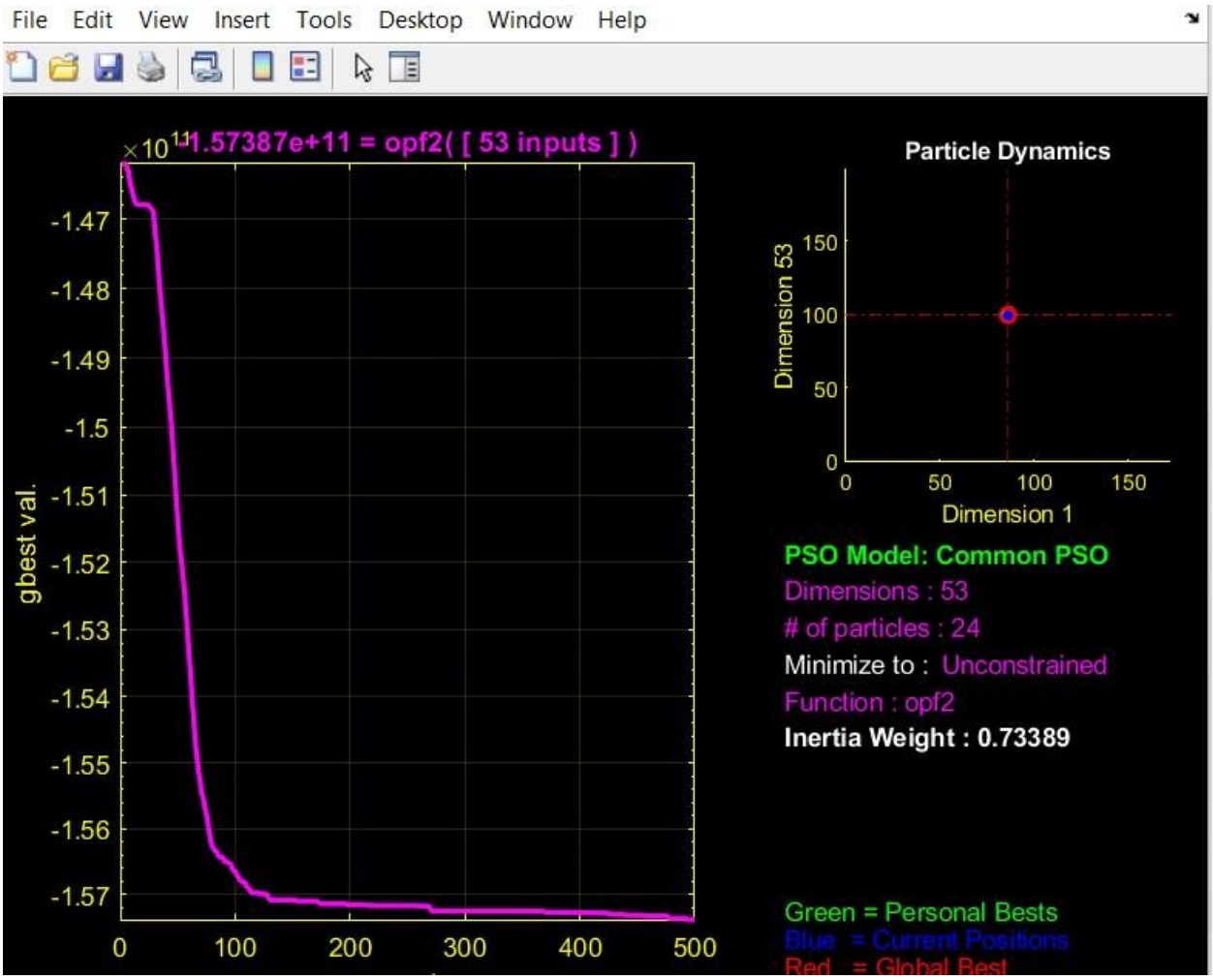
Penalty Based Cost Optimization of IEEE118BUS System

For Single line contingency System (n-1),1-3

Table no. (7.d)

	Vmin	Vmax	TL	Total Cost (F1)
Loss Optimization	0.9577	1.1086	2940.2	-157387099727.3

Output Figure.(7.d)



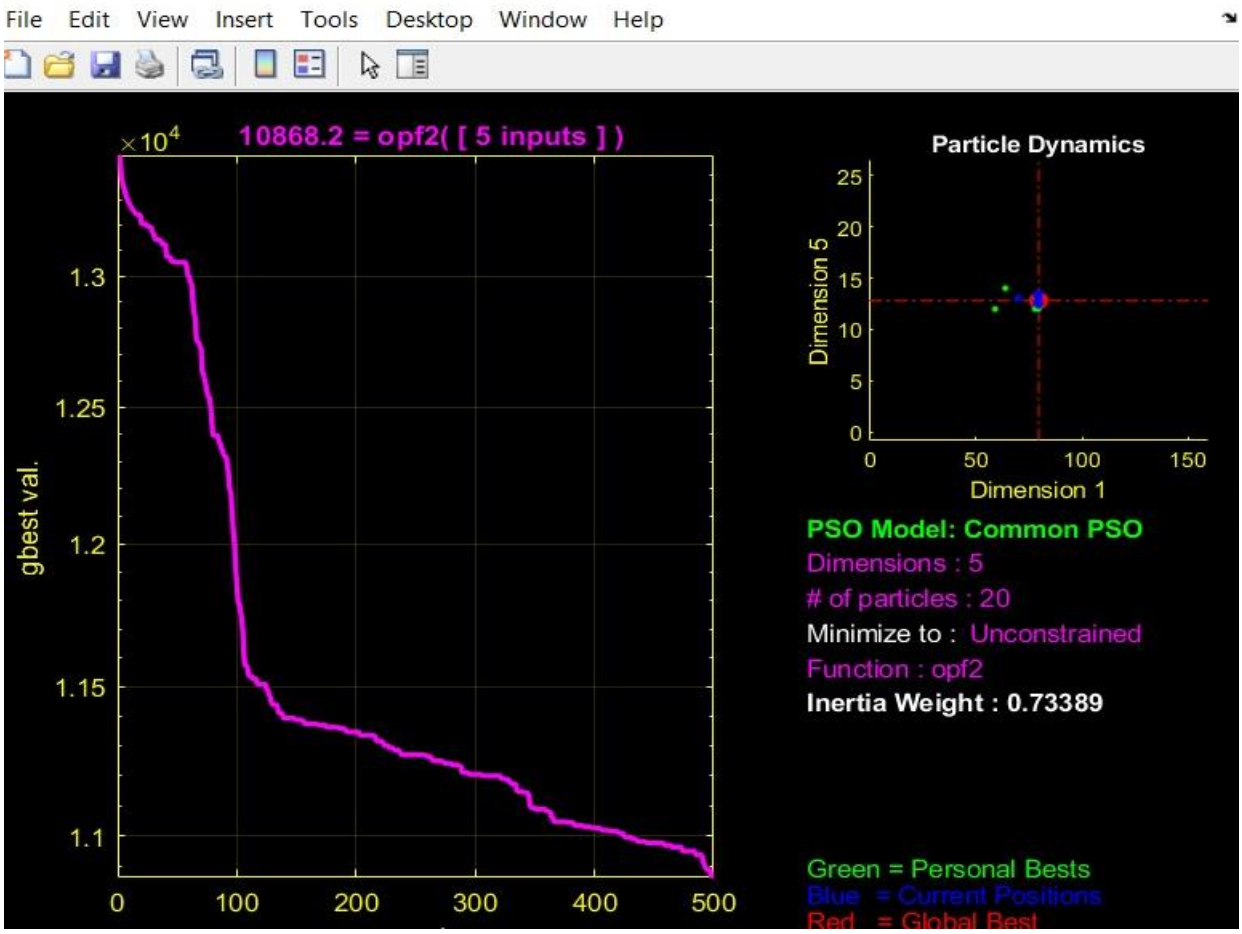
Penalty Based Cost Optimization of IEEE30BUS System

For Double line contingency System (n-2),1-3,2-6

Table no. (7.e)

	P1	P2	P3	P4	P5	P6	Vmin	Vmax	TL	Total Cost (F1)
Penalty based Cost Optimization	32.47495	79.489	49.95898	22.8	89.4684	12.826	0.9883	1.082	3.6178	1.0868x10^4

Output Figure.(7.e)



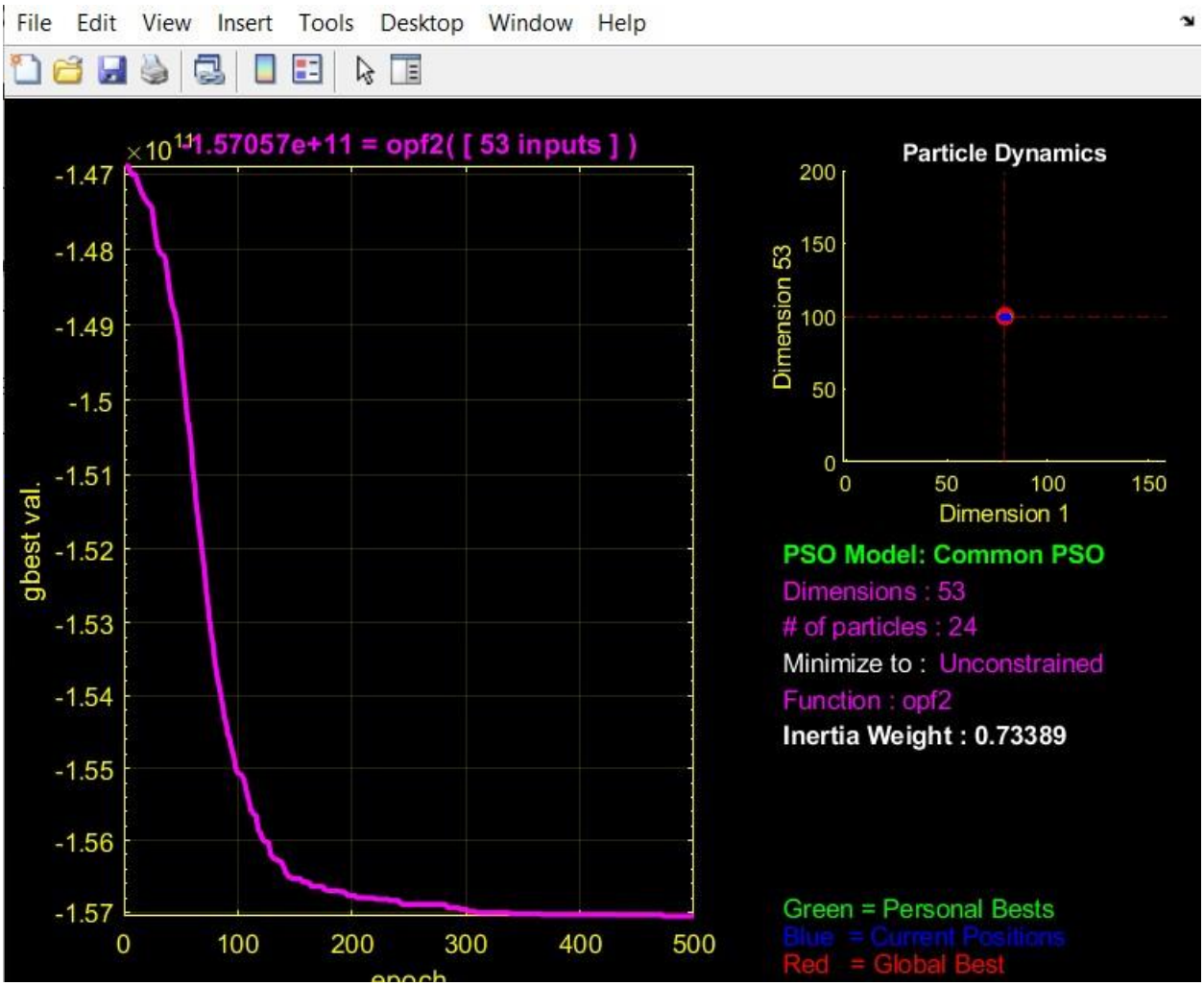
Penalty Based Cost Optimization of IEEE118BUS System

For Double line contingency System (n-1),1-3,5-6

Table no. (7.f)

	Vmin	Vmax	TL	Total Cost (F1)
Loss Optimization	0.9577	1.1086	2992.6	-157056948349.62

Output Figure.(7.f)



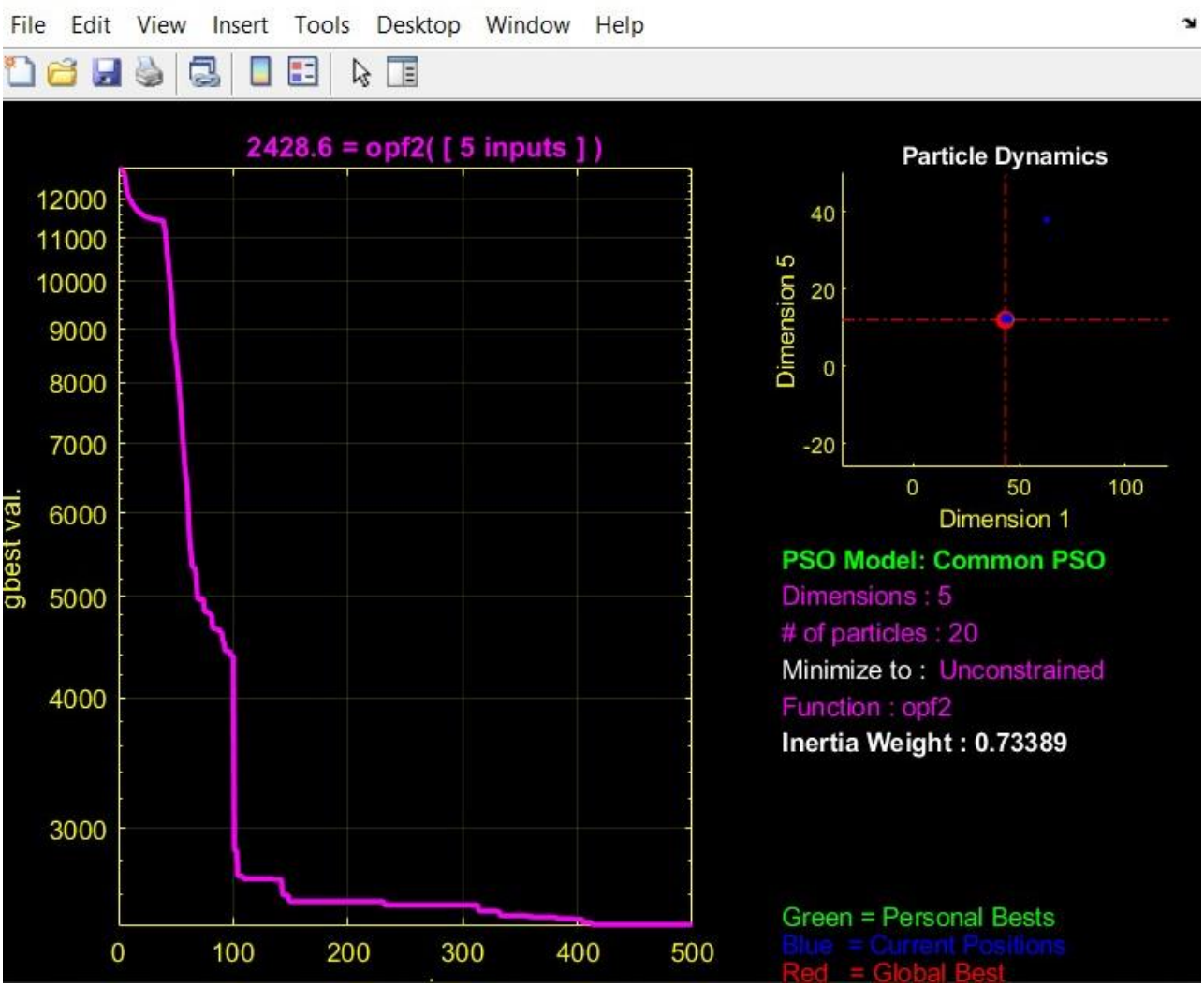
Penalty Based Cost Optimization of IEEE30BUS System

For Triple line contingency System (n-3),1-3,2-6,6-8

Table no. (7.g)

	P1	P2	P3	P4	P5	P6	Vmin	Vmax	TL	Total Cost (F1)
Penalty based Cost Optimization	34.79099	80	50	34.58953	75.29407	12.34227	0.9899	1.082	3.6169	1.0572x10^4

Output Figure.(7.g)



Chapter 8

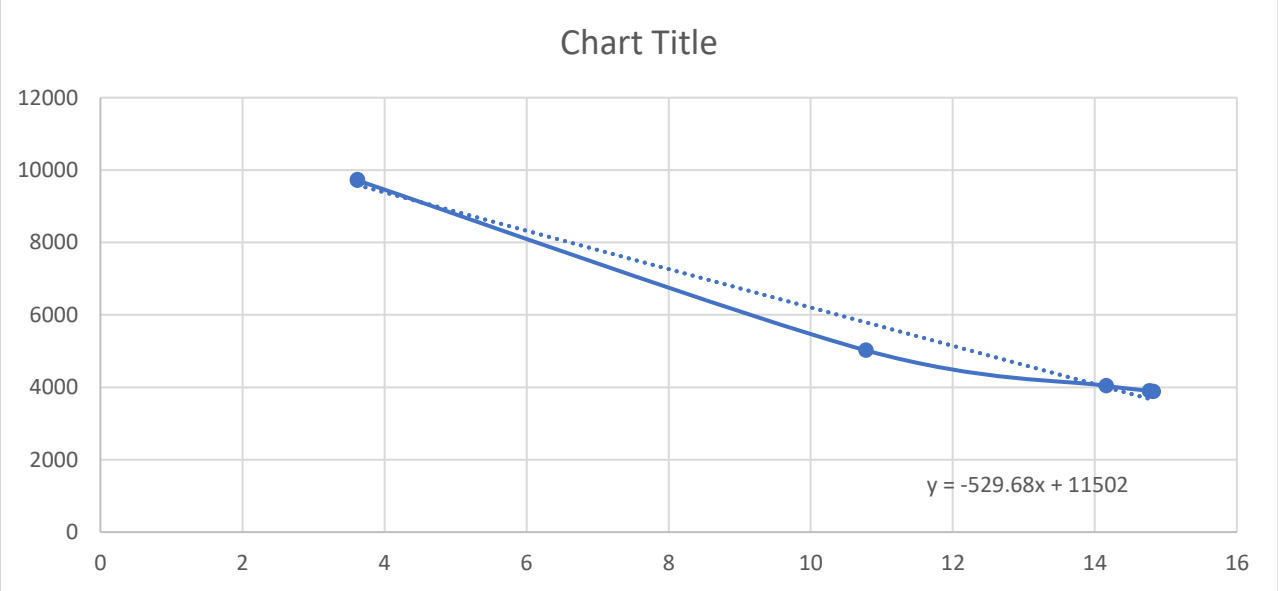
PENALTY BASED CONVERGENCE

Penalty Calculation of IEEE30BUS System

Table No.(8.a)

Penalty	P. Loss (MW)	Gen. Cost ($)
0	14.8224	3886.275
1	14.77	3901.0737
10	14.1617	4039.9593
100	10.7777	5019.482
1000	3.6167	9716.8126

Output Figure.(8.a)



Calculation

C= 11502-529.68TL

dC/dTL=-529.68

Conclusion

The security assessment, along with contingency filtering procedures developed in methods one through four, identified the worst possible contingencies, i.e. the contingencies that should be taken into account in OPF-based control tools. These are important tasks in the context of real time operation because they reduce the size of OPF problems and therefore, reduce the time required to solve them.

Generation and load rescheduling is an effective control action to avoid problems related to voltage and loss profile management, transmission line congestion and small signal instabilities. The developed methodologies use OPF control tools that identify efficacious preventive re-dispatching actions at minimum cost. The methodologies developed ensure both secure and economic operation of the system. The load curtailment strategy developed can be used in Smart Power markets for reliability of power supply and without compromising social welfare.

Although the developed OPF problems are nonlinear and non convex and a global optimum cannot be guaranteed by Particle Swarm Optimization and differential evolution algorithm, the resulting solutions are reasonable.

Finally, the developed methodologies are useful tools for system operators since they ensure the secure operation of current power systems.

Future scope of the work :-

- Application of efficient techniques to reduce the computational burden of the developed methodologies. These techniques include parallel computation, ingenious load curtailment strategies powerful methods of eigen value analysis, and closed-form sensitivity formulas.
- Development of dynamic models for the inclusion of transient stability of modern power networks.
- Investigation of the effects of distributed generation in the developed methodologies.
- Case studies on larger and real time systems in collaboration with system operator.
- Exploring the effectiveness of the proposed optimization technique in both developed and emerging power markets.

APPENDIX

Introduction to MATLAB

Matlab is a commercial software that provides a computing environment that allows for sophisticated ways of developing and debugging computer code, executing programs, and visualizing the output. Matlab is also a computer language (sort of a mix between C and Fortran) and this exercise for you to work through is mainly concerned with some of the language aspects that we will use extensively throughout the course. Please read through the more comprehensive and verbose Matlab Intro and familiarize yourself with Matlab. All of our Windows and Linux machines have Matlab installed and after starting up the program, you will be presented with an interactive window where you can type in commands as we indicate below. Please also familiarize yourself with the other components of the development environment, such as the built-in editor for Matlab programs, which are called "m-files", so that you can be more efficient in writing and debugging codes. There are numerous Matlab-provided help resources accessible through the environment, including video tutorials, access to the help pages, along with extensive documentation on the web. Also note that there is a free clone of Matlab called octave. Given that Matlab often uses freely available computational routines underneath the hood, it was fairly easy to reproduce the computational basics of Matlab. However, the Matlab people also added a bunch of proprietary visualization tools which are not available in octave. Another alternative is to use the freely available Python language and its MatPlotLib package, but we will not have time to explore such intriguing options in class.

A.1. Historical background

Cleve Moler, the chairman of the computer science department at the University of New Mexico, started developing MATLAB in the late 1970s. He designed it to give his students access to LINPACK and EISPACK without them having to learn Fortran. It soon spread to other universities and found a strong audience within the applied mathematics community. Jack Little, an engineer, was exposed to it during a visit Moler made to Stanford University in 1983. Recognizing its commercial potential, he joined with Moler and Steve Bangert .They rewrote MATLAB in C and founded MathWorks in 1984 to continue its development. These rewritten libraries were known as JACKPAC. In 2000, MATLAB was rewritten to use a newer set of libraries for matrix manipulation, LAPACK. MATLAB was first adopted by researchers and practitioners in control engineering, Little's specialty, but quickly spread to many other domains. It is now also used in education, in particular the teaching of linear algebra, numerical analysis, and is popular amongst scientists involved in image processing.

A.2. Interfacing with other languages

MATLAB can call functions and subroutines written in the programming languages C or Fortran.[22] A wrapper function is created allowing MATLAB data types to be passed and returned. The dynamically loadable object files created by compiling such functions are termed "MEX-files" (for **M**ATLAB **ex**ecutable). Since 2014 increasing two-way interfacing with Python is being added. Libraries written in Perl, Java, ActiveX or .NET can be directly called from MATLAB, and many MATLAB libraries (for example XML or SQL support) are implemented as wrappers around Java or ActiveX libraries. Calling MATLAB from Java is more complicated, but can be done with a MATLAB toolbox which is sold separately by MathWorks, or using an undocumented mechanism called JMI (Java-to-MATLAB Interface), (which should not be confused with the unrelated Java Metadata Interface that is also called JMI). As alternatives to the MuPAD based Symbolic Math Toolbox available from MathWorks, MATLAB can be connected to Maple or Mathematica .Libraries also exist to import and export MathML.

A.3. MATLAB simulation

Simulation modeling is the process of creating and analyzing a digital prototype of a physical **model** to predict its performance in the real world. **Simulation modeling** is used to help designers and engineers understand whether, under what conditions, and in which ways a part could fail and what loads it can withstand.

Introduction to simulink

Simulink provides a graphical editor, customizable block libraries, and solvers for modeling and simulating dynamic systems. It is integrated with MATLAB®, enabling you to incorporate MATLAB algorithms into models and export simulation results to MATLAB for further analysis.

Model-Based Design is a process that enables fast and cost-effective development of dynamic systems, including control systems, signal processing, and communications systems. In Model-Based Design, a system model is at the center of the development process, from requirements development through design, implementation, and testing. The model is an executable specification that you continually refine throughout the development process. After model development, simulation shows whether the model works correctly.

When software and hardware implementation requirements are included with the model, such as fixed-point and timing behavior, you can generate code for embedded deployment and create test benches for system verification, saving time and avoiding manually coded errors.

Model-Based Design allows you to improve efficiency by:

- Using a common design environment across project teams
- Linking designs directly to requirements
- Integrating testing with design to continuously identify and correct errors
- Refining algorithms through multi-domain simulation
- Generating embedded software code
- Developing and reusing test suites
- Generating documentation
- Reusing designs to deploy systems across multiple processors and hardware targets

A.4.Modeling, Simulation, and Analysis with Simulink

With Simulink®, you can move beyond idealized linear models to explore realistic nonlinear models, factoring in friction, air resistance, gear slippage, hard stops, and the other parameters that describe real-world phenomena. Simulink enables you to think of the development environment as a laboratory for modeling and analyzing systems that would not be possible or practical otherwise.

Whether you are interested in the behavior of an automotive clutch system, the flutter of an airplane wing, or the effect of the monetary supply on the economy, Simulink provides you with the tools to model and simulate almost any real-world problem. Simulink also provides examples that model a wide variety of real-world phenomena.

Tool for Modeling

Simulink provides a graphical editor for building models as block diagrams, allowing you to draw models as you would with pencil and paper. Simulink also includes a comprehensive library of sink, source, linear and nonlinear component, and connector blocks. If these blocks do not meet your needs, however, you can also create your own blocks. The interactive environment simplifies the modeling process, eliminating the need to formulate differential and difference equations in a language or program.

Models are hierarchical, so you can build models using both top-down and bottom-up approaches. You can view the system at a high level, then drill down to see increasing levels of model detail. This approach provides insight into how a model is organized.

Tool for Simulation

After you define a model, you can simulate its dynamic behavior using a choice of mathematical integration methods, either interactively in Simulink or by entering commands in the MATLAB® Command Window. Commands are particularly useful for running a batch of simulations. For example, if you are doing Monte Carlo simulations or want to apply a parameter across a range of values, you can use MATLAB scripts.

Using scopes and other display blocks, you can see the simulation results while a simulation runs. You can then change parameters and see what happens for "what if" exploration. You can save simulation results in the MATLAB workspace for post processing and visualization.

Tool for Analysis

Model analysis tools include linearization and trimming tools you can access from MATLAB, plus the many tools in MATLAB and its application toolboxes. Because MATLAB and Simulink are integrated, you can simulate, analyze, and revise your models in either environment.

A.5..Interaction with MATLAB Environment

Simulink software requires MATLAB to run, and it depends on it to define and evaluate model and block parameters. Simulink can also use many MATLAB features. For example, Simulink can use the MATLAB environment to:

- Define model inputs.
- Store model outputs for analysis and visualization.
- Perform functions within a model, through integrated calls to MATLAB operators and functions.

A.6.Basic simulation window

Simulation is a process in which you validate and verify a model by comparing simulation results with:

- Data collected from a real system.
- Functionality described in the model requirements.

Perform the simulation workflow after you've finished building your model and a simulation completes without errors. The steps in a typical simulation workflow include:

- Prepare for simulation.
- Run and evaluate simulation.

Prepare for Simulation

Define the external input and output interfaces

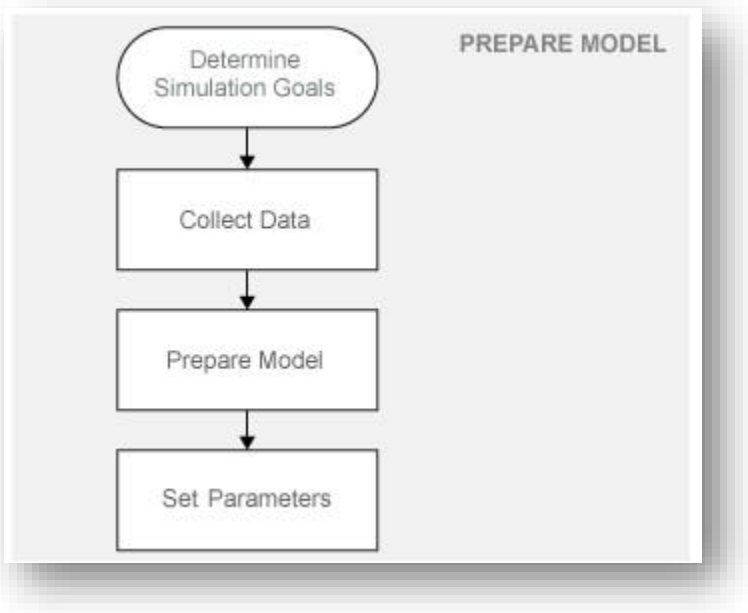


Fig. A1 Flowchart to prepare model

Determine Simulation Goals

Before simulating a model, you need to understand your goals and requirements. Ask yourself these questions to help plan your simulations:

- What questions do you want the simulation to answer?
- How accurately does the model need to represent the system?

Some possible simulation goals:

- Understand input to output causality — For a given input set and nominal parameter values, look at how the inputs flow through the system to the outputs.
- Verify model — Compare simulation results with collected data from the modeled system. Iteratively debug and improve the design.
- Optimize parameters — Change parameters and compare simulation runs.
- Visualize results — Send simulation results to a plot or print in a report.

Collect Data

Collect input and output data from an actual system. Use measured input data to drive the simulation. Use measured output data to compare with the simulation results from your model.

You will use measured input data to drive a simulation and measured output data to verify the simulation results.

Prepare Model

Preparing a model for simulation includes defining the external interfaces for input data and control signals, and output signals for viewing and recording simulation results.

Set Parameters

For the first simulation, use model parameters from the validated model. After comparing the simulation results with measured output data, change model parameters to more accurately represent the modeled system.

Run and Evaluate Simulation

Simulate your model and verify that the simulation results match the measured data from the modeled system.

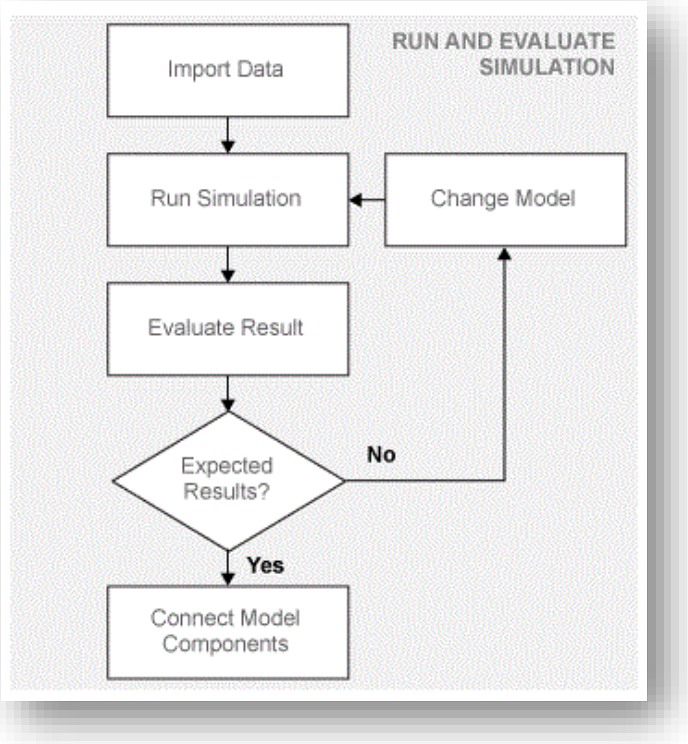


Fig. A2 Flowchart of simulation process

Import Data

Simulink® enables you to import data into your model.

- Use the Signal Builder block to import input signals from a Microsoft® Excel® file (XLSX, XLS) or a comma-separated value file (CSV). Simulink saves data imported from an Excel file using a Signal Builder block with the model and loads the data into memory when you open the model.
- For large data sets, use a MATLAB MAT-file with an Import block.

Run Simulation

Using measured input data, run a simulation and save results.

Evaluate Result

Evaluate the differences between simulated output and measured output data. Use the evaluation to verify the accuracy of your model and how well it represents the system behavior. Decide if the accuracy of your model adequately represents the dynamic system you are modeling.

Change Model

Determine the changes to improve your model. Model changes include:

- Parameters — Some parameters were initially estimated and approximated. Optimize and update parameters.
- Adding structure — Some parts or details of the system were not modeled. Add missing details.

forcecol.m

```
% forcecol.m
% function to force a vector to be a single column
%
% Brian Birge
% Rev 1.0
% 7/1/98

function[out]=forcecol(in)
len=prod(size(in));
out=reshape(in,[len,1]);
```

forcerow.m

```
% forcerow.m
% function to force a vector to be a single row
%
% Brian Birge
% Rev 2.0 - vectorized
% 7/1/98

function[out]=forcerow(in)
len=prod(size(in));
out=reshape(in,[1,len]);
```

goplotpso.m

```
% goplotpso.m
% default plotting script used in PSO functions
%
% this script is not a function,
% it is a plugin for the main PSO routine (pso_Trelea_vectorized)
% so it shares all the same variables, be careful with variable names
% when making your own plugin

% Brian Birge
% Rev 2.0
% 3/1/06

% setup figure, change this for your own machine
 clf
 set(gcf,'Position',[651    31   626   474]); % this is the computer
dependent part
 %set(gcf,'Position',[743    33   853   492]);
 set(gcf,'Doublebuffer','on');

% particle plot, upper right
 subplot('position',[.7,.6,.27,.32]);
 set(gcf,'color','k')

 plot3(pos(:,1),pos(:,D),out,'b.','Markersize',7)
```

```
hold on
 plot3(pbest(:,1),pbest(:,D),pbestval,'g.','Markersize',7);
 plot3(gbest(1),gbest(D),gbestval,'r.','Markersize',25);

 % crosshairs
 offx = max(abs(min(min(pbest(:,1)),min(pos(:,1)))),...
            abs(max(max(pbest(:,1)),max(pos(:,1)))));

 offy = max(abs(min(min(pbest(:,D)),min(pos(:,D)))),...
            abs(min(max(pbest(:,D)),max(pos(:,D)))));
 plot3([gbest(1)-offx;gbest(1)+offx],...
       [gbest(D);gbest(D)],...
       [gbestval;gbestval],...
       'r-.');
 plot3([gbest(1);gbest(1)],...
       [gbest(D)-offy;gbest(D)+offy],...
       [gbestval;gbestval],...
       'r-.');

 hold off

 xlabel('Dimension 1','color','y')
 ylabel(['Dimension ',num2str(D)],'color','y')
 zlabel('Cost','color','y')

 title('Particle Dynamics','color','w','fontweight','bold')

 set(gca,'Xcolor','y')
 set(gca,'Ycolor','y')
 set(gca,'Zcolor','y')
 set(gca,'color','k')

 % camera control
 view(2)
 try
   axis([gbest(1)-offx,gbest(1)+offx,gbest(D)-offy,gbest(D)+offy]);
 catch
   axis([VR(1,1),VR(1,2),VR(D,1),VR(D,2)]);
 end

% error plot, left side
 subplot('position',[0.1,0.1,.475,.825]);
  semilogy(tr(find(~isnan(tr))),'color','m','linewidth',2)
  %plot(tr(find(~isnan(tr))),'color','m','linewidth',2)
  xlabel('epoch','color','y')
  ylabel('gbest val.','color','y')

  if D==1
     titstr1=sprintf(['%11.6g = %s( [ %9.6g ] )'],...
                 gbestval,strrep(functname,'_','\_'),gbest(1));
  elseif D==2
     titstr1=sprintf(['%11.6g = %s( [ %9.6g, %9.6g ] )'],...
                 gbestval,strrep(functname,'_','\_'),gbest(1),gbest(2));
  elseif D==3
     titstr1=sprintf(['%11.6g = %s( [ %9.6g, %9.6g, %9.6g ] )'],...
```

```
gbestval,strrep(functname,'_','\_'),gbest(1),gbest(2),gbest(3));
  else
     titstr1=sprintf(['%11.6g = %s( [ %g inputs ] )'],...
                gbestval,strrep(functname,'_','\_'),D);
  end
  title(titstr1,'color','m','fontweight','bold');

  grid on
%  axis tight

  set(gca,'Xcolor','y')
  set(gca,'Ycolor','y')
  set(gca,'Zcolor','y')
  set(gca,'color','k')

  set(gca,'YMinorGrid','off')

% text box in lower right
% doing it this way so I can format each line any way I want
subplot('position',[.62,.1,.29,.4]);
  clear titstr
  if trelea==0
       PSOtype  = 'Common PSO';
       xtraname = 'Inertia Weight : ';
       xtraval  = num2str(iwt(length(iwt)));

     elseif trelea==2 | trelea==1

       PSOtype  = (['Trelea Type ',num2str(trelea)]);
       xtraname = ' ';
       xtraval  = ' ';

     elseif trelea==3
       PSOtype  = (['Clerc Type 1"']);
       xtraname = '\chi value : ';
       xtraval  = num2str(chi);

  end
  if isnan(errgoal)
    errgoalstr='Unconstrained';
  else
    errgoalstr=num2str(errgoal);
  end
  if minmax==1
     minmaxstr = ['Maximize to : '];
  elseif minmax==0
     minmaxstr = ['Minimize to : '];
  else
     minmaxstr = ['Target   to : '];
  end

  if rstflg==1
     rststat1 = 'Environment Change';
     rststat2 = ' ';
  else
     rststat1 = ' ';
     rststat2 = ' ';
```

```
end

titstr={'PSO Model: '      ,PSOtype;...
        'Dimensions : '    ,num2str(D);...
        '# of particles : ',num2str(ps);...
        minmaxstr          ,errgoalstr;...
        'Function : '      ,strrep(functname,'_','\_');...
        xtraname           ,xtraval;...
        rststat1           ,rststat2};

text(.1,1,[titstr{1,1},titstr{1,2}],'color','g','fontweight','bold');
hold on
text(.1,.9,[titstr{2,1},titstr{2,2}],'color','m');
text(.1,.8,[titstr{3,1},titstr{3,2}],'color','m');
text(.1,.7,[titstr{4,1}],'color','w');
text(.55,.7,[titstr{4,2}],'color','m');
text(.1,.6,[titstr{5,1},titstr{5,2}],'color','m');
text(.1,.5,[titstr{6,1},titstr{6,2}],'color','w','fontweight','bold');
text(.1,.4,[titstr{7,1},titstr{7,2}],'color','r','fontweight','bold');

% if we are training a neural net, show a few more parameters
if strcmp('pso_neteval',functname)
  % net is passed from trainpso to pso_Trelea_vectorized in case you are
  % wondering where that structure comes from
  hiddlyrstr = [];
  for lyrcnt=1:length(net.layers)
     TF{lyrcnt} = net.layers{lyrcnt}.transferFcn;
     Sn(lyrcnt) = net.layers{lyrcnt}.dimensions;
     hiddlyrstr = [hiddlyrstr,', ',TF{lyrcnt}];
  end
  hiddlyrstr = hiddlyrstr(3:end);

  text(0.1,.35,['#neur/lyr = [ ',num2str(net.inputs{1}.size),'  ',...
            num2str(Sn),' ]'],'color','c','fontweight','normal',...
            'fontsize',10);
  text(0.1,.275,['Lyr Fcn: ',hiddlyrstr],...
     'color','c','fontweight','normal','fontsize',9);

end


legstr = {'Green = Personal Bests';...
          'Blue  = Current Positions';...
          'Red   = Global Best'};
text(.1,0.025,legstr{1},'color','g');
text(.1,-.05,legstr{2},'color','b');
text(.1,-.125,legstr{3},'color','r');

hold off

set(gca,'color','k');
set(gca,'visible','off');

drawnow
```

normmat.m

```
function [out,varargout]=normmat(x,newminmax,flag)
% normmat.m
% takes a matrix and reformats the data to fit between a new range
%
% Usage:
%    [xprime,mins,maxs]=normmat(x,range,method)
%
% Inputs:
%     x - matrix to reformat of dimension MxN
%     range - a vector or matrix specifying minimum and maximum values for
the new matrix
%         for method = 0, range is a 2 element row vector of [min,max]
%         for method = 1, range is a 2 row matrix with same column size as
%                         input matrix with format of [min1,min2,...minN;
%                                                      max1,max2,...maxM];
%         for method = 2, range is a 2 column matrix with same row size as
%                         input matrix with format of [min1,max1;
%                                                      min2,max2;
%                                                      ... , ...;
%                                                      minM,maxM];
%             alternatively for method 1 and 2, can input just a 2 element
vector as in method 0
%             this will just apply the same min/max across each column or row
respectively
%     method - a scalar flag with the following function
%         = 1, normalize each column of the input matrix separately
%         = 2, normalize each row of the input matrix separately
%         = 0, normalize matrix globally
% Outputs:
%     xprime - new matrix normalized per method
%     mins,maxs - optional outputs return the min and max vectors of the
original matrix x
%         used for recovering original matrix from xprime
%
% example: x = [-10,3,0;2,4.1,-7;3.4,1,0.01]
%          [xprime,mins,maxs]=normmat(x,[0,10],0)

% Brian Birge
% Rev 2.1
% 3/16/06 - changed name of function to avoid same name in robust control
% toolbox
%-------------------------------------------------------------------------
---------------------------
if flag==0

  a=min(min((x)));
  b=max(max((x)));
  if abs(a)>abs(b)
     large=a;
     small=b;
  else
     large=b;
     small=a;
```

```
  end
  temp=size(newminmax);
  if temp(1)~=1
     error('Error: for method=0, range vector must be a 2 element row
vector');
  end
  den=abs(large-small);
  range=newminmax(2)-newminmax(1);
  if den==0
     out=x;
  else
     z21=(x-a)/(den);
     out=z21*range+newminmax(1)*ones(size(z21));
  end

%-------------------------------------------------------------------------
---------------------------
elseif flag==1
 a=min(x,[],1);
 b=max(x,[],1);
  for i=1:length(b)
     if abs(a(i))>abs(b(i))
        large(i)=a(i);
        small(i)=b(i);
     else
        large(i)=b(i);
        small(i)=a(i);
     end
  end
  den=abs(large-small);
  temp=size(newminmax);
  if temp(1)*temp(2)==2
     newminmaxA(1,:)=newminmax(1).*ones(size(x(1,:)));
     newminmaxA(2,:)=newminmax(2).*ones(size(x(1,:)));
  elseif temp(1)>2
     error('Error: for method=1, range matrix must have 2 rows and same
columns as input matrix');
  else
     newminmaxA=newminmax;
  end

  range=newminmaxA(2,:)-newminmaxA(1,:);
  for j=1:length(x(:,1))
     for i=1:length(b)
        if den(i)==0
           out(j,i)=x(j,i);
        else
           z21(j,i)=(x(j,i)-a(i))./(den(i));
           out(j,i)=z21(j,i).*range(1,i)+newminmaxA(1,i);
        end
     end
  end
%-------------------------------------------------------------------------
---------------------------
elseif flag==2
  a=min(x,[],2);
  b=max(x,[],2);
```

```
  for i=1:length(b)
     if abs(a(i))>abs(b(i))
        large(i)=a(i);
        small(i)=b(i);
     else
        large(i)=b(i);
        small(i)=a(i);
     end
  end
  den=abs(large-small);
  temp=size(newminmax);
  if temp(1)*temp(2)==2
     newminmaxA(:,1)=newminmax(1).*ones(size(x(:,1)));
     newminmaxA(:,2)=newminmax(2).*ones(size(x(:,1)));
  elseif temp(2)>2
     error('Error: for method=2, range matrix must have 2 columns and same
rows as input matrix');
  else
     newminmaxA=newminmax;
  end

  range=newminmaxA(:,2)-newminmaxA(:,1);
  for j=1:length(x(1,:))
     for i=1:length(b)
        if den(i)==0
           out(i,j)=x(i,j);
        else
           z21(i,j)=(x(i,j)-a(i))./([forcecol(den(i))]);
           out(i,j)=z21(i,j).*range(i,1)+newminmaxA(i,1);
        end
     end
  end

end
%-------------------------------------------------------------------------
---------------------------
varargout{1}=a;
varargout{2}=b;

return
```

Opf2.m

```
function [F1 f3 PP vv ]=opf2(x)
% x=rand(5,5);
 global busdata linedata gencost
 basemva=100;
 Pdt=283.4;
mm1 =length(x(:,1));
 nn1=length(gencost(:,1));
  for ii=1:mm1
        for i=1:nn1-1;
       xx=gencost(i+1,1);
       busdata(xx,7)=x(ii,i);
    end
```

```
[Pgg P vv]=pflow(busdata,linedata);
if Pgg(1)>gencost(1,6);
    Pgg(1)=gencost(1,6);
else
end
TL=basemva*sum(P);
if min(vv)<0.96
    lam2=100;
    %p2=min(vv)*1000;
else lam2=0;
end
if TL>3.6184
    lam3=11502-529.68*TL;
    %p2=min(vv)*1000;
else lam3=0;
end
lam=100*abs(sum(Pgg)-TL-Pdt)+lam2;
a1=gencost(:,2);
b1=gencost(:,3);
c1=gencost(:,4);
%F1(ii,1)=(Pgg.*Pgg)*a1+Pgg*b1+sum(c1)+lam;
f3(ii,1)=(Pgg.*Pgg)*a1+Pgg*b1+sum(c1);
 %F1(ii,1)=TL;
 F1(ii,1)=(Pgg.*Pgg)*a1+Pgg*b1+sum(c1)+lam+lam3;
  VV(ii,:)=vv;
    PP(ii,:)=Pgg;
  end
```

pflow.m

```
function [Pgg P vv]=pflow(busdata,linedata);
 % formation of  Y bus
j=sqrt(-1);
nl = linedata(:,1); nr = linedata(:,2); R = linedata(:,3);
X = linedata(:,4); Bc = j*linedata(:,5); a = linedata(:, 6);
nbr=length(linedata(:,1)); nbus = max(max(nl), max(nr));
Z = R + j*X; y= ones(nbr,1)./Z;         %branch admittance
for n = 1:nbr
if a(n) <= 0  a(n) = 1; else end
Ybus=zeros(nbus,nbus);     % initialize Ybus to zero
% formation of the off diagonal elements
for k=1:nbr;
       Ybus(nl(k),nr(k))=Ybus(nl(k),nr(k))-y(k)/a(k);
       Ybus(nr(k),nl(k))=Ybus(nl(k),nr(k));
    end
end
% formation of the diagonal elements
for  n=1:nbus
     for k=1:nbr
         if nl(k)==n
         Ybus(n,n) = Ybus(n,n)+y(k)/(a(k)^2) + Bc(k);
         elseif nr(k)==n
         Ybus(n,n) = Ybus(n,n)+y(k) +Bc(k);
         else, end
     end
end
```

```
basemva = 100;  accuracy = 0.005; maxiter =5;
ns=0; ng=0; Vm=0; delta=0; yload=0; deltad=0;
nbus = length(busdata(:,1));
for k=1:nbus
n=busdata(k,1);
kb(n)=busdata(k,2); Vm(n)=busdata(k,3); delta(n)=busdata(k, 4);
Pd(n)=busdata(k,5); Qd(n)=busdata(k,6); Pg(n)=busdata(k,7); Qg(n) =
busdata(k,8);
Qmin(n)=busdata(k, 9); Qmax(n)=busdata(k, 10);
Qsh(n)=busdata(k, 11);
    if Vm(n) <= 0  Vm(n) = 1.0; V(n) = 1 + j*0;
    else delta(n) = pi/180*delta(n);
         V(n) = Vm(n)*(cos(delta(n)) + j*sin(delta(n)));
         P(n)=(Pg(n)-Pd(n))/basemva;
         Q(n)=(Qg(n)-Qd(n)+ Qsh(n))/basemva;
         S(n) = P(n) + j*Q(n);
    end
end
for k=1:nbus
if kb(k) == 1, ns = ns+1; else, end
if kb(k) == 2 ng = ng+1; else, end
ngs(k) = ng;
nss(k) = ns;
end
Ym=abs(Ybus); t = angle(Ybus);
m=2*nbus-ng-2*ns;
maxerror = 1; converge=1;
iter = 0;
% Start of iterations
clear A  DC   J  DX
while maxerror >= accuracy & iter <= maxiter % Test for max. power mismatch
for i=1:m
for k=1:m
   A(i,k)=0;      %Initializing Jacobian matrix
end, end
iter = iter+1;
for n=1:nbus
nn=n-nss(n);
lm=nbus+n-ngs(n)-nss(n)-ns;
J11=0; J22=0; J33=0; J44=0;
   for i=1:nbr
     if nl(i) == n | nr(i) == n
        if nl(i) == n,  l = nr(i); end
        if nr(i) == n,  l = nl(i); end
        J11=J11+ Vm(n)*Vm(l)*Ym(n,l)*sin(t(n,l)- delta(n) + delta(l));
        J33=J33+ Vm(n)*Vm(l)*Ym(n,l)*cos(t(n,l)- delta(n) + delta(l));
        if kb(n)~=1
        J22=J22+ Vm(l)*Ym(n,l)*cos(t(n,l)- delta(n) + delta(l));
        J44=J44+ Vm(l)*Ym(n,l)*sin(t(n,l)- delta(n) + delta(l));
        else, end
        if kb(n) ~= 1  & kb(l) ~=1
        lk = nbus+l-ngs(l)-nss(l)-ns;
        ll = l -nss(l);
      % off diagonalelements of J1
        A(nn, ll) =-Vm(n)*Vm(l)*Ym(n,l)*sin(t(n,l)- delta(n) + delta(l));
              if kb(l) == 0  % off diagonal elements of J2
              A(nn, lk) =Vm(n)*Ym(n,l)*cos(t(n,l)- delta(n) + delta(l));end
```

```
              if kb(n) == 0  % off diagonal elements of J3
              A(lm, ll) =-Vm(n)*Vm(l)*Ym(n,l)*cos(t(n,l)- delta(n)+delta(l));
end
              if kb(n) == 0 & kb(l) == 0  % off diagonal elements of  J4
              A(lm, lk) =-Vm(n)*Ym(n,l)*sin(t(n,l)- delta(n) + delta(l));end
        else end
     else , end
   end
   Pk = Vm(n)^2*Ym(n,n)*cos(t(n,n))+J33;
   Qk = -Vm(n)^2*Ym(n,n)*sin(t(n,n))-J11;
   if kb(n) == 1 P(n)=Pk; Q(n) = Qk; end   % Swing bus P
     if kb(n) == 2  Q(n)=Qk;
         if Qmax(n) ~= 0
           Qgc = Q(n)*basemva + Qd(n) - Qsh(n);
           if iter <= 7                  % Between the 2th & 6th iterations
              if iter > 2                % the Mvar of generator buses are
                if Qgc  < Qmin(n),       % tested. If not within limits Vm(n)
                Vm(n) = Vm(n) + 0.01;    % is changed in steps of 0.01 pu to
                elseif Qgc  > Qmax(n),   % bring the generator Mvar within
                Vm(n) = Vm(n) - 0.01;end % the specified limits.
              else, end
           else,end
         else,end
     end
   if kb(n) ~= 1
     A(nn,nn) = J11;  %diagonal elements of J1
     DC(nn) = P(n)-Pk;
   end
   if kb(n) == 0
     A(nn,lm) = 2*Vm(n)*Ym(n,n)*cos(t(n,n))+J22;  %diagonal elements of J2
     A(lm,nn)= J33;         %diagonal elements of J3
     A(lm,lm) =-2*Vm(n)*Ym(n,n)*sin(t(n,n))-J44;  %diagonal of elements of J4
     DC(lm) = Q(n)-Qk;
   end
end
DX=A\DC';
for n=1:nbus
  nn=n-nss(n);
  lm=nbus+n-ngs(n)-nss(n)-ns;
    if kb(n) ~= 1
    delta(n) = delta(n)+DX(nn); end
    if kb(n) == 0
    Vm(n)=Vm(n)+DX(lm); end
 end
  maxerror=max(abs(DC));
end
V = Vm.*cos(delta)+j*Vm.*sin(delta);
deltad=180/pi*delta;
i=sqrt(-1);
k=0;
for n = 1:nbus
     if kb(n) == 1
     k=k+1;
     S(n)= P(n)+j*Q(n);
     Pg(n) = P(n)*basemva + Pd(n);
     Qg(n) = Q(n)*basemva + Qd(n) - Qsh(n);
     Pgg(k)=Pg(n);
```

```
    Qgg(k)=Qg(n);     %june 97
    elseif  kb(n) ==2
    k=k+1;
    S(n)=P(n)+j*Q(n);
    Qg(n) = Q(n)*basemva + Qd(n) - Qsh(n);
    Pgg(k)=Pg(n);
    Qgg(k)=Qg(n);  % June 1997
  end
yload(n) = (Pd(n)- j*Qd(n)+j*Qsh(n))/(basemva*Vm(n)^2);
end
busdata(:,3)=Vm'; busdata(:,4)=deltad';
Pgt = sum(Pg);  Qgt = sum(Qg); Pdt = sum(Pd); Qdt = sum(Qd); Qsht = sum(Qsh);
Pgg=abs(Pgg);
  vv=abs(V);
```

pso_Trelea_vectorized.m

```
% pso_Trelea_vectorized.m
% a generic particle swarm optimizer
% to find the minimum or maximum of any
% MISO matlab function
%
% Implements Common, Trelea type 1 and 2, and Clerc's class 1". It will
% also automatically try to track to a changing environment (with varied
% success - BKB 3/18/05)
%
% This vectorized version removes the for loop associated with particle
% number. It also *requires* that the cost function have a single input
% that represents all dimensions of search (i.e., for a function that has 2
% inputs then make a wrapper that passes a matrix of ps x 2 as a single
% variable)
%
% Usage:
%  [optOUT]=PSO(functname,D)
% or:
%  [optOUT,tr,te]=...
%        PSO(functname,D,mv,VarRange,minmax,PSOparams,plotfcn,PSOseedValue)
%
% Inputs:
%    functname - string of matlab function to optimize
%    D - # of inputs to the function (dimension of problem)
%
% Optional Inputs:
%    mv - max particle velocity, either a scalar or a vector of length D
%           (this allows each component to have it's own max velocity),
%           default = 4, set if not input or input as NaN
%
%    VarRange - matrix of ranges for each input variable,
%      default -100 to 100, of form:
%       [ min1 max1
%         min2 max2
%            ...
%         minD maxD ]
%
```

```
%    minmax = 0, funct minimized (default)
%           = 1, funct maximized
%           = 2, funct is targeted to P(12) (minimizes distance to errgoal)
%    PSOparams - PSO parameters
%      P(1) - Epochs between updating display, default = 100. if 0,
%             no display
%      P(2) - Maximum number of iterations (epochs) to train, default = 2000.
%      P(3) - population size, default = 24
%
%      P(4) - acceleration const 1 (local best influence), default = 2
%      P(5) - acceleration const 2 (global best influence), default = 2
%      P(6) - Initial inertia weight, default = 0.9
%      P(7) - Final inertia weight, default = 0.4
%      P(8) - Epoch when inertial weight at final value, default = 1500
%      P(9)- minimum global error gradient,
%                  if abs(Gbest(i+1)-Gbest(i)) < gradient over
%                  certain length of epochs, terminate run, default = 1e-25
%      P(10)- epochs before error gradient criterion terminates run,
%                  default = 150, if the SSE does not change over 250 epochs
%                               then exit
%      P(11)- error goal, if NaN then unconstrained min or max, default=NaN
%      P(12)- type flag (which kind of PSO to use)
%                  0 = Common PSO w/intertia (default)
%                  1,2 = Trelea types 1,2
%                  3   = Clerc's Constricted PSO, Type 1"
%      P(13)- PSOseed, default=0
%                = 0 for initial positions all random
%                = 1 for initial particles as user input
%
%    plotfcn - optional name of plotting function, default 'goplotpso',
%              make your own and put here
%
%    PSOseedValue - initial particle position, depends on P(13), must be
%                   set if P(13) is 1 or 2, not used for P(13)=0, needs to
%                   be nXm where n<=ps, and m<=D
%                   If n<ps and/or m<D then remaining values are set random
%                   on Varrange
% Outputs:
%    optOUT - optimal inputs and associated min/max output of function, of
form:
%         [ bestin1
%           bestin2
%             ...
%           bestinD
%           bestOUT ]
%
% Optional Outputs:
%    tr    - Gbest at every iteration, traces flight of swarm
%    te    - epochs to train, returned as a vector 1:endepoch
%
% Example:  out=pso_Trelea_vectorized('f6',2)

% Brian Birge
% Rev 3.3
% 2/18/06

function [OUT,varargout]=pso_Trelea_vectorized(functname,D,varargin)
```

```
rand('state',sum(100*clock));
if nargin < 2
   error('Not enough arguments.');
end

% PSO PARAMETERS
if nargin == 2      % only specified functname and D
   VRmin=ones(D,1)*-100;
   VRmax=ones(D,1)*100;
   VR=[VRmin,VRmax];
   minmax = 0;
   P = [];
   mv = 4;
   plotfcn='goplotpso';
elseif nargin == 3  % specified functname, D, and mv
   VRmin=ones(D,1)*-100;
   VRmax=ones(D,1)*100;
   VR=[VRmin,VRmax];
   minmax = 0;
   mv=varargin{1};
   if isnan(mv)
       mv=4;
   end
   P = [];
   plotfcn='goplotpso';
elseif nargin == 4  % specified functname, D, mv, Varrange
   mv=varargin{1};
   if isnan(mv)
       mv=4;
   end
   VR=varargin{2};
   minmax = 0;
   P = [];
   plotfcn='goplotpso';
elseif nargin == 5  % Functname, D, mv, Varrange, and minmax
   mv=varargin{1};
   if isnan(mv)
       mv=4;
   end
   VR=varargin{2};
   minmax=varargin{3};
   P = [];
   plotfcn='goplotpso';
elseif nargin == 6  % Functname, D, mv, Varrange, minmax, and psoparams
   mv=varargin{1};
   if isnan(mv)
       mv=4;
   end
   VR=varargin{2};
   minmax=varargin{3};
   P = varargin{4}; % psoparams
   plotfcn='goplotpso';
elseif nargin == 7  % Functname, D, mv, Varrange, minmax, and psoparams,
plotfcn
   mv=varargin{1};
   if isnan(mv)
```

```
        mv=4;
    end
    VR=varargin{2};
    minmax=varargin{3};
    P = varargin{4}; % psoparams
    plotfcn = varargin{5};
elseif nargin == 8  % Functname, D, mv, Varrange, minmax, and psoparams,
plotfcn, PSOseedValue
    mv=varargin{1};
    if isnan(mv)
        mv=4;
    end
    VR=varargin{2};
    minmax=varargin{3};
    P = varargin{4}; % psoparams
    plotfcn = varargin{5};
    PSOseedValue = varargin{6};
else
    error('Wrong # of input arguments.');
end

% sets up default pso params
Pdef = [100 2000 24 2 2 0.9 0.4 1500 1e-25 250 NaN 0 0];
Plen = length(P);
P    = [P,Pdef(Plen+1:end)];

df      = P(1);
me      = P(2);
ps      = P(3);
ac1     = P(4);
ac2     = P(5);
iw1     = P(6);
iw2     = P(7);
iwe     = P(8);
ergrd   = P(9);
ergrdep = P(10);
errgoal = P(11);
trelea  = P(12);
PSOseed = P(13);

% used with trainpso, for neural net training
if strcmp(functname,'pso_neteval')
    net = evalin('caller','net');
     Pd = evalin('caller','Pd');
     Tl = evalin('caller','Tl');
     Ai = evalin('caller','Ai');
      Q = evalin('caller','Q');
     TS = evalin('caller','TS');
end


% error checking
 if ((minmax==2) & isnan(errgoal))
     error('minmax= 2, errgoal= NaN: choose an error goal or set minmax to 0
or 1');
 end
```

```
 if ( (PSOseed==1) & ~exist('PSOseedValue') )
     error('PSOseed flag set but no PSOseedValue was input');
 end

 if exist('PSOseedValue')
     tmpsz=size(PSOseedValue);
     if D < tmpsz(2)
         error('PSOseedValue column size must be D or less');
     end
     if ps < tmpsz(1)
         error('PSOseedValue row length must be # of particles or less');
     end
 end

% set plotting flag
if (P(1))~=0
  plotflg=1;
else
  plotflg=0;
end

% preallocate variables for speed up
 tr = ones(1,me)*NaN;

% take care of setting max velocity and position params here
if length(mv)==1
 velmaskmin = -mv*ones(ps,D);      % min vel, psXD matrix
 velmaskmax = mv*ones(ps,D);       % max vel
elseif length(mv)==D
 velmaskmin = repmat(forcerow(-mv),ps,1); % min vel
 velmaskmax = repmat(forcerow( mv),ps,1); % max vel
else
 error('Max vel must be either a scalar or same length as prob dimension D');
end
posmaskmin  = repmat(VR(1:D,1)',ps,1);  % min pos, psXD matrix
posmaskmax  = repmat(VR(1:D,2)',ps,1);  % max pos
posmaskmeth = 3; % 3=bounce method (see comments below inside epoch loop)

% PLOTTING
 message = sprintf('PSO: %%g/%g iterations, GBest = %%20.20g.\n',me);

% INITIALIZE INITIALIZE INITIALIZE INITIALIZE INITIALIZE INITIALIZE

% initialize population of particles and their velocities at time zero,
% format of pos= (particle#, dimension)
 % construct random population positions bounded by VR
  pos(1:ps,1:D) = normmat(rand([ps,D]),VR',1);

  if PSOseed == 1         % initial positions user input, see comments above
    tmpsz                       = size(PSOseedValue);
    pos(1:tmpsz(1),1:tmpsz(2)) = PSOseedValue;
  end

 % construct initial random velocities between -mv,mv
  vel(1:ps,1:D) = normmat(rand([ps,D]),...
      [forcecol(-mv),forcecol(mv)]',1);
```

```
% initial pbest positions vals
 pbest = pos;

% VECTORIZE THIS, or at least vectorize cost funct call
 out = feval(functname,pos);   % returns column of cost values (1 for each
particle)
%---------------------------

 pbestval=out;   % initially, pbest is same as pos

% assign initial gbest here also (gbest and gbestval)
 if minmax==1
   % this picks gbestval when we want to maximize the function
    [gbestval,idx1] = max(pbestval);
 elseif minmax==0
   % this works for straight minimization
    [gbestval,idx1] = min(pbestval);
 elseif minmax==2
   % this works when you know target but not direction you need to go
   % good for a cost function that returns distance to target that can be either
   % negative or positive (direction info)
    [temp,idx1] = min((pbestval-ones(size(pbestval))*errgoal).^2);
    gbestval    = pbestval(idx1);
 end

 % preallocate a variable to keep track of gbest for all iters
 bestpos        = zeros(me,D+1)*NaN;
 gbest          = pbest(idx1,:);  % this is gbest position
   % used with trainpso, for neural net training
   % assign gbest to net at each iteration, these interim assignments
   % are for plotting mostly
    if strcmp(functname,'pso_neteval')
        net=setx(net,gbest);
    end
 %tr(1)          = gbestval;       % save for output
 bestpos(1,1:D) = gbest;

% this part used for implementing Carlisle and Dozier's APSO idea
% slightly modified, this tracks the global best as the sentry whereas
% their's chooses a different point to act as sentry
% see "Tracking Changing Extremea with Adaptive Particle Swarm Optimizer",
% part of the WAC 2002 Proceedings, June 9-13, http://wacong.com
 sentryval = gbestval;
 sentry    = gbest;

if (trelea == 3)
% calculate Clerc's constriction coefficient chi to use in his form
 kappa   = 1; % standard val = 1, change for more or less constriction
 if ( (ac1+ac2) <=4 )
     chi = kappa;
 else
     psi     = ac1 + ac2;
     chi_den = abs(2-psi-sqrt(psi^2 - 4*psi));
     chi_num = 2*kappa;
     chi     = chi_num/chi_den;
 end
end
```

```
% INITIALIZE END INITIALIZE END INITIALIZE END INITIALIZE END
rstflg = 0; % for dynamic environment checking
% start PSO iterative procedures
 cnt    = 0; % counter used for updating display according to df in the
options
 cnt2   = 0; % counter used for the stopping subroutine based on error
convergence
 iwt(1) = iw1;
for i=1:me  % start epoch loop (iterations)

     out        = feval(functname,[pos;gbest]);
     outbestval = out(end,:);
     out        = out(1:end-1,:);

     tr(i+1)          = gbestval; % keep track of global best val
     te               = i; % returns epoch number to calling program when
done
     bestpos(i,1:D+1) = [gbest,gbestval];

     %assignin('base','bestpos',bestpos(i,1:D+1));
   %--------------------------------------------------------------------
   % this section does the plots during iterations
    if plotflg==1
      if (rem(i,df) == 0 ) | (i==me) | (i==1)
         fprintf(message,i,gbestval);
         cnt = cnt+1; % count how many times we display (useful for movies)

         eval(plotfcn); % defined at top of script

      end  % end update display every df if statement
    end % end plotflg if statement

    % check for an error space that changes wrt time/iter
    % threshold value that determines dynamic environment
    % sees if the value of gbest changes more than some threshold value
    % for the same location
    chkdyn = 1;
    rstflg = 0; % for dynamic environment checking

    if chkdyn==1
     threshld = 0.05;  % percent current best is allowed to change, .05 = 5%
etc
     letiter  = 5; % # of iterations before checking environment, leave at
least 3 so PSO has time to converge
     outorng  = abs( 1- (outbestval/gbestval) ) >= threshld;
     samepos  = (max( sentry == gbest ));

     if (outorng & samepos) & rem(i,letiter)==0
         rstflg=1;
       % disp('New Environment: reset pbest, gbest, and vel');
       %% reset pbest and pbestval if warranted
%        outpbestval = feval( functname,[pbest] );
%        Poutorng    = abs( 1-(outpbestval./pbestval) ) > threshld;
%        pbestval    = pbestval.*~Poutorng + outpbestval.*Poutorng;
%        pbest       = pbest.*repmat(~Poutorng,1,D) +
pos.*repmat(Poutorng,1,D);
```

```
        pbest     = pos; % reset personal bests to current positions
        pbestval  = out;
        vel       = vel*10; % agitate particles a little (or a lot)

       % recalculate best vals
        if minmax == 1
           [gbestval,idx1] = max(pbestval);
        elseif minmax==0
           [gbestval,idx1] = min(pbestval);
        elseif minmax==2 % this section needs work
           [temp,idx1] = min((pbestval-ones(size(pbestval))*errgoal).^2);
           gbestval    = pbestval(idx1);
        end

        gbest  = pbest(idx1,:);

        % used with trainpso, for neural net training
        % assign gbest to net at each iteration, these interim assignments
        % are for plotting mostly
        if strcmp(functname,'pso_neteval')
           net=setx(net,gbest);
        end
     end  % end if outorng

     sentryval = gbestval;
     sentry    = gbest;

    end % end if chkdyn

    % find particles where we have new pbest, depending on minmax choice
    % then find gbest and gbestval
     %[size(out),size(pbestval)]
    if rstflg == 0
     if minmax == 0
        [tempi]            = find(pbestval>=out); % new min pbestvals
        pbestval(tempi,1)  = out(tempi);   % update pbestvals
        pbest(tempi,:)     = pos(tempi,:); % update pbest positions

        [iterbestval,idx1] = min(pbestval);

        if gbestval >= iterbestval
            gbestval = iterbestval;
            gbest    = pbest(idx1,:);
            % used with trainpso, for neural net training
            % assign gbest to net at each iteration, these interim
assignments
            % are for plotting mostly
             if strcmp(functname,'pso_neteval')
                net=setx(net,gbest);
             end
        end
     elseif minmax == 1
        [tempi,dum]        = find(pbestval<=out); % new max pbestvals
        pbestval(tempi,1)  = out(tempi,1); % update pbestvals
        pbest(tempi,:)     = pos(tempi,:); % update pbest positions
```

```
        [iterbestval,idx1] = max(pbestval);
        if gbestval <= iterbestval
            gbestval = iterbestval;
            gbest    = pbest(idx1,:);
            % used with trainpso, for neural net training
            % assign gbest to net at each iteration, these interim
assignments
            % are for plotting mostly
             if strcmp(functname,'pso_neteval')
                net=setx(net,gbest);
             end
        end
     elseif minmax == 2  % this won't work as it is, fix it later
        egones            = errgoal*ones(ps,1); % vector of errgoals
        sqrerr2           = ((pbestval-egones).^2);
        sqrerr1           = ((out-egones).^2);
        [tempi,dum]       = find(sqerr1 <= sqrerr2); % find particles closest
to targ
        pbestval(tempi,1) = out(tempi,1); % update pbestvals
        pbest(tempi,:)    = pos(tempi,:); % update pbest positions

        sqrerr            = ((pbestval-egones).^2); % need to do this to
reflect new pbests
        [temp,idx1]       = min(sqrerr);
        iterbestval       = pbestval(idx1);

        if (iterbestval-errgoal)^2 <= (gbestval-errgoal)^2
           gbestval = iterbestval;
           gbest    = pbest(idx1,:);
           % used with trainpso, for neural net training
            % assign gbest to net at each iteration, these interim
assignments
            % are for plotting mostly
             if strcmp(functname,'pso_neteval')
                net=setx(net,gbest);
             end
        end
     end
    end


 %   % build a simple predictor 10th order, for gbest trajectory
 %   if i>500
 %    for dimcnt=1:D
 %      pred_coef  = polyfit(i-250:i,(bestpos(i-250:i,dimcnt))',20);
 %     % pred_coef  = polyfit(200:i,(bestpos(200:i,dimcnt))',20);
 %      gbest_pred(i,dimcnt) = polyval(pred_coef,i+1);
 %    end
 %    else
%       gbest_pred(i,:) = zeros(size(gbest));
%    end

   %gbest_pred(i,:)=gbest;
   %assignin('base','gbest_pred',gbest_pred);

 %   % convert to non-inertial frame
 %    gbestoffset = gbest - gbest_pred(i,:);
```

```
 %    gbest = gbest - gbestoffset;
 %    pos   = pos + repmat(gbestoffset,ps,1);
 %    pbest = pbest + repmat(gbestoffset,ps,1);


%PSOPSOPSOPSOPSOPSOPSOPSOPSOPSOPSOPSOPSOPSOPSOPSOPSOPSOPSOPSOPSOPSOPSOPSOPSOPSO

      % get new velocities, positions (this is the heart of the PSO
algorithm)
      % each epoch get new set of random numbers
       rannum1 = rand([ps,D]); % for Trelea and Clerc types
       rannum2 = rand([ps,D]);
       if     trelea == 2
        % from Trelea's paper, parameter set 2
         vel = 0.729.*vel...                                % prev vel
               +1.494.*rannum1.*(pbest-pos)...              % independent
               +1.494.*rannum2.*(repmat(gbest,ps,1)-pos); % social
       elseif trelea == 1
        % from Trelea's paper, parameter set 1
         vel = 0.600.*vel...                                % prev vel
               +1.700.*rannum1.*(pbest-pos)...              % independent
               +1.700.*rannum2.*(repmat(gbest,ps,1)-pos); % social
       elseif trelea ==3
        % Clerc's Type 1" PSO
         vel = chi*(vel...                                  % prev vel
               +ac1.*rannum1.*(pbest-pos)...                % independent
               +ac2.*rannum2.*(repmat(gbest,ps,1)-pos)) ; % social
       else
        % common PSO algo with inertia wt
        % get inertia weight, just a linear funct w.r.t. epoch parameter iwe
         if i<=iwe
            iwt(i) = ((iw2-iw1)/(iwe-1))*(i-1)+iw1;
         else
            iwt(i) = iw2;
         end
        % random number including acceleration constants
         ac11 = rannum1.*ac1;    % for common PSO w/inertia
         ac22 = rannum2.*ac2;

         vel = iwt(i).*vel...                               % prev vel
               +ac11.*(pbest-pos)...                        % independent
               +ac22.*(repmat(gbest,ps,1)-pos);             % social
       end

       % limit velocities here using masking
        vel = ( (vel <= velmaskmin).*velmaskmin ) + ( (vel > velmaskmin).*vel
);
        vel = ( (vel >= velmaskmax).*velmaskmax ) + ( (vel < velmaskmax).*vel
);

       % update new position (PSO algo)
        pos = pos + vel;

       % position masking, limits positions to desired search space
       % method: 0) no position limiting, 1) saturation at limit,
       %         2) wraparound at limit , 3) bounce off limit
        minposmask_throwaway = pos <= posmaskmin;  % these are psXD matrices
```

```
        minposmask_keep      = pos >  posmaskmin;
        maxposmask_throwaway = pos >= posmaskmax;
        maxposmask_keep      = pos <  posmaskmax;

        if     posmaskmeth == 1
         % this is the saturation method
          pos = ( minposmask_throwaway.*posmaskmin ) + ( minposmask_keep.*pos
);
          pos = ( maxposmask_throwaway.*posmaskmax ) + ( maxposmask_keep.*pos
);
        elseif posmaskmeth == 2
         % this is the wraparound method
          pos = ( minposmask_throwaway.*posmaskmax ) + ( minposmask_keep.*pos
);
          pos = ( maxposmask_throwaway.*posmaskmin ) + ( maxposmask_keep.*pos
);
        elseif posmaskmeth == 3
         % this is the bounce method, particles bounce off the boundaries
with -vel
          pos = ( minposmask_throwaway.*posmaskmin ) + ( minposmask_keep.*pos
);
          pos = ( maxposmask_throwaway.*posmaskmax ) + ( maxposmask_keep.*pos
);

          vel = (vel.*minposmask_keep) + (-vel.*minposmask_throwaway);
          vel = (vel.*maxposmask_keep) + (-vel.*maxposmask_throwaway);
        else
         % no change, this is the original Eberhart, Kennedy method,
         % it lets the particles grow beyond bounds if psoparams (P)
         % especially Vmax, aren't set correctly, see the literature
        end


%PSOPSOPSOPSOPSOPSOPSOPSOPSOPSOPSOPSOPSOPSOPSOPSOPSOPSOPSOPSOPSOPSOPSOPSOPSOPSO
% check for stopping criterion based on speed of convergence to desired
   % error
    tmp1 = abs(tr(i) - gbestval);
    if tmp1 > ergrd
       cnt2 = 0;
    elseif tmp1 <= ergrd
       cnt2 = cnt2+1;
       if cnt2 >= ergrdep
         if plotflg == 1
          fprintf(message,i,gbestval);
          disp(' ');
          disp(['--> Solution likely, GBest hasn''t changed by at least ',...
              num2str(ergrd),' for ',...
                  num2str(cnt2),' epochs.']);
          eval(plotfcn);
         end
         break
       end
    end

   % this stops if using constrained optimization and goal is reached
    if ~isnan(errgoal)
```

```
    if ((gbestval<=errgoal) & (minmax==0)) | ((gbestval>=errgoal) &
(minmax==1))

        if plotflg == 1
            fprintf(message,i,gbestval);
            disp(' ');
            disp(['--> Error Goal reached, successful termination!']);

            eval(plotfcn);
        end
        break
    end

   % this is stopping criterion for constrained from both sides
    if minmax == 2
      if ((tr(i)<errgoal) & (gbestval>=errgoal)) | ((tr(i)>errgoal) ...
              & (gbestval <= errgoal))
        if plotflg == 1
            fprintf(message,i,gbestval);
            disp(' ');
            disp(['--> Error Goal reached, successful termination!']);

            eval(plotfcn);
        end
        break
      end
    end % end if minmax==2
   end  % end ~isnan if

 %    % convert back to inertial frame
 %     pos = pos - repmat(gbestoffset,ps,1);
 %     pbest = pbest - repmat(gbestoffset,ps,1);
 %     gbest = gbest + gbestoffset;


end  % end epoch loop

%% clear temp outputs
% evalin('base','clear temp_pso_out temp_te temp_tr;');

% output & return
 OUT=[gbest';gbestval];
 varargout{1}=[1:te];
 varargout{2}=[tr(find(~isnan(tr)))];

 return
```

psoopf.m

```
clear;
clc;

   %        IEEE 30-BUS TEST SYSTEM (American Electric Power)
%        Bus Bus  Voltage Angle   ---Load---- -------Generator----- Injected
%        No  code Mag.    Degree  MW    Mvar  MW  Mvar Qmin Qmax     Mvar
 global busdata linedata gencost
basemva=100;
busdata=[1   1    1.06    0.0     0.0   0.0    0.0  0.0   0   0        0
```

```
 2   2    1.043   0.0   21.70  12.7   40.0  0.0 -40  50       0
 3   0    1.0     0.0     2.4   1.2    0.0  0.0   0   0       0
 4   0    1.06    0.0     7.6   1.6    0.0  0.0   0   0       0
 5   2    1.01    0.0    94.2  19.0    0.0  0.0 -40  40       0
 6   0    1.0     0.0     0.0   0.0    0.0  0.0   0   0       0
 7   0    1.0     0.0    22.8  10.9    0.0  0.0   0   0       0
 8   2    1.01    0.0    30.0  30.0    0.0  0.0 -10  60       0
 9   0    1.0     0.0     0.0   0.0    0.0  0.0   0   0       0
10   0    1.0     0.0     5.8   2.0    0.0  0.0  -6  24      19
11   2    1.082   0.0     0.0   0.0    0.0  0.0   0   0       0
12   0    1.0     0       11.2  7.5    0    0     0   0       0
13   2    1.071   0        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       9.0   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       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     4.3
25   0    1       0       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       0
28   0    1       0       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   1.92   0    0     0   0      0];

%                                      Line code
%         Bus bus    R       X     1/2 B   = 1 for lines
%         nl  nr   p.u.    p.u.    p.u.     > 1 or < 1 tr. tap at bus nl
linedata=[1   2   0.0192   0.0575   0.02640    1
         % 1   3   0.0452   0.1852   0.02040    1
          2   4   0.0570   0.1737   0.01840    1
          3   4   0.0132   0.0379   0.00420    1
          2   5   0.0472   0.1983   0.02090    1
          2   6   0.0581   0.1763   0.01870    1
          4   6   0.0119   0.0414   0.00450    1
          5   7   0.0460   0.1160   0.01020    1
          6   7   0.0267   0.0820   0.00850    1
          6   8   0.0120   0.0420   0.00450    1
          6   9   0.0      0.2080   0.0    0.978
          6  10   0         .5560   0      0.969
          9  11   0         .2080   0          1
          9  10   0         .1100   0          1
          4  12   0         .2560   0      0.932
         12  13   0         .1400   0          1
         12  14    .1231    .2559   0          1
         12  15    .0662    .1304   0          1
         12  16    .0945    .1987   0          1
         14  15    .2210    .1997   0          1
         16  17    .0824    .1923   0          1
         15  18    .1073    .2185   0          1
         18  19    .0639    .1292   0          1
         19  20    .0340    .0680   0          1
```

```
         10  20    .0936    .2090   0          1
         10  17    .0324    .0845   0          1
         10  21    .0348    .0749   0          1
         10  22    .0727    .1499   0          1
         21  22    .0116    .0236   0          1
         15  23    .1000    .2020   0          1
         22  24    .1150    .1790   0          1
         23  24    .1320    .2700   0          1
         24  25    .1885    .3292   0          1
         25  26    .2544    .3800   0          1
         25  27    .1093    .2087   0          1
         28  27     0       .3960   0      0.968
         27  29    .2198    .4153   0          1
         27  30    .3202    .6027   0          1
         29  30    .2399    .4533   0          1
          8  28    .0636    .2000   0.0214     1
          6  28    .0169    .0599   0.065      1];
gencost = [1    0.00375 2   0       50 200;
        2 0.0175    1.75    0       20 80;
        5 0.0625    1       0       15 50;
        8 0.277     58.33   388.88  10 100;
        11 0.027    75.83   138.88  10 130;
        13 0.442    74.66   311.11  12 140];
    Pdt=283.4;
 l=gencost(2:6,5)';
u=gencost(2:6,6)';
ran=[l' u'];
nnn=length(gencost(:,1))-1;
Pdef = [20 500 20 2 2 0.9 0.4 1500 1e-6 5000 NaN 0 0];
 [OUT]=pso_Trelea_vectorized('opf2',nnn,1,ran,0,Pdef);
 out=abs(OUT);
 P1=out(1:nnn)
  [F1 f3 P vv]=opf2(P1')
  TL=sum(P)-Pdt
```

Bibliography

1. K. Selvi, T.Meena, N.Ramaraj, "A Generation Rescheduling Method to Alleviate Line Overloads using PSO," *IE (I) Journal-E*L.2005.

2. Y. Xiaodan, J. Hongjie, Z. Jing, L. Yan, Z. Yuan, "*Interface Control Based on Power Flow Tracing and Generator Re-redispatching,"* Automation of Electric Power Systems IEEE, 2008.

3. G.Baskar, M.R. Mohan, "Contingency Constrained Economic Load Dispatch using Improved Particle Swarm Optimization for Security Enhancement," *Electric Power System Research Elsevier,* 2008.
    a.

4. E.Muneender, M.D. Vinod Kumar, "Optimal Rescheduling of real and reactive powers of generators for zonal Congestion Management Based on FDR PSO.*IEEE T&D Asia*, 2009

5. Sujatha Balaraman, K.Kamaraj, "Congestion management in Deregulated power system using real coded genetic algorithm," *International Journal of Engineering Science and Technology* , vol 2, no.11, ,pp. 6681-6690, 2010

6. S. Balaraman , K. Kamaraj, "Application of Differential Evolution for Congestion management in power system," *Modern Applied Science* ,vol 4, no 8, August 2010.

7. Z. Jinli, J. Hongjie, Y. Xiaodan, "*Voltage Stability Control Based on real power flow tracing,* Proceedings of CSEE, IEEE,2009.

8. X. Zou, X. Luo, Z. Peng, "*Congestion Management Ensuing Voltage Stability under Multicontingency with preventive and Corrective Controls,"* IEEE, 2009.

9. Hwa-Sik Choi, Seung II Moon, "*A new Operation of series compensating device under Line Flow Congestion using the Linear zed Line Flow sensitivity*," Power Engineering Society winter meeting IEEE,2001.

10. E.M. Yap, M.Al-Dabbagh, P.C. Thum, "*UPFC Controller in Mitigating Line Congestion for Cost-Efficient Power Delivery,"*Power Engineering Conference IPEC, IEEE, 2006.

11. Xiao-Ping Zhang, L. Yao "*A Vision of Electricity network Congestion Management with FACTS and HVDC,"* DRPT2008, 6-9 April, 2008 Nanjing China

12. G. M. Huang, N.l Kumar, C Nair, "*An OPF based Algorithm to Evaluate Load Curtailment Incorporating Voltage Stability Margin Criteria*," Power Engineering Society Winter Meeting ,IEEE, 2002

13. F. HE, Y.WANG, K. CHAN, Y. ZHANG, S. MEI, "*Optimal Load Shedding Strategy Based on Particle Swarm Optimisation*," 8th international conference on Advances in Power System Control operation and Management APSCOM 2009.

14. R.N.Nayak, Y.K. Sehgal, S.Sen, "*EHV Transmission Line Capacity Enhancement through Increase in Surge Impedance Loading Level,"* Power India Conference, 2006

15. K.P. Basu, "*Power transfer Capability of Transmission Line Limited by voltage Stability: Simple Analytical Expressions*," IEEE Power Engineering Review, September 2000.

16. J.Ma, Y.H.Song,Q.Lu, S.Mei, "Framework for dynamic congestion Management in open power markets," *IEE Proc.Gener.Transm. Distrib.* Vol.149,No.2 March 2002

17. J. Kennedy, R. Eberhart, "Particle Swarm Optimisation," in the Proceedings of IEEE International Conference on Neural Networks , pp. 1942-1948, 1995.

18. R. Storn and K. Price, "Differential Evolution- A simple and Efficient Heuristic for Global Optimisation For Continueous Spaces," Journal of Global Optimisation, vol. 11, no. 4,pp. 341-359, 1997.

19. G. Venter and J. Sobieszczanski-Sobieski, "Particle Swarm Optimization," AIAA Journal, vol. 41, no. 8, pp. 1583-1589, 2003.

20. X . Yao, Liu Yand Lin G, "Evolutionary Programming Made Faster," IEEE Transation on Evolutionary Computation, vol. 3, no. 2, 82-102, 1999.

21. Y. Shi and RC Eberhart, " Parameter Selection in Particle Swarm Optimisation," Evolutionary Programming VII, Springer, Lecture Notes in Computer Science1447,591-600, 1998.

22. P. Dimitri, Nonlinear Programming (Second ed.). Cambridge, MA.: Athena Scientific.

23. Vapnyarskii, I.B., "Lagrange multipliers"Vapnyarskii, I.B., "Lagrange multipliers", in Hazewinkel, Michiel, Encyclopaedia of MathematicsVapnyarskii, I.B., "Lagrange multipliers", in Hazewinkel, Michiel, Encyclopaedia of Mathematics, Springer, http://eom.springer.de/L/l057190.htm , 2001

24. Lasdon, Leon S. Optimization theory for large systems. Macmillan series in operations research. New York: The Macmillan Company. pp.1970

25. Lasdon, Leon S. Optimization theory for large systems (reprint of the 1970 Macmillan ed.). Mineola, New York: Dover Publications, Inc. 2002

26. Hiriart-Urruty, Jean-Baptiste; Lemaréchal, Claude "XII Abstract duality for practitioners". Convex analysis and minimization algorithms, Volume II: Advanced theory and bundle methods. Grundlehren der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences]. 306. Berlin: Springer-Verlag. pp. 136–193 (and Bibliographical comments on pp. 334–335). 1993

27. Lemaréchal, Claude "Lagrangian relaxation". In Michael Jünger and Denis Naddef. Computational combinatorial optimization: Papers from the Spring School held in Schloß 2001

28. Bullis, K. 2014. "Smart Wind and Solar Power." MIT Technology Review. Accessed May 2015, http://www.technologyreview.com/featuredstory/526541/smart-wind-and-solar-power/.

29. California Independent System Operator (CAISO). "Briefing on the CAISO Renewable Integration Study." October 17, 2007.

30. California Public Utilities Commission (CPUC). "Rule 21." Last Modified March 19, 2015. http://www.cpuc.ca.gov/PUC/energy/rule21.htm.

31. Clover, Ian. 2015. "California adopts pioneering advanced inverter standards." PV Magazine, January 8. Accessed April 2015, http://www.pvmagazine.com/news/details/beitrag/californiaadopts-pioneering-advanced-inverter-standards_100017709/#axzz3ZTA3QFHL.

32. Grid Innovation Online. "SGMS- The integration of renewables in distribution networks." Undated. Accessed May 2015, http://www.gridinnovation-on-line.eu/Articles/Library/SGMS--The-Integration-Of-Renewables-In-Distribution-Networks.kl#sthash.7lfjnp6i.dpuf.

33. International Renewable Energy Agency (IRENA) (2013). "Smart Grids and Renewables: A Guide for Effective Deployment." Accessed May 19, 2015, https://www.irena.org/DocumentDownloads/Publications/smart_grids.pdf.

34. International Smart Grid Action Network (ISGAN). ISGAN Smart Grid Glossary. Accessed May 20, 2015: http://en.openei.org/wiki/ISGAN_Smart_Grid_Glossary.

35. Kupzog, Friederich, Helfried Brunner, Johann Schrammel, Susen Döbelt, Alfred Einfalt, Andreas Lugmaier, Mike Pichler, Daniel Reiter, Hans Jürgen Bacher, Laura Emmermacher, Marietta Stutz, Markus Berger, Thomas Rieder, Herwig Struber, Bernhard Kaiser, Georg Kienesberger, and Wolfgang Prüggler. 2013. Results & Findings from the Smart Grids Model Region Salzburg. Salzburg, Austria: Salzburg AG. Accessed May 4, 2015, http://www.smartgridssalzburg.at/fileadmin/user_upload/downloads/SGMS_Results_Findings_0 5-2013.pdf.

36. Mills, Andrew, and Ryan Wiser. 2010. Implications of Wide-Area Geographic Diversity for Short-Term Variability of Solar Power. LBNL-3884E. Berkeley, CA: Lawrence Berkeley National Laboratory. Accessed May 2015, http://emp.lbl.gov/sites/all/files/REPORT%20lbnl3884e.pdf.

37. Ormond, Jamie. 2014. "California's Interconnection Rule 21 and Smart Inverter & Smart Inverter Working Group." Presentation to the New England Independent System Operator, July 11, 2014. Accessed May 2015, http://www.isone.com/committees/comm_wkgrps/prtcpnts_comm/pac/mtrls/2014/jul112014/california_smart_i nverter_working_group.pdf.

38. Swedish Coordination Council for Smart Grid. "Samordningsrådet för smarta elnät, Planera för effekt!, SOU 2014:84, Chapter 5.4.1, Page 256. 2014.

39. Widegren, Karin, Energimarknadsinspektionen. Personal communications with E.ON, March–May 2015.

40. Loktak power plant, Nhpc. http://www.nhpcindia.com

41. Tata Power Plant, Kalinganagar, Odisha. https://www.tatapower.com/plants-projects/waste-heat-recovery-generation/iel-kalinganagar-135mw.aspx

42. Solar Park by NHPC in Odisha. http://nhpcindia.com/

43. Renewable Energy Towards Smart Grid: Select Proceedings of SGESC 2021 by S. C. Srivastava, Ashwani Kumar, S.N. Singh

44. Design of Smart Power Grid Renewable Energy Systems by Ali Keyhani.

45. Optimization and Security Challenges in Smart Power Grids by Panos M. Pardalos, Marco Carvalho, Vijay Pappu.

46. Integration of Renewable energy sources with smart grid by M.Kathiresh, A. Mehaboob Subahani and G.R. Kanagachidambaresan.