

# UNIVERSITI MALAYSIA PAHANG

## BORANG PENGESAHAN STATUS TESIS♦

JUDUL: APPLICATION OF PSO TECHNIQUE FOR OPTIMAL  
LOCATION OF FACTS DEVICES  
SESI PENGAJIAN: 2009/2010

Saya SUHAIRAH BINTI RAZALI ( 860616-46-5464 )  
(HURUF BESAR)

mengaku membenarkan tesis (Sarjana Muda/~~Sarjana~~ /~~Doktor Falsafah~~)\* ini disimpan di Perpustakaan dengan syarat-syarat kegunaan seperti berikut:

1. Tesis adalah hakmilik Universiti Malaysia Pahang (UMP).
2. Perpustakaan dibenarkan membuat salinan untuk tujuan pengajian sahaja.
3. Perpustakaan dibenarkan membuat salinan tesis ini sebagai bahan pertukaran antara institusi pengajian tinggi.
4. \*\*Sila tandakan ( √ )

☐

**SULIT**

(Mengandungi maklumat yang berdarjah keselamatan atau kepentingan Malaysia seperti yang termaktub di dalam AKTA RAHSIA RASMI 1972)

☐

**TERHAD**

(Mengandungi maklumat TERHAD yang telah ditentukan oleh organisasi/badan di mana penyelidikan dijalankan)

☒

**TIDAK TERHAD**

Disahkan oleh:

\_\_\_\_\_  
(TANDATANGAN PENULIS)

\_\_\_\_\_  
(TANDATANGAN PENYELIA)

Alamat Tetap:

NO. 86 LORONG PERDANA 1/1,  
TAMAN PERDANA 26600 PEKAN,  
PAHANG DARUL MAKMUR.

DR. AHMED N. ABD ALLA  
( Nama Penyelia )

Tarikh: 20 NOVEMBER 2009

Tarikh: : 20 NOVEMBER 2009

CATATAN: \* Potong yang tidak berkenaan.  
\*\* Jika tesis ini SULIT atau TERHAD, sila lampirkan surat daripada pihak berkuasa/organisasi berkenaan dengan menyatakan sekali tempoh tesis ini perlu dikelaskan sebagai atau TERHAD.  
♦ Tesis dimaksudkan sebagai tesis bagi Ijazah doktor Falsafah dan Sarjana secara Penyelidikan, atau disertasi bagi pengajian secara kerja kursus dan penyelidikan, atau Laporan Projek Sarjana Muda (PSM).

“I hereby acknowledge that the scope and quality of this thesis is qualified for the  
award of the Bachelor Degree of Electrical Engineering (Electronics)”

Signature : \_\_\_\_\_

Name : DR AHMED N. ABD ALLA

Date : 23 NOVEMBER 2009

“All the trademark and copyrights use herein are property of their respective owner. References of information from other sources are quoted accordingly; otherwise the information presented in this report is solely work of the author.”

Signature : \_\_\_\_\_

Author : SUHAIRAH BINTI RAZALI

Date : 23 NOVEMBER 2009

APPLICATION OF PSO TECHNIQUE FOR OPTIMAL LOCATION OF FACTS  
DEVICES

SUHAIRAH BINTI RAZALI

This thesis is submitted as partial fulfillment of the requirements for the award of the  
Bachelor of Electrical Engineering (Hons.) (Power System)

Faculty of Electrical & Electronics Engineering  
Universiti Malaysia Pahang

NOVEMBER, 2009

“All the trademark and copyrights use herein are property of their respective owner. References of information from other sources are quoted accordingly; otherwise the information presented in this report is solely work of the author.”

Signature : \_\_\_\_\_

Author : SUHAIRAH BINTI RAZALI

Date : 23 NOVEMBER 2009

## DEDICATION

Specially dedicated to  
My beloved parents, sisters, brother and friends  
Thank you for the endless support and encouragement

## ACKNOWLEDGEMENT

In the name of Allah S.W.T the Most Gracious, the Most Merciful. Praise is to Allah, Lord of the Universe and Pence and Prayers be upon His final Prophet and Messenger, Muhammad S.A.W.

First of all, I would like to express my sincere gratitude and appreciation to my supervisor Dr. Ahmed N. Abd Alla for his guidance, encouragement, advice and supports throughout the preparation of this thesis. His influence has helped we learn the practicalities of this project.

Secondly, I would like to sincerely thank the Universiti Malaysia Pahang (UMP) and Fakulti Kejuruteraan Elektrik dan Elektronik (FKEE) for providing good facilities for us in completing our project.

Finally, I would like to thank to my parents, Razali bin Hasan and Che Azizon binti Ab. Rahim, my lovely sisters and brother who had given me their support, encouragement and always pray for my future undertakings. Without their support I doubt it would have been possible for me to complete this study.

## ABSTRACT

The application of Particle Swarm Optimization (PSO) technique to find optimal location of Flexible AC Transmission System (FACTS) devices to achieve maximum system loadability. While finding the optimal location, thermal limit for the lines and voltage limit for the buses are taken as constraints. Two types of FACTS devices, Unified Power Flow Controller (UPFC) and thyristor Controlled Series Compensator (TCSC) are considered. The optimizations are performed on three parameters namely the location of FACTS devices, their setting and their type. Simulations are performed on IEEE 30-bus system for optimal location of FACTS devices and the results obtained are encouraging and will be useful in electrical restructuring.



## ABSTRAK

Aplikasi bagi Particle Swarm Optimization (PSO) teknik, adalah untuk mencari lokasi optimum peralatan Flexible AC Transmission System (FACTS) serta untuk mencapai sistem kemampuan beban yang maksimum. Ketika mencari lokasi optimum, had terma untuk sempadan dan had voltan untuk bas-bas itu dibawa sebagai kekangan. Dua jenis peralatan FACTS yang ditekankan di dalam projek ini ialah Unified Power Flow Controller (UPFC) dan Thyristor Controlled Series Compensator (TCSC). Pengoptimuman dipersembahkan berdasarkan tiga parameter iaitu lokasi bagi peralatan FACTS, persekitaran mereka dan jenis mereka. Simulasi-simulasi dipersembahkan berdasarkan sistem IEEE 30 bas untuk lokasi optimum bagi peralatan FACTS dan keputusan yang memperoleh adalah menggalakkan dan akan digunakan dalam membuat penyusunan semula elektrik.

## TABLE OF CONTENT

CHAPTER	TITLE	PAGE
1	INTRODUCTION	1
1.1	Introduction	1
1.2	General introduction of FACTS Devices	2
1.2.1	Unified Power Flow Controller (UPFC)	3
1.2.2	Thyristor Controlled Series Capacitor (TCSC)	3
1.3	General Introduction of Particle Swarm Optimization (PSO)	4
1.4	Objectives	4
1.5	Scope of Work	5
1.6	Problem Statement	5
2	LITERATURE REVIEW	6
2.1	Flexible Alternating Current Transmission System (FACTS) Devices	6
2.1.1	Unified Power Flow Controller	7
2.1.2	Thyristor Controlled Series Compensator	8
2.2	Power System Limit	9
2.2.1	Thermal Limit	9
2.2.2	Voltage Limit	10
2.3	Particle Swarm Optimization	11
2.4	MATLAB 7.5.0 (R2007b)	13

3	METHODOLOGY	14
	3.1 TCSC modeling	14
	3.2 UPFC Modeling	16
	3.3 PSO modeling	18
	3.4 Creating M-file Programming	19
	3.4 Creating Graphical User Interfaces (GUI)	20
4	RESULT AND DISCUSSION	
	4.1 Graphic User Interface (GUI) Main Page	27
	4.2 The solution of optimal location.	31
5	CONCLUSION AND FUTURE RECOMMENDATION	
	5.1 Conclusion	33
	5.2 Future Recommendation	33
	5.2.1 By install two FACTS devices in one time into the system.	34
	5.2.2 Find the optimal location of multi type FACTS devices.	35
	5.2.3 Performed the simulation on various IEEE bus system	35
	5.2.4 Apply another algorithm technique	35

## LIST OF FIGURE

TABLE NO.	TITLE	PAGE
2.1	UPFC Configuration	7
2.2	Basic structure of TCSC	9
3.1	Static model of line with TCSC	14
3.2	Static Power Injection Model of TCSC	15
3.3	Voltage Source Equivalent Circuit of UPFC	17
3.4	UPFC Power Injection Model	17
3.5	Starting the GUI	21
3.6	Main Page to Create New GUI	21
3.7	GUI Layout Area	22
3.8	GUI Design Layout	23
3.9	Property Inspector	24
3.10	M-file Programming	25
3.11	GUI Design	26
4.1	Work path and command window	28
4.1	Main page of the GUI	28
4.2	The Introduction layout	29
3.4	The simulation GUI	29
4.5	Run dialog box	30

4.6	The PSO Initialization	30
4.7	Exit the program	31
4.8	Result for First Iteration.	32

## LIST OF SYMBOLS

FACTS	-	Flexible Alternating Current Transmission Systems
UPFC	-	Unified Power Flow Controller
TCSC	-	Thyristor Controlled Series Capacitor
SVC	-	Static VAR Compensator
TCR	-	Thyristor-controlled Reactor
PSO	-	Particle Swarm Optimization
GA	-	Genetic Algorithms
MATLAB	-	Matrix Laboratory
DC	-	Direct Current
AC	-	Alternating Current
PIM	-	Power Injection Model
GUI	-	Graphical User Interfaces
$\alpha$	-	firing delay
$pbest$	-	Best solution (fitness) that has been achieves so far
$gbest$	-	Best value that is tracked by the global version of the particle swarm optimizer is the overall best value
$rand_1, rand_2$	-	Uniformly random numbers between 0 and 1.
$v_{id}^k$	-	Current velocity of individual $i$ in dimension $d$ at iteration $k$ .
$v_{id}^{k+1}$	-	Velocity of individual $i$ in dimension $d$ at iteration $k+1$ .
$x_{id}^k$	-	Current position of individual $i$ in dimension $d$ at iteration $k$ .
$x_{id}^{k+1}$	-	Position of individual $i$ in dimension $d$ at iteration $k+1$
$pbest_{id}$	-	Dimension $d$ of the $pbest$ of individual $i$ .

$gbest_d$	-	Dimension d of the $gbest$ of the swarm.
$c_1$ and $c_2$	-	The weighting of the stochastic acceleration that pull each particles towards pbest and gbest (cognitive and social acceleration constant respectively).
$w^k$	-	Inertia weight factor that controls the exploitation
$iter^{max}$	-	Maximum number of iterations;
$iter$	-	Current iteration number;
$w^{max}$	-	Maximum inertia weight;
$w^{min}$	-	Minimum inertia weight.
$P_i^F$	-	Real power injections
$Q_i^F$	-	Reactive power injections
$G_{ij}$	-	Conductance of the line- $ij$ .
$B_{ij}$	-	Susceptance of the line- $ij$ .
$V_{se}$	-	Series voltage sources
$V_{sh}$	-	Shunt voltage sources
$I_L$	-	Transmission line current
$P_i + jQ_i$ and $P_j + jQ_j$	-	respectively the equivalent complex power injected into the two busbars, buses $i$ and $j$ , which are practically the resultant power injections contributed by both the series and shunt branches of UPFC.
$V_i, V_j$	-	phase angle components of the voltages on buses $i$ and $j$ .
$b_{se}$	-	the leakage susceptance of the series coupling transformer.
$r$ and $\gamma$	-	respectively, magnitude and phase and angle of series voltage source, UPFC parameters.

## LIST OF APPENDICES

APPENDIX	TITLE	PAGE
A	IEEE 30-bus System	33
B	Bus Data	34
C	Line Data	35
D	Flow Chart	36
E	Programming for the MAIN_PAGE	43
F	Programming for the INTRODUCTION	46
G	Programming for the SIMULATION	48
H	Programming for the PSOsimulation	51
I	Programming for the RUN button	53
J	Programming to find the optimal location	56
K	Programming to find the optimal location	60





## **CHAPTER 1**

### **INTRODUCTION**

#### **1.1 Introduction**

The electric supply industry is undergoing a profound transformation worldwide. Market forces, scarce natural resources and an ever increasing demand for electricity are some of the drivers responsible for such an unprecedented change. Particularly in the case of transmission systems, it requires non-discriminatory open access to transmission resources. Therefore sufficient transmission capacity for supporting transmission services is a great demand to transmission network's requirement. Further to meet the demand for a substantial increase in power transfers among utilities, as a major consequence of electricity market, a much more intensive utilization of existing transmission resource is needed. The advent of Flexible AC Transmission Systems (FACTS) technology has coincided with the major restructuring of the electrical power industry. FACTS can provide benefits in increasing system transmission capacity and power flow control flexibility and rapidity. As deregulation picks up speed, making the

demand for sufficient services is becoming more critical, it is imperative to investigate the capabilities and potential applications of FACTS on power networks [1]-[3].

Population based, cooperative stochastic search algorithms are very popular in the recent years in the research arena of computational intelligence. Most of the population based search approaches are motivated by evolution as seen in nature. Particle swarm optimization (PSO), on the other hand, is motivated from the simulation of social behavior. In this project, applying PSO technique, the optimal location of FACTS devices maximum system loadability, while satisfying the power system constraints, for single type (TCSC and UPFC).

## **1.2 General Introduction of FACTS Devices**

A FACTS is a system comprised of static equipment used for the AC transmission of electrical energy. It is meant to enhance controllability and increase power transfer capability of the network. It is generally a power electronics-based device.

FACTS is defined by the IEEE as “a power electronic based system and other static equipment that provide control of one or more AC transmission system parameters to enhance controllability and increase power transfer capability”.

FACTS devices are proven to be effective in power grids in well-developed countries such as USA, Canada and Sweden. This technology can boost power transfer capability by 20-30% by increasing the flexibility of the systems. It can also increase the loadability or distance to voltage collapse power system, so that, additional loads can be added in the system without addition of new transmission and generating facilities.

In this project, two types of FACTS devices are considered, which are Unified Power Flow Controller (UPFC) and Thyristor Controlled Series Capacitor (TCSC).

### **1.2.1 Unified Power Flow Controller (UPFC)**

UPFC is shunt and series compensation devices. It is well known that UPFC is a powerful and versatile concept for power flow control that has capability of changing power flow. The rapid and almost instantaneous responses make it suitable for many applications requiring effective steady-state power flow control and/or transient and dynamic stability improvement.

UPFC is capable of providing active, reactive and voltage magnitude control under normal and network contingencies conditions without violating the operating limits. From the operational point of view, the UPFC may act as a SVC or as a TCSC or as a phase shift controller.

### **1.2.2 Thyristor Controlled Series Capacitor (TCSC)**

TCSC used in power transmission line can increase the transportability and the stability of system, and decrease the system loss significantly. It is one such device which offers smooth and flexible control for security enhancement with much faster response compared to the traditional control devices.

TCSC is connected in series with the line conductors to compensate for the inductive reactance of the line. It can operate in both capacitive and inductive mode. In capacitive mode, it reduces the transfer reactance between the buses at which the line is connected, thus increasing the maximum power that can be transmitted and reducing the effective active and reactive power losses.

### **1.3 General Introduction of Particle Swarm Optimization (PSO)**

Particle Swarm Optimization is population based stochastic optimization technique developed by Dr. Eberhart and Dr. Kennedy in 1995, inspired by social behavior of bird flocking.

PSO is an extremely simple algorithm that seems to be effective for optimizing a wide range of functions. PSO is applied for solving various optimization problems in electrical engineering.

PSO shared many similarities with evolutionary computation techniques such as Genetic Algorithms (GA). The system is initialized with a population of random solutions and searches for optima by updating generations. However, unlike GA, PSO has no evolution operators. In PSO, the potential solutions, called particles, fly through the problem space by following the current optimum particles. PSO is easy to implement and there are few parameters to adjust.

### **1.4 Objectives**

The objectives of this project are:

- i. To understand the concept of Particles Swarm Optimization (PSO) and Flexible AC Transmission System (FACTS).
- ii. To find the optimal location of FACTS devices to achieve maximum system loadability.
- iii. To identify the best performance of UPFC and TCSC in system loadability.

- iv. To implement the Particle Swarm Optimization technique using the MATLAB

## **1.5 Scope of Work**

The scopes that will be figure out in this research are:

- i. Simulation and modeling of FACTS devices.
- ii. Implement the PSO technique to find the optimal location of FACTS devices in a power system.
- iii. Performing the optimization on three parameters which are the location of the devices, their types and their settings.

## **1.6 Problem Statement**

The problems that are to be faced in planning stage are appropriate type, location, size and setting for these controllers for various applications. This project is based on the optimal location problem in the transmission line in a power system. FACTS devices is one of the most common devices used in the transmission line. However FACTS device has many types and the type has there our model. In this project, we only consider the two types of the devices, which are UPFC and TCSC.

## **CHAPTER 2**

### **LITERATURE REVIEW**

#### **2.1 Flexible Alternating Current Transmission System (FACTS) Devices**

FACTS controllers are products of FACTS technology; a group of power electronics controllers expected to revolutionize the power transmission and distribution system in many ways. FACTS controllers are beginning to appear in the developing countries to appear in the developing countries, as the need for such controllers are recognized well by research communities in this area, and electric power utilities. The FACTS controllers clearly enhance power system performance, improve quality of supply and also provide an optimal utilization of the existing resources [4].

There are several methods for finding the optimal locations of the FACTS controllers in vertically integrated systems as well as unbundled power systems. Other works have incorporated FACTS controllers in optimal power flow formulation with different objective functions [10].

### 2.1.1 Unified Power Flow Controller

The UPFC consists of two identical voltage-source inverters: one in shunt and the other one in series with the line; the general scheme is illustrated in Figure 2.1. Two inverters, namely shunt inverter and series inverter, which operate via a common DC link with a DC storage capacitor, allow UPFC to independently control active and reactive power flows on the line as well as the bus voltage. Active power can freely flow in either direction between the AC terminals of the two inverters through the DC link. Although, each inverter can generate or absorb reactive power at its own AC output terminal, they cannot internally exchange reactive power through DC link. The VA rating of the injected voltage source is determined by the product of the maximum injected voltage and the maximum line current at which power flow is still provided [8].

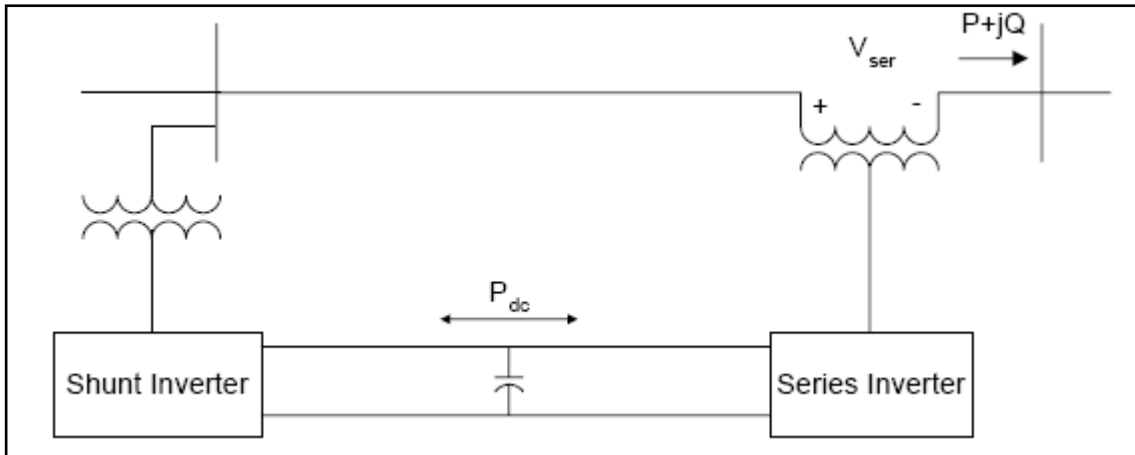


Figure 2.1: UPFC Configuration

The shunt inverter provides local bus voltage control when operated by itself. When operated in conjunction with the series inverter, the shunt inverter has two functions which are to control bus voltage by reactive power injection to the power system and to supply active power to the series inverter via the DC link for series flow control.



The series inverter, on the other hand, provides line power flow by injecting AC voltage with controllable magnitude and phase angle at the power frequency, in series with the line via an insertion transformer. This injected series voltage is, in effect, a synchronous series AC voltage source, which provides active series compensation for line voltage control and angle regulation through the transmission line current. The transmission line currents flow through this voltage sources resulting in active and reactive power exchange between the inverter and the AC system. The active power exchanged at the series AC terminal is converted by the inverter into DC power that appears at the DC link as positive or negative active power demand and transfer to the other converter located at the other side of the line [8].

### **2.1.2 Thyristor Controlled Series Compensator**

TCSC controllers use Thyristor-Controlled Reactor (TCR) in parallel with capacitor segments of series capacitor bank. The basic structure of the device is shown in Figure 2. The combination of TCR and capacitor allow the capacitive reactance to be smoothly controlled over a wide range and switched upon command to a condition where the bi-directional thyristor pairs conduct continuously and insert and inductive reactance into the line.

For operation in the capacitive region, the maximum voltage constrains operation, whereas inductive operation is limited by the maximum firing delay ( $\alpha$ ). Between these constraints is an additional limiting characteristics related to harmonics, which can cause additional heating in the surge reactor and thyristors [8].

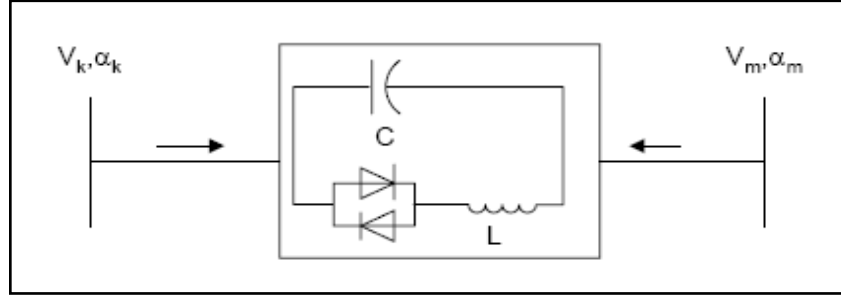


Figure 1.2: Basic structure of TCSC

## 2.2 Power System Limit

For reliability, power system has to be operated within power transfer limits. The limits will constrain the generation and transmission of active and reactive power in the system. They are usually divided into three broad categories, namely thermal, voltage and stability limits. In this project, while finding the optimal location, thermal limit for the lines and voltage limit for the buses are taken as constraints.

### 2.2.1 Thermal Limit

Thermal limits are due to thermal capability of power system equipments. As power transfer increases, current magnitude increases a key to thermal damage. For examples in a power plant, sustained operation of units beyond their maximum operation limits will result in thermal damage. The damage may be to the stator windings or to rotor windings of unit. Both active and reactive powers play a role to current magnitude.

Out in the system, transmission lines and associated equipment must also operate within the thermal limits. Sustained excessive current flow on an overhead line causes the conductors to sag thus decreasing the ground clearance and reducing safety margins, extreme levels of current flow will eventually damage the metallic structure of the conductors producing permanent sag.

Unlike overhead lines, underground cables and transformers must depend on insulation other than air to dissipate the generated heat. These types of equipment are tightly restricted in the amount of current they can safely carry. For the equipment, sustained overloading will result in a reduction in services life due to damage to the insulation. Most power system equipment can be safely overloaded. The important aspect is how much is the overload and how long it does [8].

### **2.2.2 Voltage Limit**

Both utility and customer equipment are designed to operate at a certain rated or nominal supply voltage. A large, prolonged deviation from this nominal voltage can adversely affect the performance of, as well as cause serious damage to system equipment. Current flowing through the transmission lines may produce an unacceptable large voltage drop at the receiving end of system. This voltage drop is primarily due to the large reactive power loss, which occurs as the current flows through the system. If the reactive power produced by generators and other sources are not sufficient to supply the system's demand, voltage will fall, outside the acceptable limit that is typically  $\pm 6\%$  around the nominal value.

System often requires reactive support to help prevent low voltage problems. The amount of available reactive support often determines power transfer limits. A system may be restricted to a lower level of active power transfer than desired because the

system does not possess the required reactive power reserves to sufficiently support voltage.

### 2.3 Particle Swarm Optimization

Particle Swarm Optimization (PSO) is a population based stochastic optimization technique developed by Kennedy and Eberhart. The method is derived from simulation of a simplified social model of swarms such as fish schooling and bird flocking, is based on a simple concept, has been found to be robust for solving problems featuring non-linearity and non-differentiability, multiple optima and high dimensionality through adaptation and provides high quality solutions with stable convergence.

The individuals (particles) persist over time, influencing one another's search of the problem space, as compared with genetic algorithms where the weakest chromosomes are immediately discarded. Instead of using evolutionary operators to manipulate the individuals as in other evolutionary computation algorithms, each individual in the swarm flies in the search space with a velocity which is dynamically adjustable according to its own flying experience (velocity, inertia, gravity) and its coordinates in the problem space, which are associated with the best solution (fitness) it has achieved so far. This value is called *pbest*.

Another best value that is tracked by the global version of the particle swarm optimizer is the overall best value, and its location, obtained so far by any particle in the population. This is called *gbest*. The basic concept of PSO technique lies in accelerating each particle towards its *pbest* and *gbest* locations at each time step. The modified velocity of each particle can be computed using the current velocity and the distance from *pbest* and *gbest* according to (1). The positions are modified using (2).

$$v_{id}^{k+1} = w^k \cdot v_{id}^k + c_1 \cdot rand_1 \cdot (pbest_{id} - x_{id}^k) + c_2 \cdot rand_2 \cdot (gbest_d - x_{id}^k) \quad (1)$$

$$x_{id}^{k+1} = x_{id}^k + v_{id}^{k+1} \quad (2)$$

Where:

$rand_1, rand_2$ : Uniformly random numbers between 0 and 1.

$v_{id}^k$  : Current velocity of individual  $i$  in dimension  $d$  at iteration  $k$ .

$v_{id}^{k+1}$  : Velocity of individual  $i$  in dimension  $d$  at iteration  $k+1$ .

$x_{id}^k$  : Current position of individual  $i$  in dimension  $d$  at iteration  $k$ .

$x_{id}^{k+1}$  : Position of individual  $i$  in dimension  $d$  at iteration  $k+1$

$pbest_{id}$  : Dimension  $d$  of the  $pbest$  of individual  $i$ .

$gbest_d$  : Dimension  $d$  of the  $gbest$  of the swarm.

$c_1$  and  $c_2$  : The weighting of the stochastic acceleration that pull each particles towards  $pbest$  and  $gbest$  (cognitive and social acceleration constant respectively).

$w^k$  : Inertia weight factor that controls the exploitation and exploration of the search space by dynamically adjusting the velocity and it is computed using (3).

$$w^k = w^{max} - \frac{w^{max} - w^{min}}{iter^{max}} * iter \quad (3)$$

Where:

$iter^{max}$  : Maximum number of iterations;

$iter$  : Current iteration number;

$w^{max}$  : Maximum inertia weight;

$w^{min}$  : Minimum inertia weight.

The particle velocity is limited by the maximum value  $v^{max}$ .

## 2.4 MATLAB 7.5.0 (R2007b)

MATLAB is a high-performance language for technical computing. It integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation. Typical uses include Math and computation Algorithm development Data acquisition Modeling, simulation, and prototyping Data analysis, exploration, and visualization Scientific and engineering graphics Application development, including graphical user interface building.

MATLAB is an interactive system whose basic data element is an array that does not require dimensioning. This allows solving many technical computing problems, especially those with matrix and vector formulations, in a fraction of the time it would take to write a program in a scalar no interactive language such as C or FORTRAN.

The name MATLAB stands for matrix laboratory. MATLAB was originally written to provide easy access to matrix software developed by the LINPACK and EISPACK projects. Today, MATLAB engines incorporate the LAPACK and BLAS libraries, embedding the state of the art in software for matrix computation.

MATLAB has evolved over a period of years with input from many users. In university environments, it is the standard instructional tool for introductory and advanced courses in mathematics, engineering and science. In industry, MATLAB is the tool of choice for high-productivity research, development and analysis.

MATLAB features a family of add-on application-specific solutions called toolboxes. Very important to most users of MATLAB, toolboxes allow you to learn and apply specialized technology. Toolboxes are comprehensive collection of MATLAB function (M-files) that extend the MATLAB environment to solve particular classes of problems. Areas in which toolboxes are available include signal processing, control systems, neural networks, fuzzy logic, wavelets, simulation and many others.

In this project, there are two parts of MATLAB, which are, the programming parts using the M-file and the second part is the Graphical User Interfaces (GUI). Both parts will be explained in detail in the next chapter.

## **CHAPTER 3**

### **METHODOLOGY**

#### **3.1 TCSC modeling**

The modeling of TCSC of static model of line with TCSC and static power injection model of TCSC are referred in reference [9]-[11]. A static Power Injection Model (PIM) of TCSC has been used. The injection model represents the TCSC as a device that injects certain amount of active and reactive power in a node.

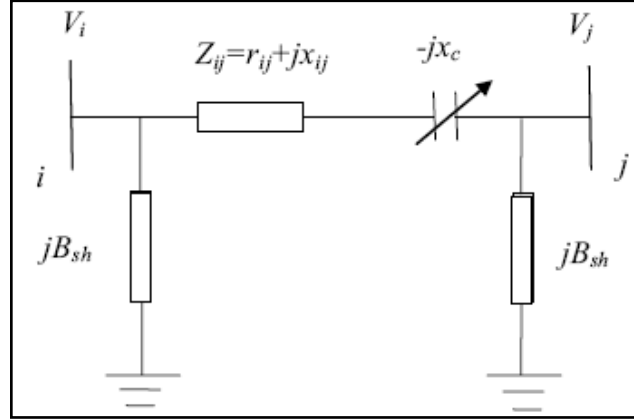


Figure 3.1: Static model of line with TCSC

Figure 3.1 shows a model of transmission line with TCSC connected between buses  $i$  and  $j$ . The transmission line is represented by its lumped  $\Pi$ -equivalent parameters, connected between the two buses. During steady state, the TCSC can be considered as a static reactance  $-jx_c$ . The controllable reactance  $x_c$  is directly used as the control variable in the power flow equations. The corresponding power injection model of TCSC, incorporated in the transmission line, is shown in Figure 3.2. The real ( $P_i^F$ ) and reactive ( $Q_i^F$ ) power injections, due to TCSC at buses  $i$  and  $j$  are given by the following equations:

$$P_i^F = V_i^2 \Delta G_{ij} - V_i V_j [\Delta G_{ij} \cos(\delta_i - \delta_j) + \Delta B_{ij} \sin(\delta_i - \delta_j)] \quad (4)$$

$$Q_i^F = -V_i^2 \Delta B_{ij} - V_i V_j [\Delta G_{ij} \sin(\delta_i - \delta_j) - \Delta B_{ij} \cos(\delta_i - \delta_j)] \quad (5)$$

$$P_j^F = V_j^2 \Delta G_{ij} - V_i V_j [\Delta G_{ij} \cos(\delta_i - \delta_j) - \Delta B_{ij} \sin(\delta_i - \delta_j)] \quad (6)$$

$$Q_j^F = -V_j^2 \Delta B_{ij} + V_i V_j [\Delta G_{ij} \sin(\delta_i - \delta_j) + \Delta B_{ij} \cos(\delta_i - \delta_j)] \quad (7)$$

Where,

$$\Delta G_{ij} = \frac{x_c r_{ij} (x_c - x_{ij})}{(r_{ij}^2 + x_{ij}^2) \{r_{ij}^2 + (x_{ij} - x_c)^2\}} \quad (8)$$



$$\Delta B_{ij} = \frac{x_c(r_{ij}^2 - x_{ij}^2 + x_c x_{ij})}{(r_{ij}^2 + x_{ij}^2) \{r_{ij}^2 + (x_{ij} - x_c)^2\}} \quad (9)$$

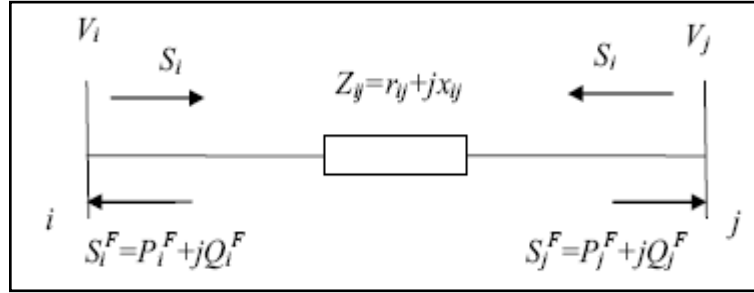


Figure 3.2: Static Power Injection Model of TCSC

Where,  $V_i$ ,  $V_j$  and  $\delta_i, \delta_j$  are voltage and angle at buses  $i$  and  $j$ , respectively.  $G_{ij}$  and  $B_{ij}$  are the conductance and susceptance of the line- $ij$ .

### 3.2 UPFC Modeling

Steady-state investigation of UPFC involves power flow studies which include the calculation of bus voltages, branch loadings, real and reactive transmission losses, and the impact of UPFC on the above mentioned system parameters. In order to evaluate UPFC overall steady-state performance and adequate model is required. A UPFC model using power injection concept, in this model, two voltage sources are used to represent the fundamental components of the pulse width modulated controlled output voltage waveforms of the two branches in the UPFC. The impedance of the two coupling

transformers is included in the proposed model and losses of UPFC are taken into account. Figure 3.3 depicts voltage source equivalent circuit of UPFC [12].

The series injection branch, a series injection voltage source, performs the main functions of controlling power flow whilst the shunt branch is used to provide real power demanded by the series branch and the losses in UPFC. However, in the proposed model, this function of reactive compensation of shunt branch is completely neglected. As shown in Figure 3.3, the series and shunt injection branch are modeled with two ideal controllable voltage sources,  $V_{se}$  and  $V_{sh}$  respectively, while  $X_{se}$  and  $X_{sh}$ , respectively, and denote the leakage reactance of the two coupling transformers.  $I_L$  represents transmission line current [12].

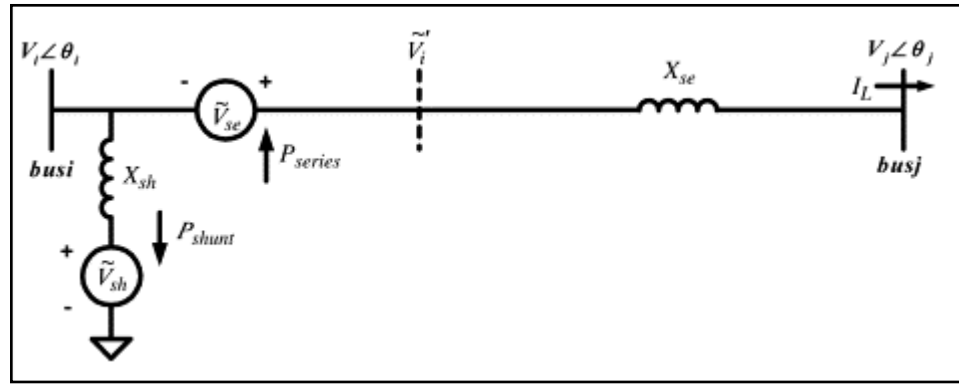


Figure 3.3: Voltage Source Equivalent Circuit of UPFC

Series voltage source  $V_{se}$ , can be mathematically expressed as follows:

$$V_{se} = rV_i e^{j\gamma} \quad (11)$$

Where,

$$0 \leq r \leq r_{max} \text{ and } 0 \leq \gamma \leq 2\pi$$

For the purpose of simplifying the formulation procedure of the power injection model, which has been derived in rectangular form is adopted here as shown in Figure 3.4.

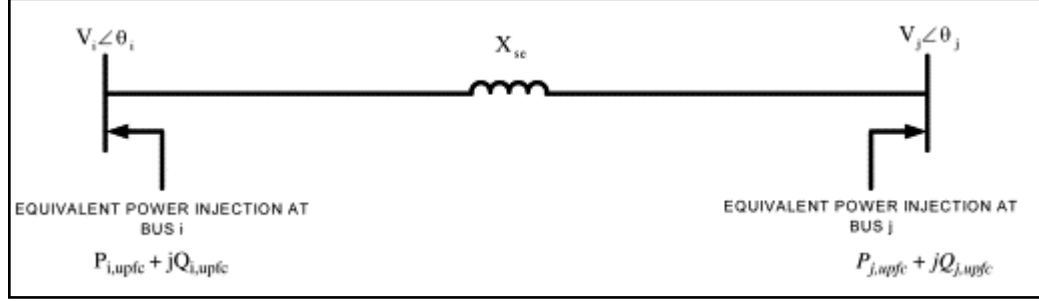


Figure 3.4: UPFC Power Injection Model

The components of equivalent power injections at buses  $i$  and  $j$ ,  $P_i$ ,  $Q_i$ ,  $P_j$  and  $Q_j$  are formulated as follows:

$$P_i = 0.02 r b_{se} V_i^2 \sin \gamma - 1.02 r b_{se} V_i V_j \sin(\theta_i - \theta_j + \gamma) \quad (12)$$

$$Q_i = -r b_{se} V_i^2 \cos \gamma \quad (13)$$

$$P_j = r b_{se} V_i V_j \sin(\theta_i - \theta_j + \gamma) \quad (14)$$

$$Q_j = r b_{se} V_i V_j \cos(\theta_i - \theta_j + \gamma) \quad (15)$$

In equation ((12),(13),(14) and (15));

$P_i + jQ_i$  and  $P_j + jQ_j$  : respectively the equivalent complex power injected into the two busbars, buses  $i$  and  $j$ , which are practically the resultant power injections contributed by both the series and shunt branches of UPFC.

$V_i, V_j$  : respectively, the phase angle components of the voltages on buses  $i$  and  $j$ .

$b_{se}$  : the leakage susceptance of the series coupling transformer.

$r$  and  $\gamma$  : respectively, magnitude and phase and angle of series voltage source, UPFC parameters.

### 3.3 PSO modeling

As stated at the previous chapter, PSO is initialized with a group of random particles (solution) and then searches for optima by updating generations. In every iteration, each particle is updated by following two “best” values. The first one is the best solution it has achieved so far. This value called *pbest*. Another one called *gbest* which is global best, this value is tracked by the particle swarm optimizer is the best value, obtained so far by any particle in the population.

After finding the two values, the particle updates its velocity and positions with equation (1) and (2). The pseudo code of the procedure is as follows:

```

For each particle
    Initialize particle
End
Do
    For each particle
        Calculate fitness value
        If the fitness value is better than the best pbest value in history set
            current value as the new pbest
    End
    Choose the particle with the best fitness value of all the particles as the gbest
    For each particle
        Calculate particle velocity according equation (1)
        Update particle position according equation (2)
    End
End

```

Particle velocity on each dimension are clamped to a maximum velocity  $V_{max}$ . If the sum of accelerations would cause the velocity on that dimension to exceed  $V_{max}$ , which is a parameter specified by the user. Then the velocity on that dimension is limited to  $V_{max}$ .

### 3.4 Creating M-file Programming

Based on the TCSC, UPFC and PSO modeling above, a programming had been create to find the optimal location of FACTS devices. Refer Appendix E.

### 3.5 Creating Graphical User Interfaces (GUI)

The main reason GUIs are used is because it makes things simple for the end-users of the program. If GUIs were not used, people would have to work from the command line interface, which can be extremely difficult and frustrating. This part will explain the step that had been used to create the GUI.

To start the GUI, click at File > New > GUI, and you should see the following screen appear as shown in Figure 3.5. The GUIDE Quick Start dialog will appear as shown in Figure 3.6, user can create a new GUI by choosing the one of the appropriate GUIDE templates. To open an existing GUI in GUIDE choose Open Existing GUI, you can choose a GUI from your current directory or browse other directories. In this part choose Blank GUI (default) to create the new templates. The result should be appearing as shown in Figure 3.7.

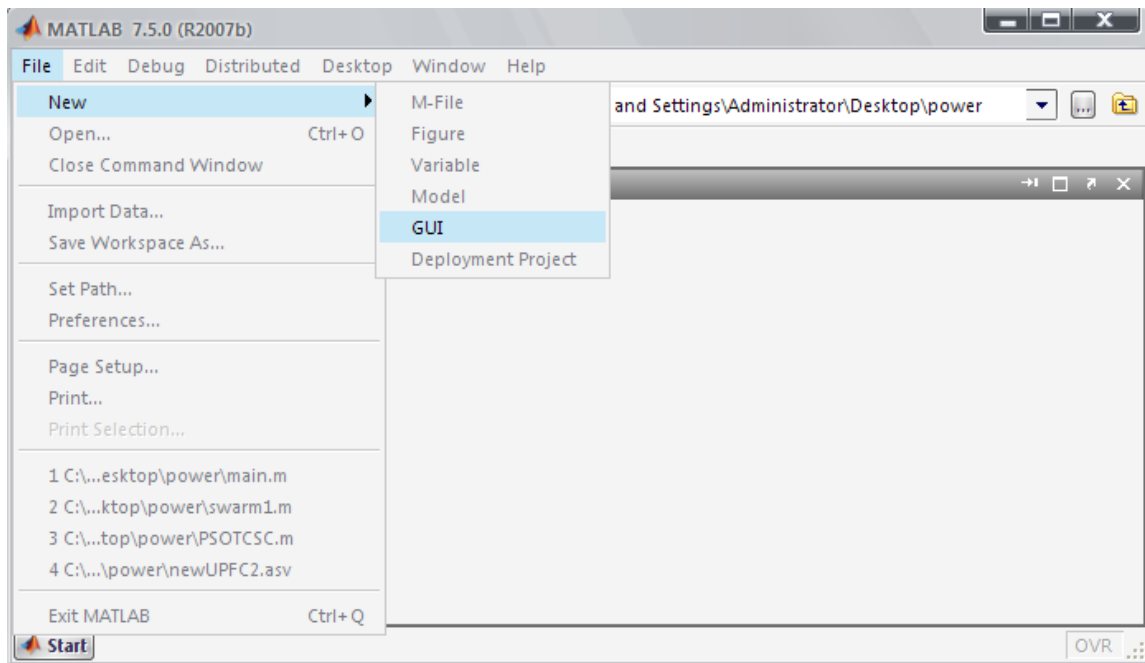


Figure 3.5: Starting the GUI

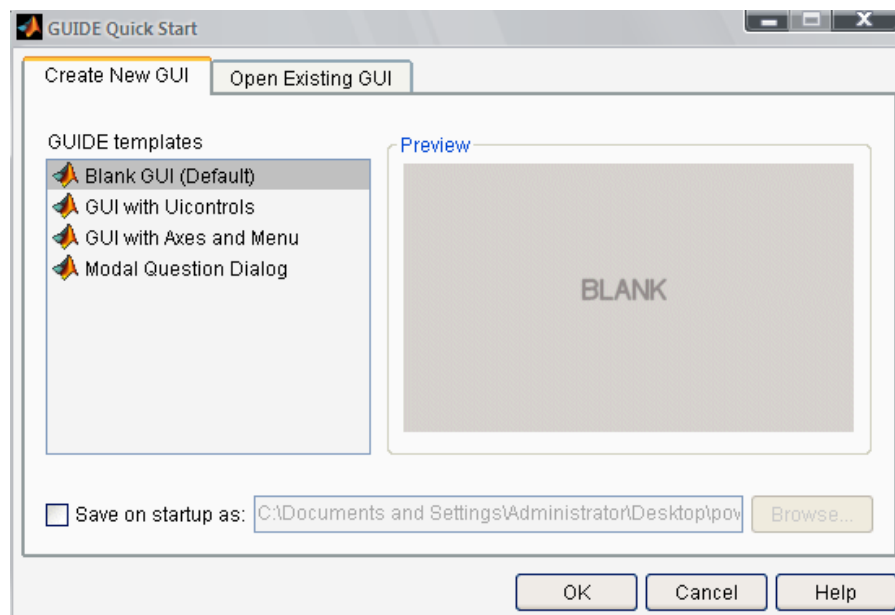


Figure 3.6: Main Page to Create New GUI

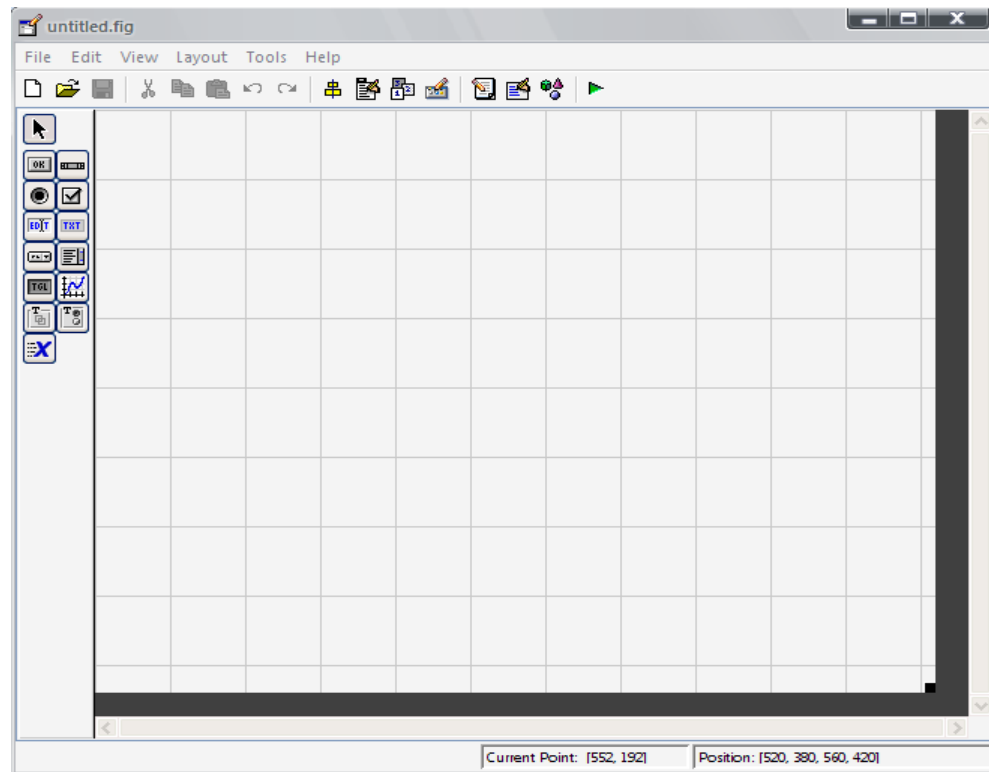


Figure 3.7: GUI Layout Area

At the GUI layout area, programmer can use their creativity to design the GUI by selecting and aligning the GUI components by dragging it into the layout. After design the layout, used the Property Inspector to give each component a name and to set the characteristics of each component, such as its color, the text to displays, font size, font style and so on.

For the main page GUI (Figure 3.11) we will need the following components:



Three Axes components



Two Static Text components



One Push Button component

Add in all the components to the GUI by clicking on the icon and placing it onto the grid. At this point, your GUI should look similar to the Figure 3.8 below:

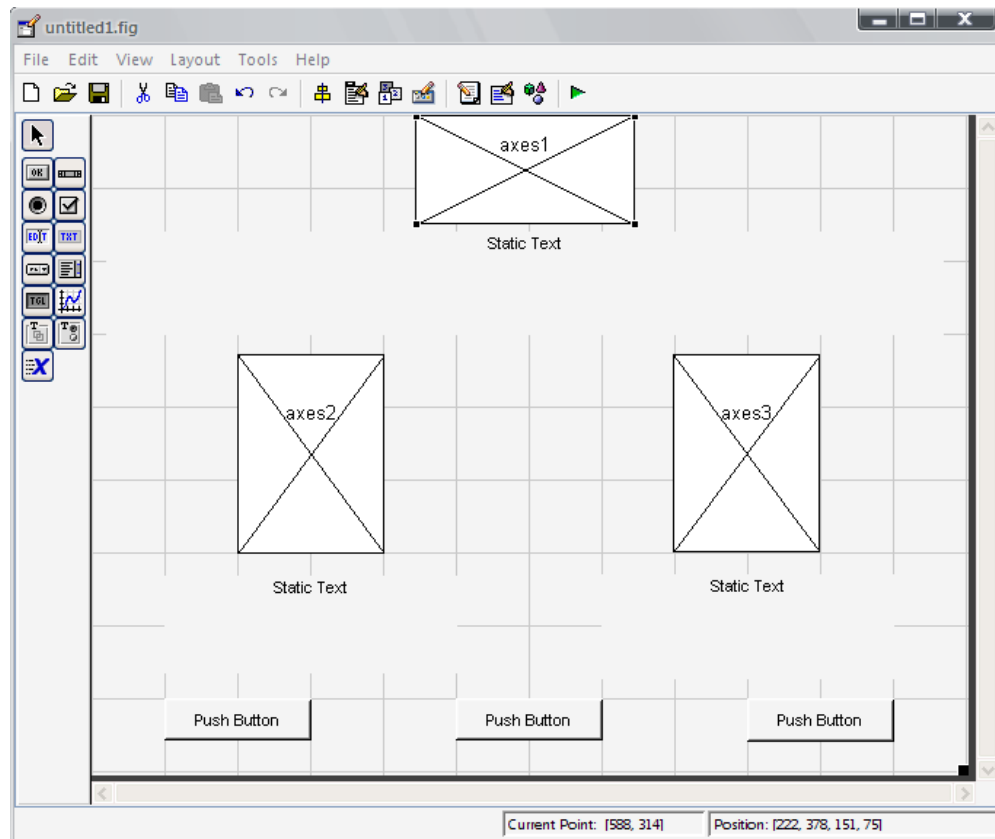


Figure 3.8: GUI Design Layout

Now, edit the properties of these components. For example here we look at the static text. Double click one of the Static Text component and then the following Figure 3.9 will appear. It is called the Property Inspector and it allows you to modify the properties of a component.



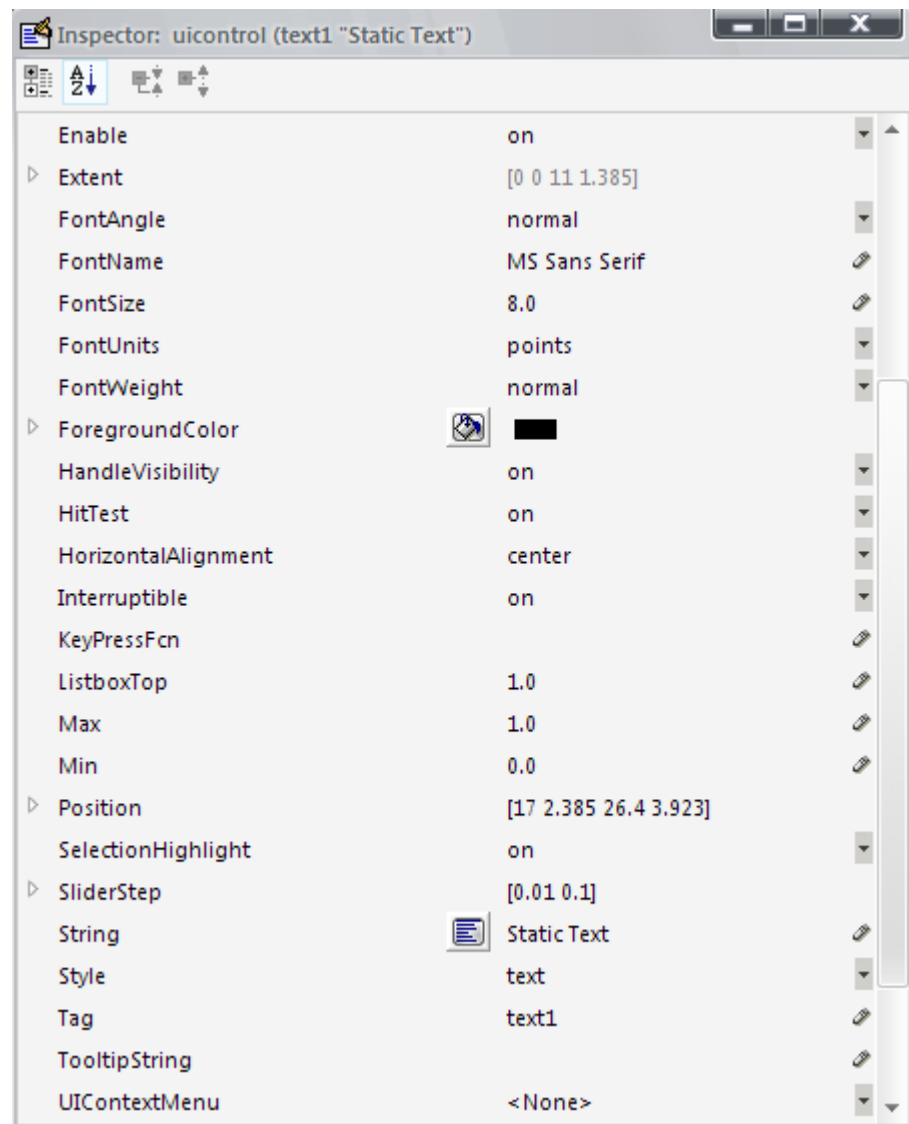


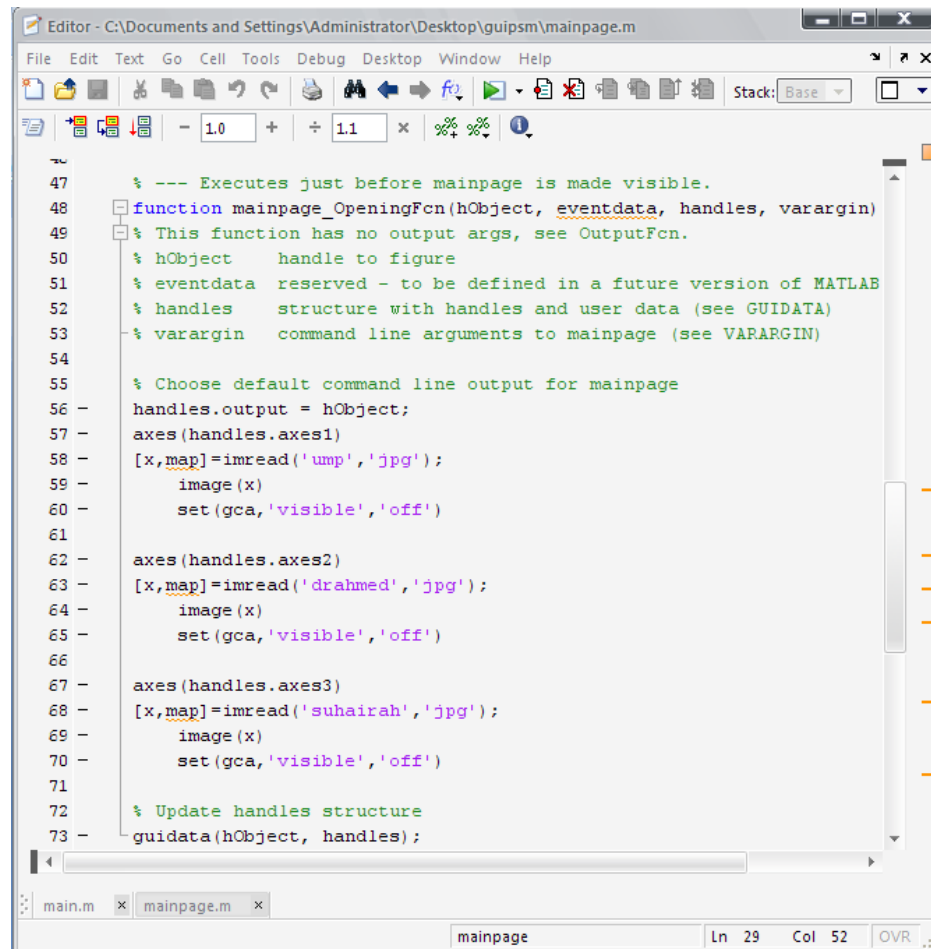
Figure 3.9: Property Inspector

Do the same step for all the other axes and push button components. After you done it, save the GUI under any file name. When you save this file, MATLAB automatically generates two files: *mainpage.m* (Figure 3.10) and *mainpage.fig* (Figure 3.11).

The .m file contains all the code for the GUI here the behavior of the GUI can be programmed by several code or function. This code or function will controls how the GUI respond to events such as button clicks, slider movement, menu item selection or

the creation and deletion of component. This programming takes the form of set of function, called callbacks, for each component and for the GUI itself.

While the .fig file contains the graphics of your interface.



```

47 % --- Executes just before mainpage is made visible.
48 function mainpage_OpeningFcn(hObject, eventdata, handles, varargin)
49 % This function has no output args, see OutputFcn.
50 % hObject    handle to figure
51 % eventdata  reserved - to be defined in a future version of MATLAB
52 % handles    structure with handles and user data (see GUIDATA)
53 % varargin   command line arguments to mainpage (see VARARGIN)
54
55 % Choose default command line output for mainpage
56 handles.output = hObject;
57 axes(handles.axes1)
58 [x,map]=imread('ump','jpg');
59 image(x)
60 set(gca,'visible','off')
61
62 axes(handles.axes2)
63 [x,map]=imread('drahmed','jpg');
64 image(x)
65 set(gca,'visible','off')
66
67 axes(handles.axes3)
68 [x,map]=imread('suhairah','jpg');
69 image(x)
70 set(gca,'visible','off')
71
72 % Update handles structure
73 guidata(hObject, handles);

```

Figure 3.10: M-file Programming

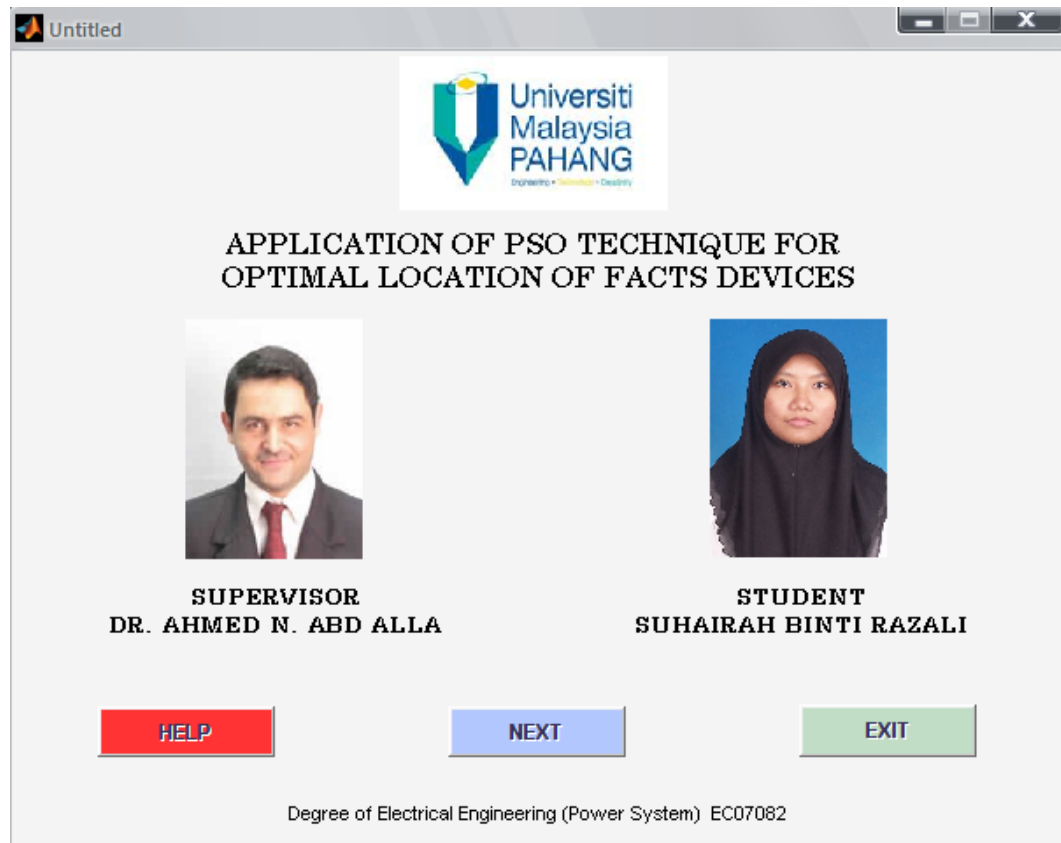


Figure 3.11: GUI Design

## **CHAPTER 4**

### **RESULT AND ANALYSIS**

#### **4.1 Graphic User Interface (GUI)**

In this project, Application of PSO Technique for Optimal Location of FACTS Devices, a GUI has been created by using MATLAB. This GUI is consists of three part or layout which are 'MAIN\_PAGE' layout, 'INTRODUCTION' layout and 'SIMULATION' layout. The 'MAIN\_PAGE' layout show the brief detail of the programmer, supervisor, title and the main part before proceed in detail. This layout consists of three button, which are NEXT and EXIT. Each push button will execute its own GUI. When the 'HELP' button is clicked, it will call 'HELP.fig'.

Before calling the GUI main page, make sure that you have choose the appropriate work path as shown in figure below. Work path is the location of the project file that has been saved.

Then write the main file's name in the command window that shown in Figure 4.1. Figure 4.2 is the main page of this project.

The programming for the main page is shown in Appendix E.

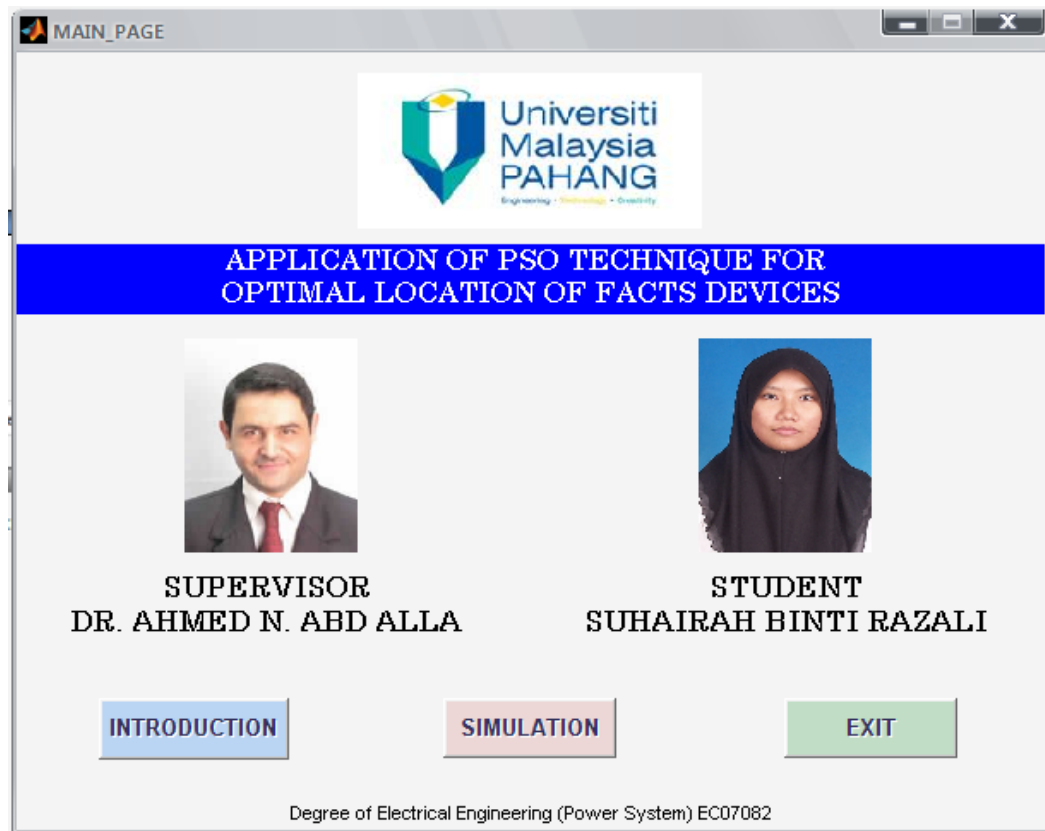
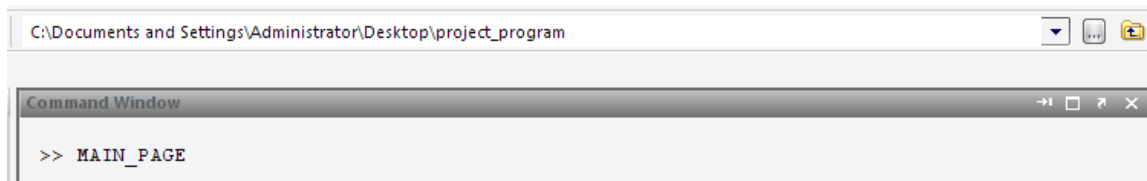


Figure 4.2: Main page of the GUI

Click at the 'INTRODUCTION' button, and then the other layout come out, refer the Figure 4.3. This layout gives the brief information about the FACTS devices and also PSO. To return back to the main page, click EXIT button. The programming to this layout will be attached to the Appendix F.

The EXIT button brings the INTRODUCTION layout to the MAIN\_PAGE layout.

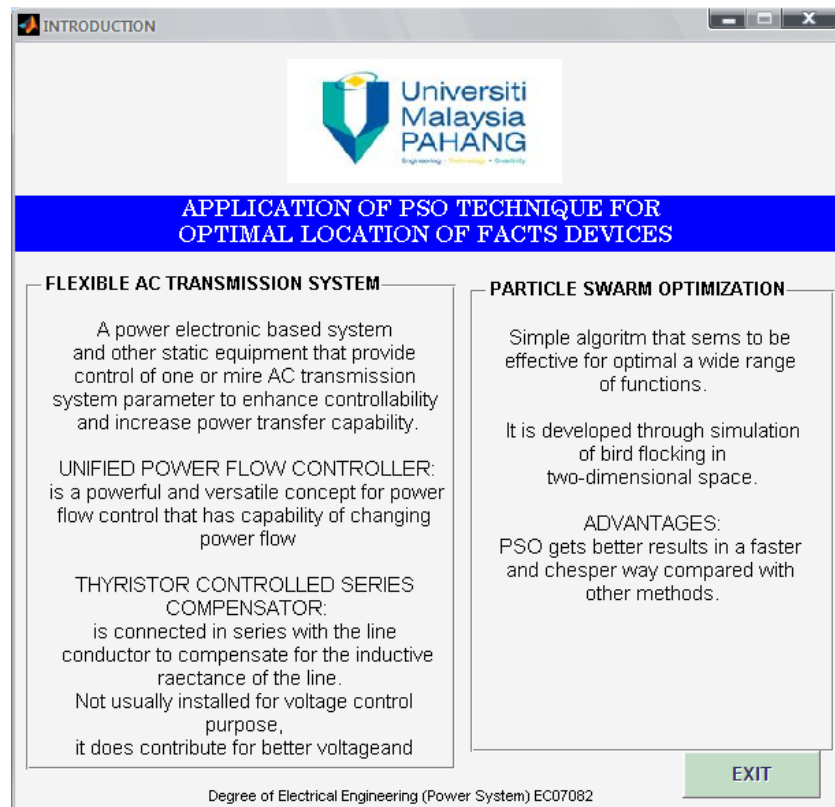


Figure 4.3: The Introduction layout

From the main page layout, go to the button 'SIMULATION', then this layout will come out as Figure 4.4.



Figure 4.4: The simulation GUI.

The purpose of this layout is to show how the PSO work. We can see that, the particle in system moving randomly towards a best value. So this best value will be implementing in the FACTS devices as the optimal location. Click the run button, then a dialog box as shown in Figure 4.5 will pop out.

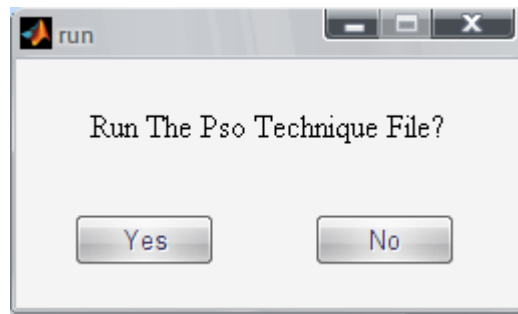


Figure 4.5 : Run dialog box

The 'Yes' button allow the program to execute, while the 'No' button terminate the running. The programming is set to run the 49 swarm size with iteration of 30 times. Figure 4.6 shows what is the swarm size in the system.

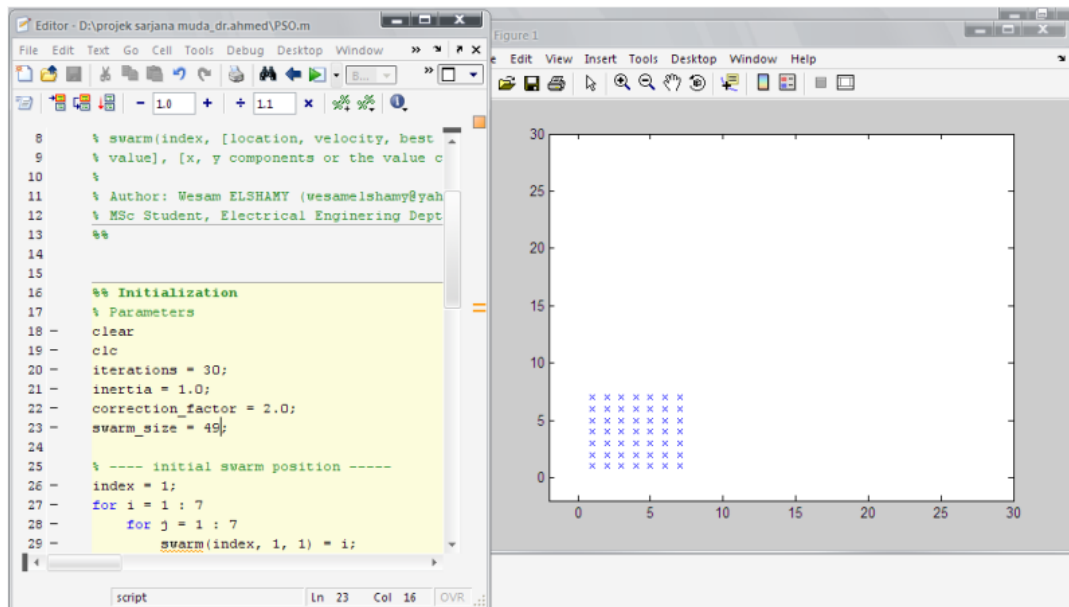


Figure 4.6: The PSO Initialization

Appendix G shows the programming of the ‘SIMULATION’ and Appendix H for the ‘PSOsimulation’. To exit the whole program, click the ‘EXIT’ button on the main page and the ‘Yes’ to exit the program and ‘No’ to continue the program. Appendix I shows the programming.

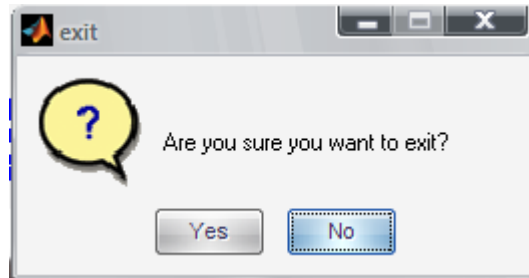


Figure 4.7: Exit the program

#### **4.2 The solution of optimal location.**

Refer to the Appendix J and K for the programming to find the optimal location of FACTS devices.

The result from the programming will be different in every iteration, this is because in every iteration the PSO will update the value. At the end of the execution, MATLAB give the total loss of the system and will ask to write the `xpbest` and `vgbest` to display the particles best solution and the global best solution.



```
      Total loss                      17.986    23.093
Testing has been carried out successfully
Type xpbest to see the particle solution
Type xgbest to see the global solution
>> xpbest

xpbest =

      6      25
      9      19
      8       6
      9       9
      5      14
      1       8
      7      17
     23      24
      6      21
      6       2

>> xgbest

xgbest =

      6      25
```

Figure 4.8: Result for First Iteration.

## **CHAPTER 5**

### **CONCLUSION AND FUTURE RECOMMENDATION**

#### **5.1 Conclusion**

By using the PSO technique, the optimal location of FACTS devices can be determined. The PSO technique as stated in the previous chapter will choose the tested transmission line randomly, and they will find the best value by doing the iteration. Every iteration will come out with the different solution because PSO will always update the best value in every iteration.

#### **5.2 Future Recommendation**

There are few suggestion for the future recommendation in order to find the practical solution for the distribution system problem.

### **5.2.1 By install two FACTS devices in one time into the system.**

In this project, we only considered only one FACTS device to be installed in a time. In the case 1, UPFC had been installed in the transmission line of the IEEE 30-bus system, while in case 2, TCSC had been installed in the transmission line of the IEEE 30-bus system.

In the future, we can add two FACTS devices in a time. Instead using UPFC or TCSC in a time, both of the devices will be installed in a time but in the different transmission line. And then, we can compare the result, which solution give optimal location in the power system.

### **5.2.2 Find the optimal location of multi type FACTS devices.**

The application of particle swarm optimization algorithm to find the optimal location of multi type FACTS devices in a power system in order to eliminate or alleviate the line over loads. The optimization can be performed n the parameter, namely the location of the devices, their types, their setting and installation cost of FACTS devices for single and multiple contingencies. This recommendation is a procedure to place multi type FACTS devices along the system branches based on the contingency severity index values to alleviate system overloads and to improve the system security margin during single and double contingencies.

### **5.2.3 Performed the simulation on various IEEE bus system**

Instead of using the IEEE 30-bus system, there are various IEEE bus system such as IEEE 6-bus system, IEEE 14-bus system, IEEE 118-bus system or using TNEB 69-bus system as test system to the future project.

### **5.2.4 Apply another algorithm technique**

By applying another algorithm technique, such as, Genetic Algorithm (GA) or Evolutionary Programming (EP) algorithm. Find the optimal location of FACTS devices in power system using that technique. Both techniques will give the different result or different solution, so we can do the comparison which technique will perform the best solution, which one required simple programming and which one is the most efficient algorithm technique.

Genetic Algorithms are global search techniques, based on the mechanisms of natural selection and genetic. They can search several possible solutions simultaneously and they do not require any prior knowledge or special properties of the objective function. The Gas start with random generation of initial population and then the selection, crossover and mutation are proceeded until the maximal generation is reached. Additionally, Gas are practical algorithm and easy to be implemented in the power system analysis [13].

The Evolutionary Programming Algorithm is a computational optimization method, which uses the mechanic of evolution to find the global optimal solution of complex optimization problems. The EP Algorithm starts with random generation of initial individuals in a population and then the mutation and selection are proceeded until the best individual, which has the highest fitness, is found [14].

## REFERENCE

- [1] N.G. Hingorani and L.Gyugyi, *Understanding FACTS Concepts and Technology of Flexible AC Transmission Systems*, IEEE Press, 2000, ISBN 0-7803-3455-8.
- [2] R.M.Mathur and R.K. Varma, *Thyristor based FACTS controllers for Electrical Transmission System*. John Wiley & Sons Inc, 2002.
- [3] Y.H. Song and X.F. Wang, *Operation of Market Oriented Power System*, Springer-Verlag Ltd, 2003, ISBN: 1-85233-670-6.
- [4] James Kennedy, Russell Eberhart. *Particle Swarm Optimization*. Proc. IEEE Int'l. Conf. on Neural Networks (Perth, Australia),IEEE Service Center, Piscataway, NJ, IV:1942-1948. Available at: <http://www.engr.iupui.edu/~shi/Coference/psopap4.html>
- [5] M. Saravanan, S. Mary Raja Slochanal, P. Venkatesh, Prince Stephen Abraham. J. *Application of PSO technique for optimal location of FACTS devices considering system loadability and cost of installation*.
- [6] Citing Internet sources URL <http://www.swarmintelligence.org/index.php>
- [7] J. Baskaran#, Dr. V. Palanisamy (2005). *Genetic Algorithm applied to Optimal Location of FACTS Device in a Power System Network considering economic saving cost*. Academic Open Internet Journal. Available at: <http://www.acadjournal.com/2005/v15/part6/p3/>
- [8] Dr. Nadarajah Mithulananthan, Mr. Artit Sode-yome, Mr. Naresh Acharya(2003). *Application of FACTS Controllers in Thailand Power Systems*. RTG Budget-Joint Research Project, Fiscal.

[9] Lennart Angquist, Gunnar Ingestrom, Hans-Ake Jonsson (1996). *Dynamical Performance of TCSC Schemes*. ABB Power System AB Sweden.

[10] Srinivasa Rao Pudi, S.C. Srivastava, Senior Member, IEEE (2008). *Optimal Placement of TCSC Based on A Sensitivity Approach for Congestion Management*. Fifth National Power Systems Conference (NPSC), IIT Bombay.

[11] Seyed Abbas Taher, Hadi Besharat. *Transmission Congestion Management by Determining Optimal Location of FACTS Devices in Deregulated Power Systems*. American Journal of Applied Sciences 5 (3): 242-247, 2008, ISSN 1546-9239.

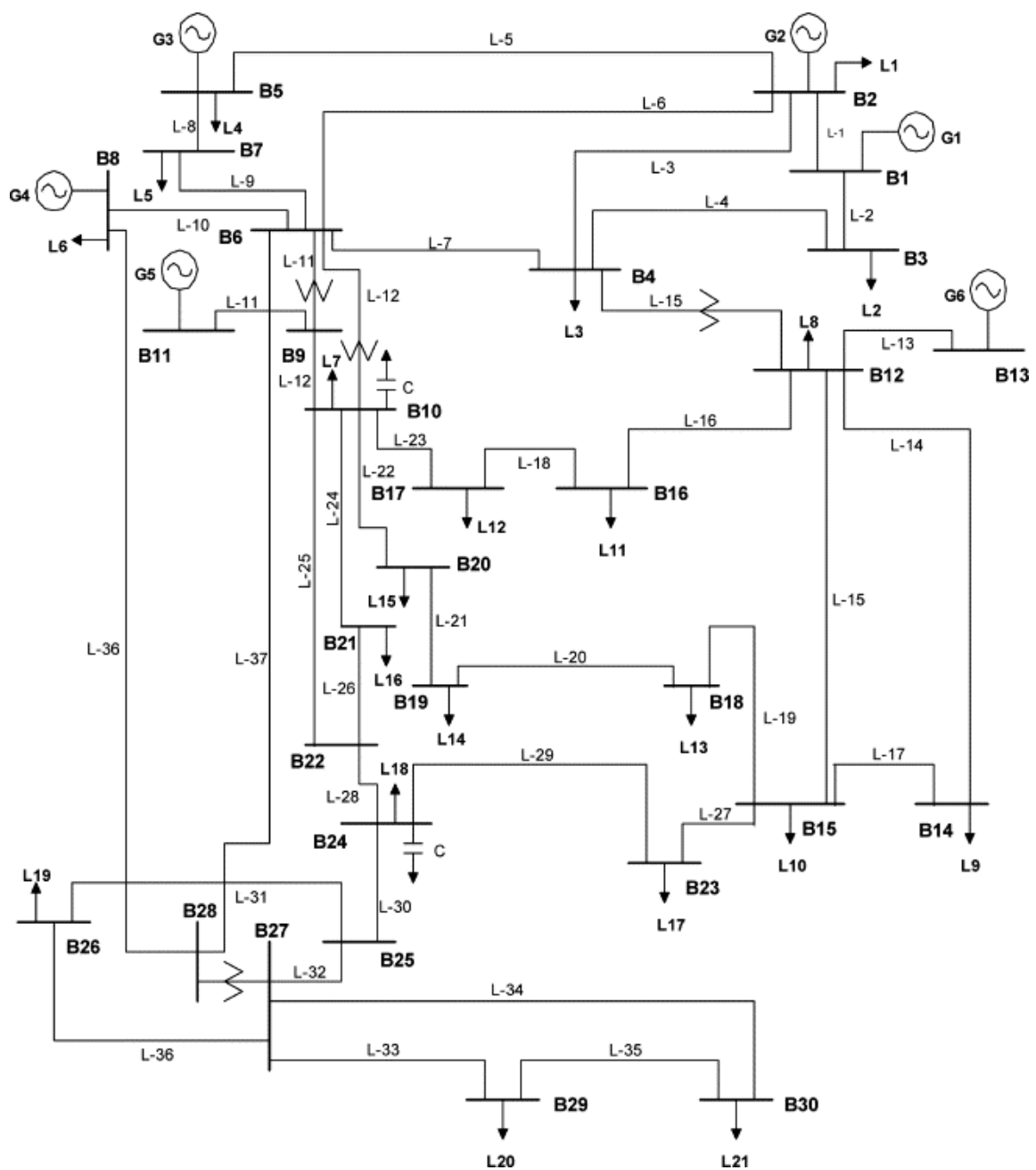
[12]	Citing	Internet	sources	URL
<a href="http://www.sciencedirect.com/science?_ob=ArticleURL&amp;_udi=B6V2T-4CF16W3-2&amp;_user=4406426&amp;_rdoc=1&amp;_fmt=&amp;_orig=search&amp;_sort=d&amp;_docanchor=&amp;view=c&amp;_searchStrId=1065565538&amp;_rerunOrigin=google&amp;_acct=C000063100&amp;_version=1&amp;_urlVersion=0&amp;_userid=4406426&amp;md5=4ac5eaf18b2d2a132c3905a200e03e28">http://www.sciencedirect.com/science?_ob=ArticleURL&amp;_udi=B6V2T-4CF16W3-2&amp;_user=4406426&amp;_rdoc=1&amp;_fmt=&amp;_orig=search&amp;_sort=d&amp;_docanchor=&amp;view=c&amp;_searchStrId=1065565538&amp;_rerunOrigin=google&amp;_acct=C000063100&amp;_version=1&amp;_urlVersion=0&amp;_userid=4406426&amp;md5=4ac5eaf18b2d2a132c3905a200e03e28</a>				

[13] L.J. Cai, I.Erich, G. Stamtsis (2004). *Optimal Choice and Allocation of FACTS Devices in Deregulated Electricity Market using Genetic Algorithms*.

[14] Weerakorn Ongsakul, Peerapol Jirapong (2005). *Optimal Allocation of FACTS Devices to Enhance Total Transfer Capability Using Evolutionary Programming*. School of Environment, Resources, and Development Asian Institute of Technology, Pathumthani, Thailand.

## APPENDIX A

## IEEE 30-bus System



## APPENDIX B

## Bus Data

Bus number	Bus voltage		Bus generation		Bus load	
	Magnitude per unit	Phase angle (°)	Real MW	Reactive MVAR	Real MW	Reactive MVAR
1	1.0600	0.00	243.25	-17.91	0.00	0.00
2	1.0450	-5.01	40.00	51.57	21.70	12.70
3	1.0229	-6.96	0.00	0.00	2.40	1.20
4	1.0142	-8.57	0.00	0.00	7.60	1.60
5	1.0100	-13.53	0.00	34.86	94.20	19.00
6	1.0122	-10.21	0.00	0.00	0.00	0.00
7	1.0036	-12.10	0.00	0.00	22.80	10.90
8	1.0100	-10.92	0.00	31.57	30.00	30.00
9	1.0535	-12.56	0.00	0.00	0.00	0.00
10	1.0489	-13.79	0.00	0.00	5.80	2.00
11	1.0820	-12.56	0.00	14.82	0.00	0.00
12	1.0584	-13.67	0.00	0.00	11.20	7.50
13	1.0710	-13.67	0.00	9.65	0.00	0.00
14	1.0460	-13.66	0.00	0.00	6.20	1.60
15	1.0394	-14.27	0.00	0.00	8.20	2.50
16	1.0472	-13.98	0.00	0.00	3.50	1.80
17	1.0433	-14.06	0.00	0.00	9.00	5.80
18	1.0307	-14.79	0.00	0.00	3.20	0.90
19	1.0286	-14.91	0.00	0.00	9.50	3.40
20	1.0329	-14.69	0.00	0.00	2.20	0.70
21	1.0400	-13.86	0.00	0.00	7.50	11.20
22	1.0398	-13.93	0.00	0.00	0.00	0.00
23	1.0300	-14.56	0.00	0.00	3.20	1.60
24	1.0257	-14.59	0.00	0.00	8.70	6.70
25	1.0194	-14.53	0.00	0.00	0.00	0.00
26	1.0018	-14.94	0.00	0.00	3.50	2.30
27	1.0241	-14.23	0.00	0.00	0.00	0.00
28	1.0085	-10.78	0.00	0.00	0.00	0.00
29	1.0043	-15.46	0.00	0.00	2.40	0.90
30	0.9938	-16.34	0.00	0.00	10.60	1.90



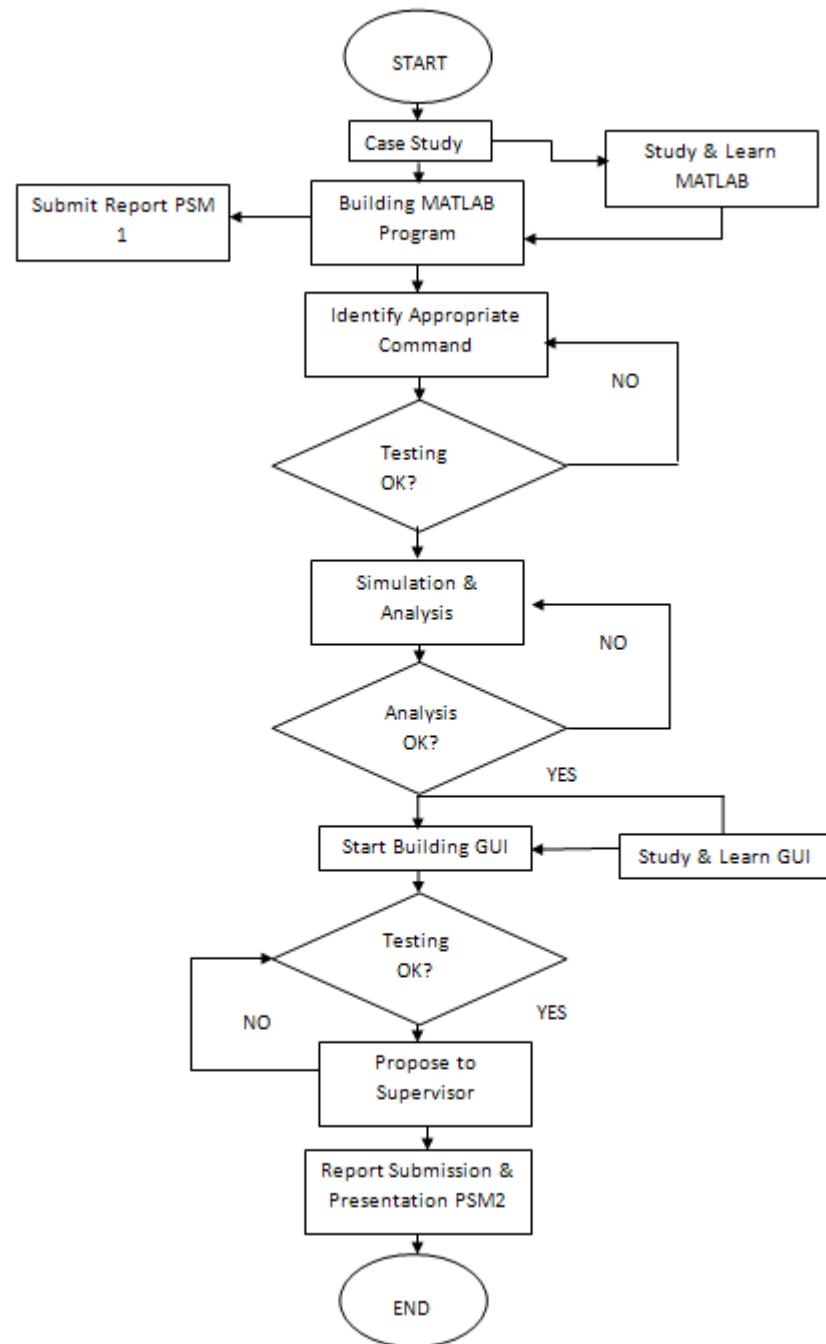
## APPENDIX C

## Line Data

Line number and between buses		Line impedance		Half-line charging susceptance, $B/2$ (pu)
Line No.	Buses	$R$ (pu)	$X$ (pu)	
1	1 and 2	0.0192	0.0575	0.0264
2	1 and 3	0.0452	0.1852	0.0204
3	2 and 4	0.0570	0.1737	0.0184
4	3 and 4	0.0132	0.0379	0.0042
5	2 and 5	0.0472	0.1983	0.0209
6	2 and 6	0.0581	0.1763	0.0187
7	4 and 6	0.0119	0.4140	0.0045
8	5 and 7	0.0460	0.1160	0.0102
9	6 and 7	0.0267	0.0820	0.0085
10	6 and 8	0.0120	0.0420	0.0045
11	9 and 11	0.0000	0.2080	0.0000
12	9 and 10	0.0000	0.1100	0.0000
13	12 and 13	0.0000	0.1400	0.0000
14	12 and 14	0.1231	0.2559	0.0000
15	12 and 15	0.0662	0.1304	0.0000
16	12 and 16	0.0945	0.1987	0.0000
17	14 and 15	0.2210	0.1997	0.0000
18	16 and 17	0.0824	0.1932	0.0000
19	15 and 18	0.1070	0.2185	0.0000
20	18 and 19	0.0639	0.1292	0.0000
21	19 and 20	0.0340	0.0680	0.0000
22	10 and 20	0.0936	0.2090	0.0000
23	10 and 17	0.0324	0.0845	0.0000
24	10 and 21	0.0348	0.0749	0.0000
25	10 and 22	0.0727	0.1499	0.0000
26	21 and 22	0.0116	0.0236	0.0000
27	15 and 23	0.1000	0.2020	0.0000
28	22 and 24	0.1150	0.1790	0.0000
29	23 and 24	0.1320	0.2700	0.0000
30	24 and 25	0.1885	0.3292	0.0000
31	25 and 26	0.2544	0.3800	0.0000
32	25 and 27	0.1093	0.2087	0.0000
33	27 and 29	0.2198	0.4153	0.0000
34	27 and 30	0.3202	0.6027	0.0000
35	29 and 30	0.2399	0.4533	0.0000
36	8 and 28	0.0363	0.2000	0.0214
37	6 and 28	0.0169	0.0599	0.0065

## APPENDIX D

## Flow Chart



## APPENDIX E

## Programming for the MAIN\_PAGE

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%           Design By: SUHAIRAH BINTI RAZALI           EC07082           %
%           Degree of Electrical Engineering (Power System)           %
%           UNIVERSITI MALAYSIA PAHANG                   %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function varargout = MAIN_PAGE(varargin)
% MAIN_PAGE M-file for MAIN_PAGE.fig
%   MAIN_PAGE, by itself, creates a new MAIN_PAGE or raises the
existing
%   singleton*.
%
%   H = MAIN_PAGE returns the handle to a new MAIN_PAGE or the
handle to
%   the existing singleton*.
%
%   MAIN_PAGE('CALLBACK',hObject,eventData,handles,...) calls the
local
%   function named CALLBACK in MAIN_PAGE.M with the given input
arguments.
%
%   MAIN_PAGE('Property','Value',...) creates a new MAIN_PAGE or
raises the
%   existing singleton*. Starting from the left, property value
pairs are
%   applied to the GUI before MAIN_PAGE_OpeningFcn gets called. An
%   unrecognized property name or invalid value makes property
application
%   stop. All inputs are passed to MAIN_PAGE_OpeningFcn via
varargin.
%
%   *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only
one
%   instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help MAIN_PAGE

% Last Modified by GUIDE v2.5 23-Nov-2009 00:00:27

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                  'gui_Singleton',   gui_Singleton, ...
                  'gui_OpeningFcn', @MAIN_PAGE_OpeningFcn, ...
                  'gui_OutputFcn',  @MAIN_PAGE_OutputFcn, ...

```

```

        'gui_LayoutFcn', [] , ...
        'gui_Callback', []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before MAIN_PAGE is made visible.
function MAIN_PAGE_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to MAIN_PAGE (see VARARGIN)

% Choose default command line output for MAIN_PAGE
handles.output = hObject;

axes(handles.axes1)
[x,map]=imread('ump','jpg');
    image(x)
    set(gca,'visible','off')

axes(handles.axes2)
[x,map]=imread('drahmed','jpg');
    image(x)
    set(gca,'visible','off')

axes(handles.axes3)
[x,map]=imread('suhairah','jpg');
    image(x)
    set(gca,'visible','off')

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes MAIN_PAGE wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = MAIN_PAGE_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure

```

```
varargout{1} = handles.output;
```

```
% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
user_response = INTRODUCTION, close MAIN_PAGE;
```

```
% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
user_response = SIMULATION, close MAIN_PAGE;
```

```
% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
button=questdlg('Are you sure you want to
exit?', 'exit', 'Yes', 'No', 'No');
switch button
    case 'Yes'
        close all;
    case 'No'
        quite cancel;
end
```

## APPENDIX F

## Programming for the INTRODUCTION

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%           Design By: SUHAIRAH BINTI RAZALI           EC07082           %
%           Degree of Electrical Engineering (Power System)           %
%           UNIVERSITI MALAYSIA PAHANG                   %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function varargout = INTRODUCTION(varargin)
% INTRODUCTION M-file for INTRODUCTION.fig
%   INTRODUCTION, by itself, creates a new INTRODUCTION or raises
the existing
%   singleton*.
%
%   H = INTRODUCTION returns the handle to a new INTRODUCTION or the
handle to
%   the existing singleton*.
%
%   INTRODUCTION('CALLBACK',hObject,eventData,handles,...) calls the
local
%   function named CALLBACK in INTRODUCTION.M with the given input
arguments.
%
%   INTRODUCTION('Property','Value',...) creates a new INTRODUCTION
or raises the
%   existing singleton*. Starting from the left, property value
pairs are
%   applied to the GUI before INTRODUCTION_OpeningFcn gets called. An
unrecognized property name or invalid value makes property
application
%   stop. All inputs are passed to INTRODUCTION_OpeningFcn via
varargin.
%
%   *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only
one
%   instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help INTRODUCTION

% Last Modified by GUIDE v2.5 23-Nov-2009 00:32:32

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                  'gui_Singleton',   gui_Singleton, ...
                  'gui_OpeningFcn', @INTRODUCTION_OpeningFcn, ...
                  'gui_OutputFcn',  @INTRODUCTION_OutputFcn, ...
                  'gui_LayoutFcn',  [], ...

```

```

        'gui_Callback', []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before INTRODUCTION is made visible.
function INTRODUCTION_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to INTRODUCTION (see VARARGIN)

% Choose default command line output for INTRODUCTION
handles.output = hObject;

axes(handles.axes1)
[x,map]=imread('ump','jpg');
image(x)
set(gca,'visible','off')

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes INTRODUCTION wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = INTRODUCTION_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
user_response = MAIN_PAGE, close INTRODUCTION;

```

## APPENDIX G

## Programming for the SIMULATION

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%           Design By: SUHAIRAH BINTI RAZALI           EC07082           %
%           Degree of Electrical Engineering (Power System)           %
%           UNIVERSITI MALAYSIA PAHANG                   %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function varargout = SIMULATION(varargin)
% SIMULATION M-file for SIMULATION.fig
%     SIMULATION, by itself, creates a new SIMULATION or raises the
existing
%     singleton*.
%
%     H = SIMULATION returns the handle to a new SIMULATION or the
handle to
%     the existing singleton*.
%
%     SIMULATION('CALLBACK',hObject,eventData,handles,...) calls the
local
%     function named CALLBACK in SIMULATION.M with the given input
arguments.
%
%     SIMULATION('Property','Value',...) creates a new SIMULATION or
raises the
%     existing singleton*. Starting from the left, property value
pairs are
%     applied to the GUI before SIMULATION_OpeningFcn gets called. An
%     unrecognized property name or invalid value makes property
application
%     stop. All inputs are passed to SIMULATION_OpeningFcn via
varargin.
%
%     *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only
one
%     instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help SIMULATION

% Last Modified by GUIDE v2.5 23-Nov-2009 01:21:20

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                  'gui_Singleton',   gui_Singleton, ...

```



```

        'gui_OpeningFcn', @SIMULATION_OpeningFcn, ...
        'gui_OutputFcn', @SIMULATION_OutputFcn, ...
        'gui_LayoutFcn', [] , ...
        'gui_Callback', []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before SIMULATION is made visible.
function SIMULATION_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles     structure with handles and user data (see GUIDATA)
% varargin    command line arguments to SIMULATION (see VARARGIN)

movegui ('center')
handles.output = 'Yes';

% Choose default command line output for SIMULATION
handles.output = hObject;

axes(handles.axes1)
[x,map]=imread('ump','jpg');
    image(x)
    set(gca,'visible','off')

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes SIMULATION wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = SIMULATION_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles     structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

```

```
% --- Executes on button press in pushbutton1.  
function pushbutton1_Callback(hObject, eventdata, handles)  
% hObject    handle to pushbutton1 (see GCBO)  
% eventdata  reserved - to be defined in a future version of MATLAB  
% handles    structure with handles and user data (see GUIDATA)  
user_response = run;
```

## APPENDIX H

## Programming for the PSOsimulation

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%           Design By: SUHAIRAH BINTI RAZALI           EC07082           %
%           Degree of Electrical Engineering (Power System)           %
%           UNIVERSITI MALAYSIA PAHANG                   %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Particle Swarm Optimization Simulation
% Simulates the movements of a swarm to minimize the objective function
%
% $$ \left( x-15 \right) ^{2}+ \left( y-20 \right) ^{2} = 0 $$
%
% The swarm matrix is
%
% swarm(index, [location, velocity, best position, best
% value], [x, y components or the value component])
%
% Author: Wesam ELSHAMY (wesamelshamy@yahoo.com)
% MSc Student, Electrical Engineering Dept., Faculty of Engineering
% Cairo University, Egypt
%%

%% Initialization
% Parameters
clear
clc
iterations = 30;
inertia = 1.0;
correction_factor = 2.0;
swarm_size = 49;

% ---- initial swarm position ----
index = 1;
for i = 1 : 7
    for j = 1 : 7
        swarm(index, 1, 1) = i;
        swarm(index, 1, 2) = j;
        index = index + 1;
    end
end

swarm(:, 4, 1) = 1000;           % best value so far
swarm(:, 5, :) = 0;             % initial velocity

%% Iterations
for iter = 1 : iterations

    %-- evaluating position & quality ---

```

```

    for i = 1 : swarm_size
        swarm(i, 1, 1) = swarm(i, 1, 1) + swarm(i, 2, 1)/1.3;
%update x position
        swarm(i, 1, 2) = swarm(i, 1, 2) + swarm(i, 2, 2)/1.3;
%update y position
        x = swarm(i, 1, 1);
        y = swarm(i, 1, 2);

        val = (x - 15)^2 + (y - 20)^2;           % fitness evaluation
        (you may replace this objective function with any function having a
        global minima)

        if val < swarm(i, 4, 1)                 % if new position is
better
            swarm(i, 3, 1) = swarm(i, 1, 1);    % update best x,
            swarm(i, 3, 2) = swarm(i, 1, 2);    % best y postions
            swarm(i, 4, 1) = val;               % and best value
        end
    end

    [temp, gbest] = min(swarm(:, 4, 1));        % global best position

    %--- updating velocity vectors
    for i = 1 : swarm_size
        swarm(i, 2, 1) = rand*inertia*swarm(i, 2, 1) +
correction_factor*rand*(swarm(i, 3, 1) - swarm(i, 1, 1)) +
correction_factor*rand*(swarm(gbest, 3, 1) - swarm(i, 1, 1));    %x
velocity component
        swarm(i, 2, 2) = rand*inertia*swarm(i, 2, 2) +
correction_factor*rand*(swarm(i, 3, 2) - swarm(i, 1, 2)) +
correction_factor*rand*(swarm(gbest, 3, 2) - swarm(i, 1, 2));    %y
velocity component
    end

    %% Plotting the swarm
    clf
    plot(swarm(:, 1, 1), swarm(:, 1, 2), 'x')    % drawing swarm
movements
    axis([-2 30 -2 30]);
    pause(.2)
end

```

## APPENDIX I

## Programming for the RUN button

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%           Design By: SUHAIRAH BINTI RAZALI           EC07082           %
%           Degree of Electrical Engineering (Power System)           %
%           UNIVERSITI MALAYSIA PAHANG                   %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function varargout = run(varargin)
% RUN M-file for run.fig
%     RUN, by itself, creates a new RUN or raises the existing
%     singleton*.
%
%     H = RUN returns the handle to a new RUN or the handle to
%     the existing singleton*.
%
%     RUN('CALLBACK',hObject,eventData,handles,...) calls the local
%     function named CALLBACK in RUN.M with the given input arguments.
%
%     RUN('Property','Value',...) creates a new RUN or raises the
%     existing singleton*. Starting from the left, property value
pairs are
%     applied to the GUI before run_OpeningFcn gets called. An
%     unrecognized property name or invalid value makes property
application
%     stop. All inputs are passed to run_OpeningFcn via varargin.
%
%     *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only
one
%     instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help run

% Last Modified by GUIDE v2.5 22-Nov-2009 13:32:15

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                  'gui_Singleton',   gui_Singleton, ...
                  'gui_OpeningFcn', @run_OpeningFcn, ...
                  'gui_OutputFcn',  @run_OutputFcn, ...
                  'gui_LayoutFcn',   [] , ...
                  'gui_Callback',    []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else

```

```

        gui_mainfcn(gui_State, varargin{:});
    end
    % End initialization code - DO NOT EDIT

    % --- Executes just before run is made visible.
    function run_OpeningFcn(hObject, eventdata, handles, varargin)
    % This function has no output args, see OutputFcn.
    % hObject    handle to figure
    % eventdata  reserved - to be defined in a future version of MATLAB
    % handles    structure with handles and user data (see GUIDATA)
    % varargin   command line arguments to run (see VARARGIN)

    movegui ('center')
    handles.output = 'Yes';

    % Choose default command line output for run
    handles.output = hObject;

    % Update handles structure
    guidata(hObject, handles);

    % UIWAIT makes run wait for user response (see UIRESUME)
    % uiwait(handles.figure1);

    % --- Outputs from this function are returned to the command line.
    function varargout = run_OutputFcn(hObject, eventdata, handles)
    % varargout  cell array for returning output args (see VARARGOUT);
    % hObject    handle to figure
    % eventdata  reserved - to be defined in a future version of MATLAB
    % handles    structure with handles and user data (see GUIDATA)

    % Get default command line output from handles structure
    varargout{1} = handles.output;
    handles.output = 'No';

    % --- Executes on button press in pushbutton1.
    function pushbutton1_Callback(hObject, eventdata, handles)
    % hObject    handle to pushbutton1 (see GCBO)
    % eventdata  reserved - to be defined in a future version of MATLAB
    % handles    structure with handles and user data (see GUIDATA)
    user_response = 'Yes';
    switch user_response
        case 'Yes', close, PSOSimulation;
    end

    % --- Executes on button press in pushbutton2.
    function pushbutton2_Callback(hObject, eventdata, handles)
    % hObject    handle to pushbutton2 (see GCBO)
    % eventdata  reserved - to be defined in a future version of MATLAB

```

```
% handles      structure with handles and user data (see GUIDATA)
user_response = 'No'
switch user_response
    case 'No' , close ;
end
```

## APPENDIX J

## Programming to find the optimal location

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%           Design By: SUHAIRAH BINTI RAZALI           EC07082           %
%           Degree of Electrical Engineering (Power System)           %
%           UNIVERSITI MALAYSIA PAHANG                   %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```
function fpbest = PSOFACTS(xpbest,Particles,d)
```

```

% IEEE 30-BUS TEST SYSTEM (American Electric Power)
basemva=100; accuracy=0.001; accel=1.8; maxiter=100;
% Bus Bus Voltage Angle --Load-- ---Generator---
% Injected
% No Code Mag. Degree MW Mvar MW Mvar Qmin Qmax
Mvar
busdata=[ 1 1 1.06 0 0.0 0.0 0.0 0.0 0 0
0;
2 2 1.043 0 21.70 12.7 40.0 0.0 -40 50
0;
3 0 1.0 0 2.4 1.2 0.0 0.0 0 0
0;
4 0 1.06 0 7.6 1.6 0.0 0.0 0 0
0;
5 2 1.01 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;
7 0 1.0 0 22.8 10.9 0.0 0.0 0 0
0;
8 2 1.01 0 30.0 30.0 0.0 0.0 -10 40
0;
9 0 1.0 0 0.0 0.0 0.0 0.0 0 0
0;
10 0 1.0 0 5.8 2.0 0.0 0.0 0 0
19;
11 2 1.082 0 0.0 0.0 0.0 0.0 -6 24
0;
12 0 1.0 0 11.2 7.5 0.0 0.0 0 0
0;
13 2 1.071 0 0.0 0.0 0 0 -6 24
0;
14 0 1.0 0 6.2 1.6 0 0 0 0
0;
15 0 1.0 0 8.2 2.5 0 0 -6 24
0;
16 0 1.0 0 3.5 1.8 0 0 0 0
0;
17 0 1.0 0 9.0 5.8 0 0 0 0
0;
18 0 1.0 0 3.2 0.9 0 0 0 0
0;

```



```

0;      19  0    1.0    0    9.5  3.4    0    0    0    0
0;      20  0    1.0    0    2.2  0.7    0    0    0    0
0;      21  0    1.0    0   17.5 11.2    0    0    0    0
0;      22  0    1.0    0    0.0  0.0    0    0    0    0
0;      23  0    1.0    0    3.2  1.6    0    0    0    0
0;      24  0    1.0    0    8.7  6.7    0    0    0    0
4.3;    25  0    1.0    0    0.0  0.0    0    0    0    0
0;      26  0    1.0    0    3.5  2.3    0    0    0    0
0;      27  0    1.0    0    0.0  0.0    0    0    0    0
0;      28  0    1.0    0    0.0  0.0    0    0    0    0
0;      29  0    1.0    0    2.4  0.9    0    0    0    0
0;      30  0    1.0    0   10.6  1.9    0    0    0    0
0];

```

```

% Line Data
%
%      Bus  Bus  R      X      1/2B      for Line code or
%      n1   nr   pu      pu      pu      tap setting value
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 0.0      0.5560  0.0      0.969;
9  11 0.0      0.2080  0.0      1;
9  10 0.0      0.1100  0.0      1;
4  12 0.0      0.2560  0.0      0.932;
12 13 0.0      0.1400  0.0      1;
12 14 0.1231  0.2559  0.0      1;
12 15 0.0662  0.1304  0.0      1;
12 16 0.0945  0.1987  0.0      1;
14 15 0.2210  0.1997  0.0      1;
16 17 0.0824  0.1923  0.0      1;
15 18 0.1073  0.2185  0.0      1;
18 19 0.0639  0.1292  0.0      1;
19 20 0.0340  0.0680  0.0      1;
10 20 0.0936  0.2090  0.0      1;
10 17 0.0324  0.0845  0.0      1;
10 21 0.0348  0.0749  0.0      1;
10 22 0.0727  0.1499  0.0      1;

```

```

21    22    0.0116    0.0236    0.0        1;
15    23    0.1000    0.2020    0.0        1;
22    24    0.1150    0.1790    0.0        1;
23    24    0.1320    0.2700    0.0        1;
24    25    0.1885    0.3292    0.0        1;
25    26    0.2544    0.3800    0.0        1;
25    27    0.1093    0.2087    0.0        1;
28    27    0.0000    0.3960    0.0        0.968;
27    20    0.2198    0.4153    0.0        1;
27    30    0.3202    0.6027    0.0        1;
29    30    0.2399    0.4533    0.0        1;
8     28    0.0636    0.2000    0.0214    1;
6     28    0.0169    0.0599    0.065     1];

%
%
nbus = length(busdata(:,1)); % No. of Transmission Line
nbr =length(linedata(:,1)); %No. of transmission line

New=busdata;

for t1=1:Particles
    lo_tcsc=xpbest(t1,1);
    lo=lo_tcsc;
    % add TCSC location
    X1=linedata(lo,1);
    X2=linedata(lo,2);
    % Add TCSC MODEL
    xc=0.0575;
    rij=linedata(t1,3);
    xij=linedata(t1,4);
    Gij=xc*(rij^2-xij^2+xc*xij)/(rij^2+xij^2)*(rij^2+(xij-xc)^2);
    Bij=xc*rij*(xc-xij)/(rij^2+xij^2)*(rij^2+(xij-xc)^2);
    Vi=busdata(X1,3);
    di=busdata(X1,4);
    Vj=busdata(X2,3);
    dj=busdata(X2,4);
    Pi=Vi^2*Gij-Vi*Vj*(Gij*cos(di-dj)+Bij*sin(di-dj));
    Qi=-Vi^2*Bij-Vi*Vj*(Gij*sin(di-dj)+Bij*cos(di-dj));
    Pj=Vj^2*Gij-Vi*Vj*(Gij*cos(di-dj)-Bij*sin(di-dj));
    Qj=-Vj^2*Gij+Vi*Vj*(Gij*sin(di-dj)+Bij*cos(di-dj));
    %insert TCSC model
    busdata(X1,6)=busdata(X1,6)+Qi;
    busdata(X1,7)=busdata(X1,7)+Pi;
    busdata(X2,6)=busdata(X2,6)+Qj;
    busdata(X2,7)=busdata(X2,7)+Pj;

    % add UPFC
    lo_upfc=xpbest(t1,2);
    lou=lo_upfc;
    % add UPFC location
    X3=linedata(lou,1);
    X4=linedata(lou,2);
    %Add UPFC MODEL
    sr=2.4;
    sbse=0.0001;
    sgamma=5;

```

```

    sthetai=0;
    sthetaj=0;
    sPiupfc=(0.02*r*bse*Vi^2*sin*gamma)-1.02*r*bse*Vi*Vj*sin*(thetai-
thetaj+gamma);
    sQiupfc=-r*bse*Vi^2*cos*gamma;
    sPjupfc=r*bse*Vi*Vj*sin*(thetai-thetaj+gamma);
    sQjupfc=r*bse*Vi*Vj*cos*(thetai-thetaj+gamma);
    %
    Pupfc=0.2;
    Qupfc=0.5;
%insert UPFC model
busdata(X3,6)=busdata(X1,6)+Qupfc;
busdata(X3,7)=busdata(X1,7)+Pupfc;
busdata(X4,6)=busdata(X2,6)+Qupfc;
busdata(X4,7)=busdata(X2,7)+Pupfc;

%[Ybus]=lfybus(linedata);
%[Vm,delta,P,Q,S,VBc,a,nbr,nbus,nr,nl]=lfnewton(busdata,linedata);
%[Snkr,Snki]=lineflow(busdata,linedata,Vm,delta,P,Q,S,V,Bc,nr,nl,basemv
a);
lfybus;           %Forms the bus admittance matrix
lfnewton;         %Power flow solution by newton method
busout;           %Print the power flow solution on the screen
lineflow;        %Computes and displays the line flow and losses
x1=Vm;
x2=deltad;
% x3=P;
% x4=Q;
% neural output
for t2=1:nbus
    tt(t2,t1)=x1(t2);    % Vm
    tt1(t2,t1)=x2(t2)*3.14/180; % deltad
end
% pp(1,t1)=Pd; % total power losses
% Remove TCSC and UPFC
busdata=New;
end
fpbest(t1)=min(max(tt));
disp('Testing has been carried out successfully');
end

```

## APPENDIX K

## Programming to find the optimal location

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%           Design By: SUHAIRAH BINTI RAZALI           EC07082           %
%           Degree of Electrical Engineering (Power System)           %
%           UNIVERSITI MALAYSIA PAHANG                   %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% SWARM Swarm minimization
%%
% [X,BEST]=swarm(Particles,d,Range,n,C0,C1,C2,FUN)
% minimizes function FUN using Swarm Minimization.
%Particles: number of creatures in swarm
(maxgen)
% d: number of Parameter
% Range: range of initial uniform distribution of creature position
% N: number of steps ( iterations )
(Popnumber)
% C0: influence of old velocity to new velocity           (0.66)
% C1: influence of personal best vector two new velocity   (2)
% C2: influence of collective's best vector two new velocity (2)
% FUN: string containing name of function to minimize.
%%
%Return X (Particles,n,d) vector with swarm locations, BEST collective
% value
%% d= two device FACTS
Particles=10; d=2; n=25; Range=[1 41;1 41]; c0=0.66; c1=2; c2=2;
% function [xx,best] = swarm1(Particles, d, Range, n, c0, c1 , c2)
xx = [];
%Handle=waitbar(0,'Please wait...');
%Initialization of PSO parameters
%wmax=0.9;
%wmin=0.4;
%for iter=1:n
%W(iter)=wmax-((wmax-wmin)/n)*iter;
%end
% choose initial position and location of particles
%Initialization of positions of agents
%%%%%% agents are initialized between Range(i,2),Range(i,1) randomly
for i=1:d
    a=Range(i,1)+(Range(i,2)-Range(i,1))*rand(Particles,1);
    for j=1:Particles
        a1(j,i)=a(j);
    end
end
% Find integer No.
for i1=1:d
    for j1=1:Particles
        n22=a1(j1,i1);
        a1(j1,i1)=ceil(n22.*rand(1,1));
    end
end

%%%%v=rand(count,d)*vstddev;

```

```

%Initialization of velocities of agents
%Between c3=0,c4=0.66
c3=0;
c4=0.1;
for i=1:d
v=c3+(c4-c3)*rand(Particles,1);
for j=1:Particles
v1(j,i)=v(j);
end
end
% compute personal and global optima for initial configuration
xpbest = a1; % initial vector with personal optima locations
%fpbest = feval(fun, xpbest); % initial vector with personal optima
values
fpbest = PSOFACKS(xpbest,Particles,d); % initial vector with personal
optima values
[ fgbest , xgbest] = min(fpbest); % global best optima value
xgbest = xpbest(xgbest,:); % global best optimal location
%P=ones(count,1)*xpbest(xgbest,:);

for i=1:n % iteration loop
% change velocity
v1 =c0*v1+c1*repmat(rand(Particles, 1), 1,d).*(xpbest-a1)+...
c2*repmat(rand(Particles, 1),1,d).*(repmat(xgbest, Particles, 1)-a1);

for i=1:d
v1(:,i)=min(v1(:,i),ones(Particles,1)*0.2);
v1(:,i)=max(v1(:,i),ones(Particles,1)*0.0001);
end
a1 = a1 + v1; % update positions
m=a1;
% to fly back by indentification parameter to limited range
for i=1:d
a1(:,i)=min(a1(:,i),ones(Particles,1)*Range(i,2));
a1(:,i)=max(a1(:,i),ones(Particles,1)*Range(i,1));
end

for i1=1:d
for j1=1:Particles
n22=a1(j1,i1);
a1(j1,i1)=ceil(n22.*rand(1,1));
end
end
f= PSOFACKS(a1,Particles,d); % evaluate function at new swarm positions
% check for new individual optima (each particle seperately)
haschanged = find(min(f,fpbest)<fpbest);
k=size(haschanged,2);
for h=1:k
xpbest(haschanged(h),:) = a1(haschanged(h),:);
fpbest(haschanged(h)) = f(haschanged(h));
end
% check for new collective optimum
if (min(fpbest)<fgbest)
[ fgbest ,newbest] = min(fpbest);
xgbest = xpbest(newbest,:);
end

```

```
%end
xx=xgbest; % append current swarm positions (best FACTS place for TCSC
and UPFC)
best = fgbest;
disp('Type xpbest to see the particle solution');
disp('Type xgbest to see the global solution');
```

