A Genetic Algorithm Approach In Load Flow Analysis

By

CHINTAN DWIVEDI

11MEEE03

DEPARTMENT OF ELECTRICAL ENGINEERING

INSTITUTE OF TECHNOLOGY

NIRMA UNIVERSITY

AHMEDABAD-382481

MAY 2013

A Genetic Algorithm Approach In Load Flow Analysis

Major Project Report

Submitted in Partial Fulfillment of the Requirements

For the degree of

MASTER OF TECHNOLOGY

IN

ELECTRICAL ENGINEERING

(ELECTRICAL POWER SYSTEMS)

By

CHINTAN DWIVEDI

11MEEE03

DEPARTMENT OF ELECTRICAL ENGINEERING

INSTITUTE OF TECHNOLOGY

NIRMA UNIVERSITY

AHMEDABAD-382481

MAY 2013

Undertaking for Originality of the Work

I **Mr.CHINTAN DWIVEDI**, Roll.No.**11MEEE03** give undertaking that the Major Project entitled **"A Genetic Algorithm Approach In load Flow Analysis"**, submitted by me, towards the partial fulfillment of the requirements or the degree of Master of Technology in **Electrical power Systems** of Nirma University, Ahmedabad,is the original work carried out by me and I give assurance that no attempt of plagiarism has been made. I understand that in the event of any similarity found subsequently with any published work or any dissertation work elsewhere; it will result in severe disciplinary action

.....

Signature of Student

Date:

Place:

.....

Endorsed by

Prof.Shankar Godwal

Department of Electrical Engineering

Institute of Technology

Nirma University

Ahmedabad

Certificate

This is to certify that the Major Project Report entitled **"A Genetic Algorithm Approach In Load Flow Analysis"** submitted by Mr.**CHINTAN DWIVEDI**(**Roll No: 11MEEE03**) towards the partial fulfillment of the requirements for the award of Degree in Master of Technology (Electrical Engineering) in the field of Electrical Power Systems of Nirma University is the record of work carried out by him under our supervision and guidance. The work submitted has in our opinion reached a level required for being accepted for examination. The results embodied in this major project work to the best of our knowledge have not been submitted to any other University or Institution for award of any Degree or Diploma.

Date:

.....

Project Guide:

Prof.Shankar Godwal

Department of Electrical Engineering

Institute of Technology

Nirma University

Ahmedabad

Head of Department

Department of Electrical Engineering

Institute of Technology

Nirma University

Ahmedabad

Director

Institute of Technology

Nirma University

Ahmedabad

Acknowledgements

I am highly grateful to **Prof.Shankar Godwal**,Department of Electrical Engineering,Institute of Technology,Nirma University for their sincere advice ,encouragement and continuous guidance in my work.

I express my sincere appreciation and thanks to **Dr.S.C.Vora** for constant encouragement and needful help during various stages of work. I warmly acknowledge and express my special thanks for their inspiring discussions and infallible suggestions.

I am very much obliged to for guiding throughout the work. My sincere thanks to **Dr.P.N.Tekwani**,Head of Electrical Engineering Department. I would also like to thank **Dr.K.Kotecha**,Director of Institute of Technology,Nirma university for allowing me to carry out my project work in Institute I am also thankful to Nirma University for providing all kind of required resources. Also I would like to thank all the other faculty members of Department of Electrical Engineering ,Nirma University

I would also like to thank my friends who have provided their continuous encouragement and support.I am also thankful to all those who have helped me directly or indirectly during the project work.

At the last but not the least I would like to express my gratitude to the Almighty and my parents.I would like to say that the cooperation made by everybody in the completion of this report would be remember.

Chintan Dwivedi

11MEEE03

Abstract

The Load Flow problem is highly Non-linear and Multi -modal Optimization problem. Conventional optimization methods that make use of derivative and gradient optimization may not be able to identify global optimization, hence it is desired to develop optimization techniques that are efficient to overcome these deficiencies.The advantages of evolutionary computing techniques are that, these are relatively versatile for handling various quantitative constraints.

The Project entitled Genetic Algorithm will encoded and applied to solve multiple load flow solution problem. Genetic algorithm is a kind of stochastic search algorithm based on the mechanics of natural selection and natural genetics. They combine the concepts of survival of the fittest with genetic operators such as selection, crossover and mutation abstracted from nature to form a surprisingly robust mechanism that has been successfully applied to solve a variety of search and optimization problems.

The main Objective of this desertion is the Real Encoded Genetic Algorithm in MATLAB and implemented to 5 bus and IEEE14 bus system.A floating number representation instead of the binary number representation is introduce for the accuracy.The results are compared with the conventional methods i.e Newton-Raphson(NR).

Contents

Undertaking for Originality of the Work iii

Certificate iv

Acknowledgements v

Abstract vi

List of Figures ix

List of Tables x

Abbreviation Notation and Nomenclature 1

1 Introduction 1
- 1.1 Background . . . 1
- 1.2 Literature Review . . . 2
- 1.3 Problem Identification . . . 5
- 1.4 Objective Of the Project . . . 6
- 1.5 Outline Of Thesis . . . 6

2 Load Flow Analysis 8
- 2.1 Objective Of LF study . . . 9
- 2.2 Basic techniques for power-flow studies . . . 10
  - 2.2.1 Types Of Buses . . . 11
  - 2.2.2 General power Formulation . . . 12
- 2.3 Iterative Numerical (LF) solution methods . . . 12

3 Genetic Algorithm 17
- 3.1 Introduction To GA . . . 17
  - 3.1.1 Basic Terminology In GA [11][13] . . . 18
  - 3.1.2 Out line of the Basic GA . . . 19
- 3.2 Binary V/S Floating Coding Approach . . . 23
  - 3.2.1 Components Of a Continuous Genetic Algorithm . . . 23

4 GA In Load Flow Analysis **29**
4.1 Design of GA components 29
4.1.1 Steps 30

5 Coding Implementation And Results **37**
5.1 Testing of method on 5 Bus system 39
5.2 IEEE 14 Bus System 44

6 Conclusion and Future work **48**
6.1 Conclusion 48
6.2 Future work 49

A Introduction to MATLAB7.9.0 R2009a GA Tool Box **51**

B GA programming **54**

References **57**

Index **59**

List of Figures

2.1 Single-Variable Linear Approximation [Taylor 1967] 14

3.1 Problem solution using evolutionary Algorithms 17
3.2 GA Program Flow Chart 20

4.1 Generalized code for formulation of Ybus 30
4.2 Chromosome representation 31
4.3 Chromosome generation 31
4.4 Declaring the parameter 32
4.5 Objective function 32
4.6 Objective function 33
4.7 Fitness function calling 33
4.8 parent 34
4.9 offspring 34
4.10 offspring 34
4.11 Single point Cross over 35
4.12 Mutation the Voltage and Delta 35
4.13 Stopping criteria 36

5.1 Simple System 37
5.2 The change of determinant of Jacobian matrix 38
5.3 5 bus system 39
5.4 Real and reactive power at load buses 39
5.5 Line Impedances 40
5.6 Admittance Matrix 40
5.7 P and Q mismatch at bus2 41
5.8 P and Q mismatch at bus3 41
5.9 P and Q mismatch at bus4 42
5.10 P mismatch at 5 42
5.11 Objective And fitness function 43
5.12 Chromosome 43
5.13 Objective Function 45
5.14 Objective Function V/s Generation 47

List of Tables

I Comparison of load flow methods 16

I Encoding types 21
II Study of different Mutation operator 22

I The result and determinant of normal case 38
II The result and determinant of Heavy load 38
III Comparison between GA Solution and NR solution 44

Abbreviation

GA ..... Genetic Algorithm
LF ..... Load Flow
OPF ..... Optimal Power Flow
EP ..... Evolutionary programming
EC ..... Evolutionary computation
AI ..... Artificial Intelligence
GALF ..... Genetic Algorithm Load Flow
GS ..... Gauss-Seidel
NR ..... Newton Raphson
FDLF ..... Fast Decoupled Load Flow
P ..... Real Power
Q ..... Reactive Power
NRLF ..... Newton Raphson Load Flow
δ ..... Load Angle

Nomenclature

$x_{i+m,j}$ ..... Mutated parents (offspring)
$x_{i,j}$ ..... Parents
N ..... Gaussian random variable with mean,μ and variance,γ^2
β ..... Mutation scale 0 to 1
x_{jmax} ..... Maximum random number
x_{jmin} ..... Minimum random number
f_i ..... Fitness for the i^{th}random number
f_{max} ..... Maximum fitness
$\Delta P_i^{(k)}$ ..... mismatch in real power
$\Delta Q_i^{(k)}$ ..... mismatch in reactive power
Y ..... Admittance matrix
P^{sch} ..... Schedule real power
Q^{sch} ..... Schedule reactive power

Prefixes

Δ ..... Difference or Error

Chapter 1

Introduction

1.1 Background

The load flow studies are the backbone of the design of a power system. They are the means by which the future operation of the system is known ahead of time. The load flow problem is one of the basic problems in the power system engineering, and can be expressed as a set of non-linear simultaneous algebraic equations, and thus it is to have multiple solutions. A load flow study is the determination of many parameter such as voltage, current, power, and power factor or reactive power at various points in an electrical network under existing or contemplated conditions of normal operation, so power flow calculations provide power flows and voltages for a specified power system subject to the regulating capability of generators, condensers, and tap changing under load transformers as well as specified net interchange between individual operating systems.

This information is essential for the continuous evaluation of the current performance of a power system and for analyzing the effectiveness of alternative plans for system expansion to meet increased load demand.The load flow is the most frequently carried out study by power utilities and is required to be performed at almost all the stages of power system planning,optimization,operation,control,and contingency analysis.

1.2 Literature Review

This paper Survey is presented on the currently available numerical techniques for power-system load-flow calculation using the digital computer. The review deals with methods that have received widespread practical application, recent attractive developments, and other methods that have interesting or useful characteristics. The analytical bases, computational requirements, and comparative numerical performances of the methods are discussed. Attention is given to the problems and technique of adjustments in load-flow solutions, and the suit abilities of various methods for modern applications such as security monitoring and optimal load flow are examined. This review is concluded by emphasizing the importance of sparse-matrix techniques in the exploitation and further development of load-flow algorithms. Modern methods using spare-matrix factorization need great attention to programming efficiency if they are to fully realize their considerable advantages over previous methods. The onus is therefore placed on the load-flow analyst to gain sufficient understanding of the details and implication of these techniques[1]

This paper focus on the results of an investigation into the interdependence between the loading conditions and the number of physically possible load flows in a power system are presented.The relation between practically acceptable load flows and the multiplicity of load flow solutions 4 nonlinear load flow equations is presented .Due to the nonlinearity of network equations there are many real load flow solutions. The number of solutions, generally,decreases non-uniformly with increasing nodal powers.The number of P,$|V|$ nodes influences the manner in which the number of solutions decreases.This paper concluded using conventional computational methods one can obtain any not-right solution instead of the right solution. The solution obtained depends very much on the starting point of the iteration process. One of the best methods of avoiding the computation of not right solutions is the inclusion of the power voltage dependence coefficients equal to 1*per*. in the network equations.[2]

This paper introduces genetic algorithms (GA) as a complete entity, in which knowledge of this emerging technology can be integrated together to form the framework of a design tool for industrial engineers. An attempt has also been made to explain why" and when GA should be used as an optimization tool. application basis.The papers purpose is to introduce this emerging technology to engineers who may have little or no knowledge of GA.It is also reasonable to believe that this article contains sufficient, useful material to encourage and awaken the interest to newcomers,so that GA may be implemented to solve their practical problems.

As a challenge to GA is undoubtedly the real-time and adaptive capability. The real-time issue is not so much hinged on the computing speed, since the parallelising of GA should improve the computational speed quite considerably and comfortably. Rather, it is the unpredictability of GA that is the main cause of concern. As GA is a stochastic,discrete, and nonlinear process, guaranteed response times are not applicable.[3]

This paper entitle A new approach based on genetic algorithms to find multiple load flow solutions.Genetic algorithm(GA) are search procedures based on the mechanics of natural genetics and natural selection. A genetic algorithm with sharing scheme is proposed and applied to simultaneously find multiple load flow solutions. Numerical experiments on two example systems show that the proposed algorithm can be used as an efficient guide for simultaneously and globally searching multiple solution regions.At present,the parallel search of solutions is realized on the sequential computers. Since genetic algorithms are highly parallelizing,they involve central challenges to the parallel processing in power system computation [4]

In this paper A real-coded genetic algorithm is presented and applied to solve multiple load flow solution problem. Genetic algorithm is a kind of stochastic search algorithm based on the mechanics of natural selection and natural genetics. They combine the concepts of survival of the fittest with genetic operators such as selec-

tion, crossover and mutation abstracted from nature to form a surprisingly robust mechanism that has been successfully applied to solve a variety of search and optimization problems. Elitist method is also used in this research, and blending models are implemented for crossover operator. In the proposed work, five busbars typical test system are used to demonstrate the efficiency and performance of the proposed method. The results show that, genetic algorithm is on-line load flow solution problem for small-scale power systems, but for large-scale power systems, it is recommended that the load flow solution using genetic algorithm is for planning studies. The main important feature of the purposed method is to give high accurate solution with respect to the conventional methods.[5]

In this paper assessment of different Genetic Algorithm (GA) selection, crossover and mutation techniques in term of convergence to the optimal solution for single objective reactive power optimization problem is presented and investigated. The problem is formulated as a nonlinear optimization problem with equality and inequality constraints. Also, in this paper a simple cost appraisal for the potential annual cost saving of these GA techniques due to reactive power optimization will be conducted. Wale and Hale 6 bus system was used in this study. [6]

This paper presents to minimize the losses using Genetic Algorithm.While minimizing the power losses, the problem is to improve the voltage profile in the power system by using control tool such as generator voltage,shunt VAR and the transformer tap changing.In this paper capacitor consider as the control.Moreovver to locate, the series capacitor, the candidate lines are determined by using sensitive analysis.In this study both active and reactive power losses are minimized and optimum operating conditions of control variables are determined by using GA [7]

This paper presents new theoretical results about the behaviour of the load flow solution near a Jacobian singularity. The principal result is the derivation of an analytic closed form relation between the specified injections and the resulting voltages

in the neighbourhood of a singularity. This result is a companion to the conventional load flow sensitivity analysis which is valid only if the operating point is not at a Jacobian singularity. The new closed form relation derived here is theoretically important since it can predict and explain the main load flow phenomena observed through simulation analysis near a singularity.These are:(i)the non-existence of solutions for certain injection changes, (ii)the bifurcation of the voltages into two nearby solutions,(iii)The sudden collapse of voltages for small injection changes,(iv)the nature of the collapse, that is,which buses are more susceptible to the collapse.Numerical simulations support the validity of the theoretical result by comparing the closed form analytic relation near a singularity with exact load flow simulations. [8]

Some of open source softwares are available on the internet. This software provides complete packages based on Newton's approach for the optimal power flow problem solutions. In this work MATPOWER [9] and PSAT [10] has been used to solve the IEEE6 bus power system by Newton's based linear programming method and study Global optimization tool box in MATLAB User guide[11]

1.3 Problem Identification

The load flow problem[12] is of critical importance in the day to day operation of modern power systems.by far the most popular method of solution for the LF is that NRLF.The NRLF method has the advantages of being fast converging in usually 6 iterations.The system can be diverged when the power system is extremely loaded.When the power system is operating very close to its ceiling point, the Jacobian of the power flow equation set tends to be singular.In NRLF the performance is highly dependent on the initial values in the power system problem,it's difficult to determine abnormal operating solution.
Several methods are used for the computation processes but With increasing computer speeds, researchers are increasingly applying artificial and computational intelligence

techniques, especially in power system problems. These methods offer several advantages over traditional numerical methods. Among these techniques is that of genetic algorithm. Genetic algorithms (GA) are efficient stochastic search algorithms that emulate natural phenomena. They have been used successfully to solve wide range of optimization problems.Because of existence of local optima, these algorithms offer promise in solving large-scale problems.

1.4 Objective Of the Project

The objective of the project is to prove the GA approach in the load flow analysis provide more accurate results as compare to the conventional load flow method.Algorithm will build in MATLAB for implementation of load flow analysis.The algorithm will be implement for a particular(5 and IEEE14bus)system and compare the result with conventional load flow method with the help MAT Power result.

1.5 Outline Of Thesis

Chapter1:Contains the background of the LF,Literature review,problem identification and objective of the project.

chapter2:This chapter includes the Objective of the LF problem,basic load flow computational techniques, iterative solution method i.e.GS,NR,FDLF,Their comparative analysis.

chapter3:This chapter includes basic introduction to GA,different terminology used in the GA (such as Fitness function,chromosome etc.),continuous Genetic Algorithm approach,advantage of continuous GA over Binary encoding, idea relevant to Fitness function and Objective function and Multiple objective optimization technique

chapter4:This chapter introduce the implementation of the GA in the LF,methodology i.e step by step introduction of GA in Load Flow via coding approach has been introduce.Cross over operator,mutation operator, different selection scheme and convergence criteria.

chapter5:In this chapter 5 bus and IEEE 14 bus system has been tested and compare the result with conventional method that is NR method.

chapter6:This chapter include the conclusion based on the result and the future scope of the desertions work.

Chapter 2

Load Flow Analysis

Even though electric power networks are composed of components which are (or can be approximated to be) linear, electric power flow, real and reactive, is a nonlinear quantity. The calculation of load flow in a network is the solution to set of nonlinear equations.This is only an elementary treatment of this problem:there is still quite a bit of activity in the professional literature concerning load flow algorithms. The reason for this is that electric utility networks are often quite large, having thousands of buses, so that that amount of computational effort required for a solution is substantial.A lot of effort goes into doing the calculation efficiently.

Power flow in a network is determined by the voltage at each bus of the network and the impedances of the lines between buses.Power flow into and out of each of the buses that are network terminals is the sum of power flows of all of the lines connected to that bus. The load flow problem consists of nding the set of voltages magnitude and angle, which,together with the network impedances,produces the load flows that are known to be correct at the system terminals. To start, it view the power system as being a collection of buses, connected together by lines. At each of the buses,which it may regard as nodes, it may connect equipment which will supply power to or remove power from the system.

2.1 Objective Of LF study

1 Power flow analysis is very important in planning stages of new networks or addition to existing ones like adding new generator sites, meeting increase load demand and locating new transmission sites.

2 The load flow solution gives the nodal voltages and phase angles and hence the power injection at all the buses and power flows through interconnecting power channels.

3 It is helpful in determining the best location as well as optimal capacity of proposed generating station, substation and new lines.

4 It determines the voltage of the buses. The voltage level at the certain buses must be kept within the closed tolerances.

5 System transmission loss minimizes.

6 Economic system operation with respect to fuel cost to generate all the power needed

7 The line flows can be known. The line should not be overloaded, it means, we should not operate the close to their stability or thermal limits.

Five main properties are required of a load-flow solution method:

(A)High computational speed :This is especially important when dealing with large systems, real time applications (on-line), multiple case load flow such as in system security assessment, and also in interactive applications.

(B)Low computer storage : This is important for large systems and in the use of computers with small core storage availability, e.g. mini-computers for on-line application.

(C)Reliability of solution :It is necessary that a solution be obtained for ill-conditioned problems, in outage studies and for real time applications.

(D)Versatility :An ability on the part of load flow to handle conventional and special features (e.g. the adjustment of tap ratios on transformers,different representations of power system apparatus), and its suitability for incorporation into more.

(E)complicated processes

(F)Simplicity :The ease of coding a computer program of the load-flow algorithm.

2.2 Basic techniques for power-flow studies

A power-flow study (load-flow study) is an analysis of the voltages, currents, and power flows in a power system under steady-state conditions. In such a study, we make an assumption about either a voltage at a bus or the power being supplied to the bus for each bus in the power system and then determine the magnitude and phase angles of the bus voltages, line currents, etc. that would result from the assumed combination of voltages and power flows.The simplest way to perform power-flow calculations is by iteration

- There are four quantities of interest associated with each bus:
  - Real Power, P
  - Reactive Power, Q
  - Voltage Magnitude, V
  - Voltage Angle.
- At every bus of the system, two of these four quantities will be specified and the remaining two will be unknowns.
- Each of the system buses may be classified in accordance with which of the two quantities are specified

2.2.1 Types Of Buses

In load flow calculation the buses are classified into three type

a. **Slack bus**:

- voltage magnitude and angle are specified.
- The real and reactive power are unknowns.
- The bus selected as the slack bus must have a source of both real and reactive power, since the injected power at this bus must swingto take up the slack in the solution.
- The best choice for the slack bus (since, in most power systems,any buses have real and reactive power sources) requires experience with the particular system under study. The behavior of the solution is often influenced by the bus chosen.

b. **Voltage Controlled bus(P-V bus):**

- Any bus for which the voltage magnitude and the injected real power are specified is classified as a voltage controlled (or P-V) bus.
- The injected reactive power is a variable (with specified upper and lower bounds) in the power flow analysis.
- A P-V bus must have a variable source of reactive power such as a generator.

c. **Load bus(P-Q bus)**:

- A load bus is defined as any bus of the system for which the real and reactive power are specified.
- Load buses may contain generators with specified real and reactive power outputs,however, it is often convenient to designate any bus with specified injected complex power as a load bus.

2.2.2 General power Formulation

a. Read system data i.e. line data and bus data.

b. From the line data, equation of Y-bus has been built.

c. Make an initial estimate for the voltages at each bus in the system.

$$P_i^k = \sum \|V_i| * |V_j| * |Y_{ij}| * cos(\Theta_{ij} - \delta_i - \delta_j) \tag{2.1}$$

$$Q_i^k = \sum \|V_i| * |V_j| * |Y_{ij}| * sin(\Theta_{ij} - \delta_i - \delta_j) \tag{2.2}$$

Where for load buses,$P_i^{(k)}$ and$Q_i(k)$ are calculated from these equation and $\Delta P_i^{(k)}$ and$\Delta Q_i^{(k)}$ are calculate from these equation

$$\Delta P_i^{(k)} = P_{sch} - P_i^{(k)} \tag{2.3}$$

$$\Delta Q_i^{(k)} = Q_{sch} - Q_i^{(k)} \tag{2.4}$$

d. The new voltage magnitudes and phase angles are computed from following equation.

$$\delta_i^{(k+1)} = \delta_i^{(k)} - \Delta\delta_i^{(k+1)} \tag{2.5}$$

$$|V_i^{(k+1)}| = |V_i^{(k)}| - |\Delta V_i^{(k+1)}| \tag{2.6}$$

e. the process is continue until the residual$\Delta P_i^{(k)}$ and$\Delta Q_i^{(k)}$are less than specified accuracy

$$\Delta P_i^{(k)} \leq \varepsilon \tag{2.7}$$

$$\Delta Q_i^{(k)} \leq \varepsilon \tag{2.8}$$

2.3 Iterative Numerical (LF) solution methods

The solution of the simultaneous nonlinear power flow equations requires the use of iterative techniques for even the simplest power systems.There are many methods for

solving nonlinear equations, such as:

1 Gauss-Siedel Iterative Method(GS)

2 Newton-Raphson Method(N-R)

3 Fast Decoupled Load Flow Method(FDLF)

- **Gauss-Siedel Iterative Method(GS)**

  Its basic procedure is:

  a. Calculate the bus admittance matrix Y_{bus} including the admittances of all transmission lines, transformers, etc., between busses but exclude the admittances of the loads or generators themselves.

  b. Select a slack bus: one of the busses in the power system, whose voltage will arbitrarily be assumed as $1.0\angle 0$.

  c. Select initial estimates for all bus voltages: usually, the voltage at every load bus assumed as $1.0\angle 0$ (flat start) lead to good convergence.

  d. Write voltage equations for every other bus in the system. The generic form is

  $$V_i = \frac{1}{Y_{ii}}\left[\frac{P_i + jQ_{ii}}{V*} - \sum Y_{ik}V_k\right]$$

  where $k \neq i$,k=1 to N

  e. Calculate an updated estimate of the voltage at each load bus in succession using except for the slack bus.

  f. Compare the differences between the old and new voltage estimates: if the differences are less than some specified tolerance for all busses, stop. Otherwise, repeat step 5.

  g. Confirm that the resulting solution is reasonable: a valid solution typically has bus voltages.

- **Newton-Raphson Method**

  At each iteration of the Newton-Raphson method, the nonlinear problem is

approximated by a linear matrix equation (Jacobian matrix). The linearizing approximation can best be visualized in the case of a single-variable problem as shown in figure (3.2).

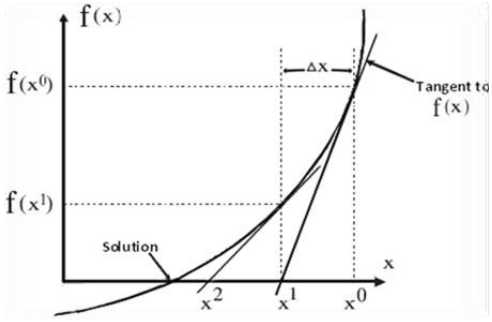


Figure 2.1: Single-Variable Linear Approximation [Taylor 1967]

The Jacobian matrix equation can be written as

$$\begin{pmatrix} J_1 & J_2 \\ J_3 & J_4 \end{pmatrix} \begin{pmatrix} \Delta\delta_1 \\ . \\ . \\ \Delta\delta_N \\ \Delta V_1 \\ . \\ . \\ . \\ \Delta V_N \end{pmatrix} = - \begin{pmatrix} \Delta P_1 \\ . \\ . \\ \Delta P_N \\ \Delta Q_1 \\ . \\ . \\ . \\ \Delta Q_N \end{pmatrix}$$

where$\Delta\delta_k = \delta_k^{v+1} - \delta_k^v$ and$\Delta V_k = V_k^{v+1} - V_k^v$ Newton-Raphson load flow method may have the final formulation of

$$\begin{pmatrix} \Delta P \\ \Delta Q \end{pmatrix} = \begin{pmatrix} J_1 & J_2 \\ J_3 & J_4 \end{pmatrix} \begin{pmatrix} \Delta\delta \\ \Delta V \end{pmatrix}$$

Using Newton-Raphson method to solve the mismatch powers equations either in polar coordinates with (ΔV) and ($\Delta\delta$) as variables or in rectangular coordinates with (Δe) and (Δf) as variables.

- **Fast Decoupled LF Method**

  Fast decoupled load flow method, possibly the most popular method used by utilities, is well known for its speed of solution, reduced memory, and reliable convergence (Nanda 1987). The algorithm is simpler, faster and more reliable than Newton?s method and has lower storage requirements. The fast decoupled load flow method is based on Newton?s load flow method with the modifications of neglecting the J2 and J3 Jacobian sub matrices due to the weak coupling between "P-V" and "Q-δ" quantities in power transmission system. Together with other approximations, the fast decoupled load flow equations become

$$\begin{pmatrix} \frac{\Delta P}{V} \end{pmatrix} = \begin{pmatrix} B' \end{pmatrix} \begin{pmatrix} \Delta\delta \end{pmatrix}$$
$$\begin{pmatrix} \frac{\Delta Q}{V} \end{pmatrix} = \begin{pmatrix} B'' \end{pmatrix} \begin{pmatrix} \Delta V \end{pmatrix}$$

  where$B'_{km} = -\frac{1}{x_{km}}$ for m$\neq k$,$B''_{km} = -B_{km}$ for m$\neq k$ and$B''_{km} = \sum B_{km}$ for m=k.

S.no.	Types of iterative method	Advantages	Disadvantages
1	Gauss-seidel(GS) method	Easy programmed Ease solution technique Low storage Low computational time	Slow converge in complex system No of iteration proportional to buses Sensitive to choice of reference bus
2	Newton-raphson(NR) method	Convergence is very fast Number of iterations independent of the number of buses Jacobian is very sparse matrix	Solution technique is difficult Sensitive to initial solution iteration time increasing as the buses. Sensitive to initial solution
3.	AC decoupled	Used in planning and contingency Useful for analyzing topology changes	No of iteration more than NR

Table I: Comparison of load flow methods

Chapter 3

Genetic Algorithm

3.1 Introduction To GA

Genetic Algorithms (GA) are adaptive methods which may be used to solve search and optimization problems. Over many generations, natural populations evolve according to the principles of natural selection and survival of the fittest.Genetic Algorithms work with a population of individuals, each representing a possible solution to a given problem.A GA allows a population composed of many individuals to evolve under specified selection rules to a state that maximizes the fitness (i.e., minimizes the cost function). [10]

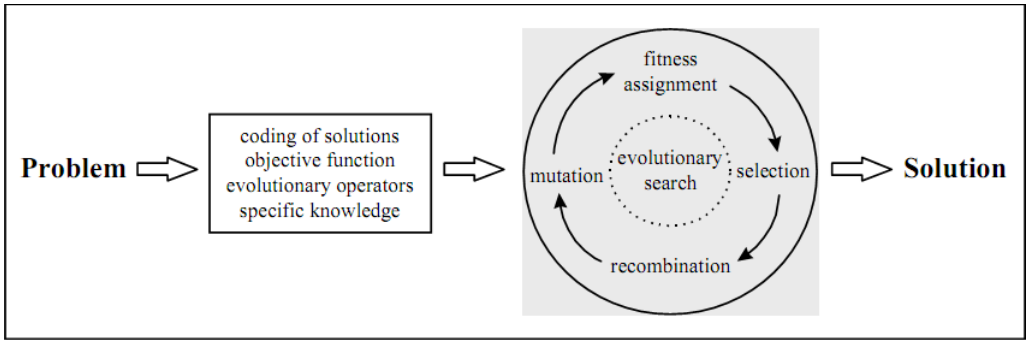


Figure 3.1: Problem solution using evolutionary Algorithms

Each individual is assigned a Fitness score according to how good a solution to the

problem it is. The highly-fit individuals are given opportunities to reproduce, by cross breeding with other individuals in the population. This produces new individuals as offspring, which share some features taken from each parent. The least fit members of the population are less likely to get selected for reproduction, and so die out. A whole new population of possible solutions is thus produced by selecting the best individuals from the current generation, and mating them to produce a new set of individuals.

This new generation contains a higher proportion of the characteristics possessed by the good members of the previous generation. In this way, over many generations,good characteristics are spread throughout the population. By favoring the mating If the genetic algorithm has been designed well, the population will converge to an optimal solution to the problem. There are some differences between genetic algorithms and traditional searching Algorithms(such as numerical techniques).of the more fit individuals, the most promising areas of the search space are explored.

3.1.1 Basic Terminology In GA [11][13]

1 **Chromosome**:An array of parameters or genes that is passed to the cost function.

2 **Converge**:Arrive at the solution.A gene is said to have converged when 95 of the chromosomes contain the same allele for that gene. GAs are considered converged when they stop finding better solutions.

3 **Cost function**:Function to be optimized.

4 **Crossover**: An operator that forms a new chromosome from two parent chromosomes by combining part of the information from each.

5 **Fitness**:Opposite of cost. A value associated with a chromosome that assigns a relative merit to that chromosome.

6 **Fitness function**:Has the negative output of the cost function. Mathematical subroutine that assigns a value or fitness to a set of variables.

7 **Individual**: A single member of a population that consists of a chromosome and its cost function.

8 **Mating pool**:A subset of the population selected for potential parents.

9 **Mutation**:A reproduction operator that randomly alters the values of genes in a parent chromosome.

10 **Mutation rate**:Percentage of bits in a population mutated each iteration of the GA.

11 **Objective function**:The function to be optimized.

12 **Offspring**:An individual generated by any process of reproduction.

13 **Population**:A group of individuals that interact (breed) together.

14 **Recombination**:Combining the information from two parent chromosomes via crossover.

15 **Reproduction**:The creation of offspring from two parent.

16 **Selection**:The process of choosing parents for reproduction (usually based on fitness).

3.1.2 Out line of the Basic GA

A **Start**:Generate random population of n chromosomes(suitable solution for the problems).

B **Fitness**:Evaluate the fitness f(x)of each chromosomes x in the population.

C **New population**:Create a new population by repeating following steps until the new population is complete.

Selection Select two parent chromosomes from a population according to their fitness(better the fitness,the bigger the chance to be selected)

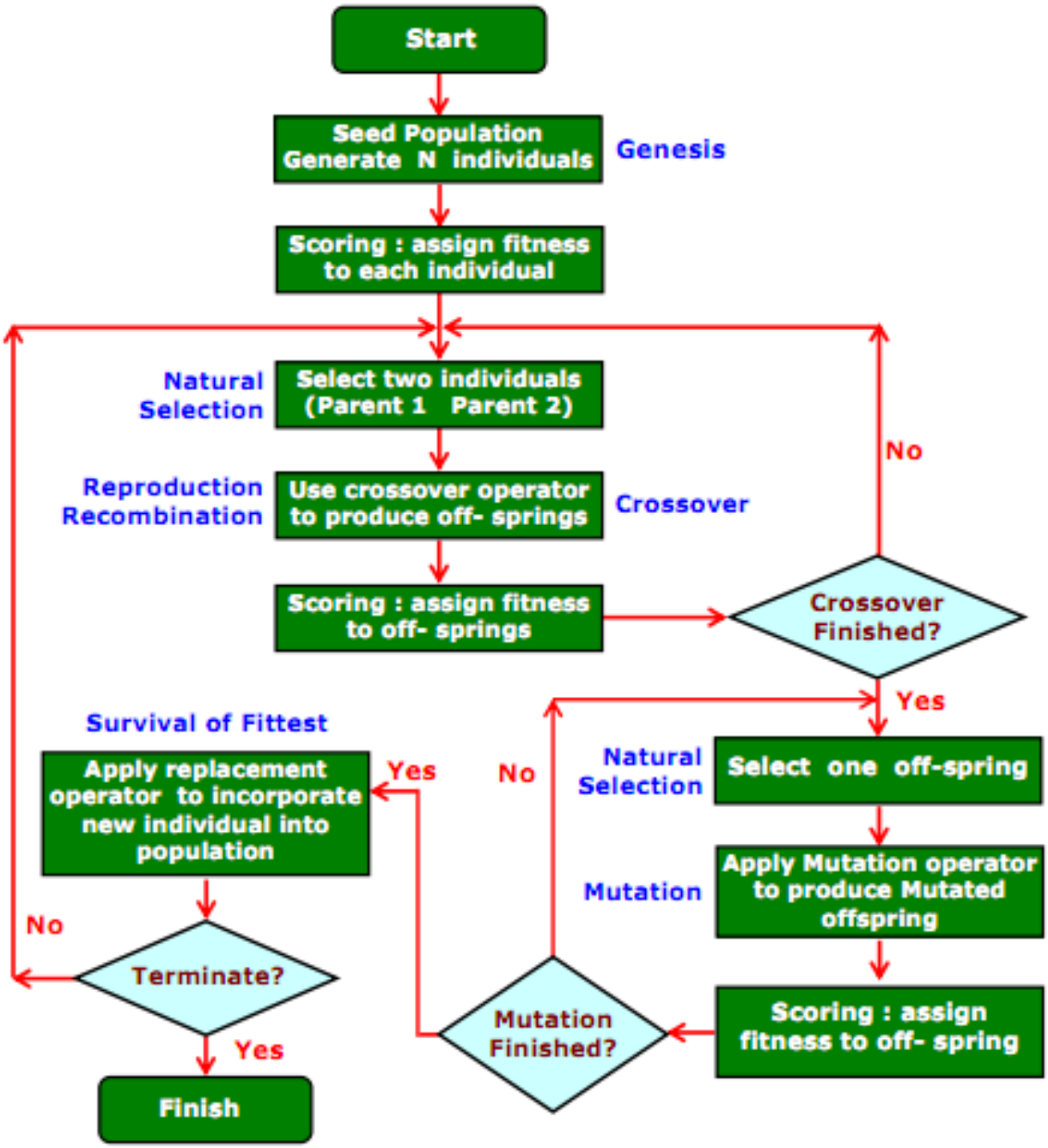


Figure 3.2: GA Program Flow Chart

Crossover With the crossover probability,cross over the parents to form new offspring(children).If no crossover was performed.offspring is the exact copy of parents.

Mutation :With a mutation probability,mutate new offspring at each locus(position in chromosomes).

Accepting :Place new offspring in the new population.

D **Replace**:Use new generated population for further run of the algorithm.

E **Test**:If the end condition is satisfied,stop,and return the best solution in current population.

F **Loop**;Go to step 2.

- **Types Of Encoding[12]**: Before a Genetic Algorithm can be put to work on any problem, a method is needed to encode potential solutions to that problem in a form so that computer can process.Following methods are used for encoding.

S.No	Type	Description and example
1	Binary	chromosome represent by 0 or 1 like 0101
2	value	here real value is implemented ex 1.23,ADFH,(back)
3	Permutation	used in ordering problem string represent position in a sequence
4	Tree	object,such as function or commands in programming language.

Table I: Encoding types

- **Types of GA Operator** These are used in GA to maintain the genetic diversity,is a necessity of process of evolution.Following are the genetic operator.

1 **Reproduction or Selection**:The genetic algorithm, through selection, determines which individuals will go to the reproduction phase.the most

Mutation operator	Chromosome Encoded	Description
Boundary	integer,float	Replace the value either from the upper bound lower bound
Non-uniform	integer,float	prevent from the population staging in early stage and find tune later state
Uniform	integer,float	replace the value of the chosen gene with a uniform random value user defined.
Gaussian	float,integer	the new value is clipped if fall out side the user defined.

Table II: Study of different Mutation operator

commonly used method for selecting chromosome for parents to cross over are:

- Roulette wheel selection method(Fitness -proportionate selection)
- Rank selection
- Tournament selection method.
- Steady state selection
- Boltzmann selection

2 **Cross Over or Recombination**:is a genetic operator that combines two chromosome to produce new chromosome.The idea behind is that new may be better than both of the parents chromosomes,the cross over operator are many types as follow;

- **One-point Crossover**
- **Two point,Uniform,Arithmetic and Heuristic crossover**

3 **Mutation**:It is genetic operator used to maintain genetic diversity from one generation of a population of a chromosome to next.

- **Termination criteria or convergence criteria**:for genetic process may be triggered by finding an acceptable approximate solution and bring the search to halt. The termination criteria can be one or more of the following criteria.

- Using diversity measure.
- After a specified number of generations.
- Finding an acceptable approximate solution.
- Repetition until no change in the solution.

3.2 Binary V/S Floating Coding Approach

The binary GA solves many optimization problems that stump traditional techniques, lets look a bit closer at the quantization limitation. What if we are attempting to solve a problem where the values of the variables are continuous and we want to know them to the full machine precision.

In such a problem each variable requires many bits to represent it. If the number of variables is large, the size of the chromosome is also large. When the variables are naturally quantized, the binary GA fits nicely. However, when the variables are continuous, it is more logical to represent them by floating-point numbers. In addition, since the binary GA has its precision limited by the binary representation of variables, using floating point numbers instead easily allows representation to the machine precision.This continuous GA also has the advantage of requiring less storage than the binary GA because a single floating-point number represents the variable instead of N bits integers.The continuous GA is inherently faster than the binary GA, because the chromosomes do not have to be decoded prior to the evaluation of the cost function.

3.2.1 Components Of a Continuous Genetic Algorithm

The Variables and Cost Function A cost function generates an output from a set of input variables (a chromosome). The cost function may be a mathematical function or an experiment. The objective is to modify the output in some desirable fashion by finding the appropriate values for the input variables. The goal is to solve some optimization problem where we search for an optimum

(minimum) solution in terms of the variables of the problem. If the chromosome has (Nvar) variables (an N-dimensional optimization problem) given by (b1, b2, , bNvar), then the chromosome is written as an array with $(1 \times Nvar)$ elements so that:

$$\text{chromosome} = [b_1, b_2, b_3, ..., b_{Nvar}]$$

In this case, the variable values are represented as floating-point numbers. Each chromosome has a cost found by evaluating the cost function (f) at the variables $(b_1, b_2, ..., b_{Nvar})$.

$$cost = f(chromosome) = f(b_1, b_2, ..., b_{Nvar})$$

The main aim of this primary problem in this research is the continuous functions introduced below.

The Objective and Fitness Functions?:The objective function is used to provide a measure of how individuals have performed in the problem domain. In the case of a minimization problem, the most fit individuals will have the lowest numerical value of the associated objective function. This raw measure of fitness is usually only used as an intermediate stage in determining the relative performance of individuals in a GA. Another function, the fitness function, is normally used to transform the objective function value into a measure of relative fitness.

Variable Encoding, Precision, And Bounds Here, the difference between binary and continuous Genetic Algorithms is shown. It is no longer needed to consider how many bits are necessary to represent accurately a value. Instead, (V) and (δ)have continuous values that are limited between appropriate bounds Although the values are continuous, a digital computer represents numbers by a finite number of bits. When we refer to the continuous genetic algorithm, it means that the computer uses its internal precision and roundoff to define the

precision of the value. Now, the algorithm is limited in precision to the roundoff error of the computer.

Initial Population The genetic algorithm starts with a group of chromosomes known as the population. We define an initial population of (N_{ind}) chromosomes. A matrix represents the population with each row in the matrix being a textbf($1 \times Nvar$)array (chromosome) of continuous values. Given an initial population of (N_{ind}) chromosomes, the full matrix of ($Nind \times Nvar$) random values is generated. All variables are normalized to have values between (0) and (1), the range of a uniform random number generator. The values of a variable are unnormalized in the cost function. If the range of values is between (b_{lo}) and (b_{hi}), then the unnormalized values are given by:

$$\mathbf{b}=(b_{hi} - b_{lo})b_{norm} + b_{lo}$$

b_{hi} : highest number in the variable range.

b_{lo}:lowest number in the variable range.

b_{norm} : normalized value of variable.

This society of chromosomes is not a democracy: the individual chromosomes are not all created equal.Each ones worth is assessed by the cost function. So at this point, the chromosomes are passed to the cost function for evaluation.

Natural Selection Survival of the fittest translates into discarding the chromosomes with the highest cost. First, the (N_{ind}) costs and associated chromosomes are ranked from lowest cost to highest cost. Then, only the best are selected to continue, while the rest are deleted. The selection rate,(X_{rate}), is the fraction of (N_{ind}) that survives for the next step of mating. The number of chromosomes that are kept each generation is:

$$N_{keep} = X_{rate}.N_{ind}$$

Natural selection occurs each generation or iteration of the algorithm. Of the (N_{ind})chromosomes, only the top (N_{keep}) survive for mating, and the bottom($N_{ind} - N_{keep}$) are discarded to make room for the new offspring. Deciding how many chromosomes to keep is somewhat arbitrary. Letting only a few chromosomes survive to the next generation limits the available genes in the offspring. Keeping too many chromosomes allows bad performers a chance to contribute their traits to the next generation.

Selection For Mating In this process, two chromosomes are selected from the mating pool of (Nkeep) chromosomes to produce two new offspring. Pairing takes place in the mating population until (Nind Nkeep) offspring are born to replace the discarded chromosomes. Pairing chromosomes in a genetic algorithm can be as interesting and varied as pairing in an animal species. Well look at a variety of selection methods.

a.Weighted random pairing (roulette-wheel)

b.Tournament selection.

Each of the parent selection schemes results in a different set of parents. As such, the composition of the next generation is different for each selection scheme. Roulette-wheel and tournament selection are standard for most genetic algorithms. It is very difficult to give advice on which selection scheme works best. In our problem, we follow the rank-weighting roulette-wheel and tournament parent selection procedures.

Recombination As for the binary algorithm, two parents are chosen, and the offspring are some combination of these parents. Many different approaches have been tried for crossing over in continuous genetic algorithm. The simplest methods choose one or more points in the chromosome to mark as the crossover points. Then the variables between these points are merely swapped between the two parents. The problem with real-valued crossover methods is that no new information is introduced: each continuous value that was randomly initiated in the initial population is propagated to the next generation, only in

different combinations. Although this strategy works fine for binary representations, there is now a continuum of values, and in this continuum we are merely interchanging two data points. These approaches totally rely on mutation to introduce new genetic

Mutation Random mutations alter a certain percentage of the genes in the list of chromosomes. We can sometimes find our method working too well. If care is not taken, the genetic algorithm can converge too quickly into one region of the cost surface. If this area is in the region of the global minimum, that is good. However, some functions, such as the one we are modeling, have many local minima. If nothing is done to solve this tendency to converge quickly, it may end up in a local rather than a global minimum. To avoid this problem of overly fast convergence (premature convergence), the routine is forced to explore other areas of the cost surface by randomly introducing changes, or mutations, in some of the variables. The basic method of mutation is not much more complicated for the continuous genetic algorithm. Do we also allow mutations on the best solutions? Generally not. They are designated as elite solutions destined to propagate unchanged. Such elitism is very common in genetic algorithms. Why throw away a perfectly good answer? The equation of mutation used here is.

$$b_{mut} = (b_{hi} - b_{lo})b_{norm} + b_{lo}$$

b_{mut}: variable under mutation.

Multiple Objective Optimization(MOO) In many applications, the cost function has multiple, often times, conflicting objectives. The most important approach to (MOO) are: weighted cost functions.

$$\textbf{cost}=\sum W_i f_i \text{ i=1,2.....N}$$

Where:

f_i=is the cost function (i).

W_i =is the weighting factor and $\sum W_i = 1$.

The problem with this method is determining appropriate values of (W_i). Different weights produce different costs for the same (f_i). This approach requires assumptions on the relative worth of the cost functions prior to running the genetic algorithm. Implementing this multiple objective optimization approach in a genetic algorithm only requires modifying the cost function to fit the form of and does not require any modification to the genetic algorithm.Thus,

$$cost = W f_1 + (1 - W) f_2$$

This approach is adopted in this research for its simplicity, easy of programming and gives us the required accuracy.

Chapter 4

GA In Load Flow Analysis

To overcome these limitations and difficulties of the conventional NR method, the proposed floating number GA representation is introduced.Genetic algorithms (GA) application to power systems is found in areas such as economic dispatch, power system planning, reactive power allocation, and power flow problem.

- The GA optimization process is almost insensitive to initial settings of the solution variable.
- It has the ability to determine the multiple power flow solutions.
- When the power system is highly stressed, the GA algorithm is capable of finding the power flow solution because this process does not require the formation of Jacobian matrix.

4.1 Design of GA components

To apply GAS to solve the load-flow problem, the method of representation of the load-flow solutions in the chromosomes, the fitness function for evaluating the fitness of the chromosomes, and the methods of crossover and mutation are required. These are presented below.

4.1.1 Steps

a. The first step is to make a admittance matrix.following data should be enter as follow.

nbus=number of buses.

nlns=no of lines.

ld=[FromBus ToBus complex impedance susptance]

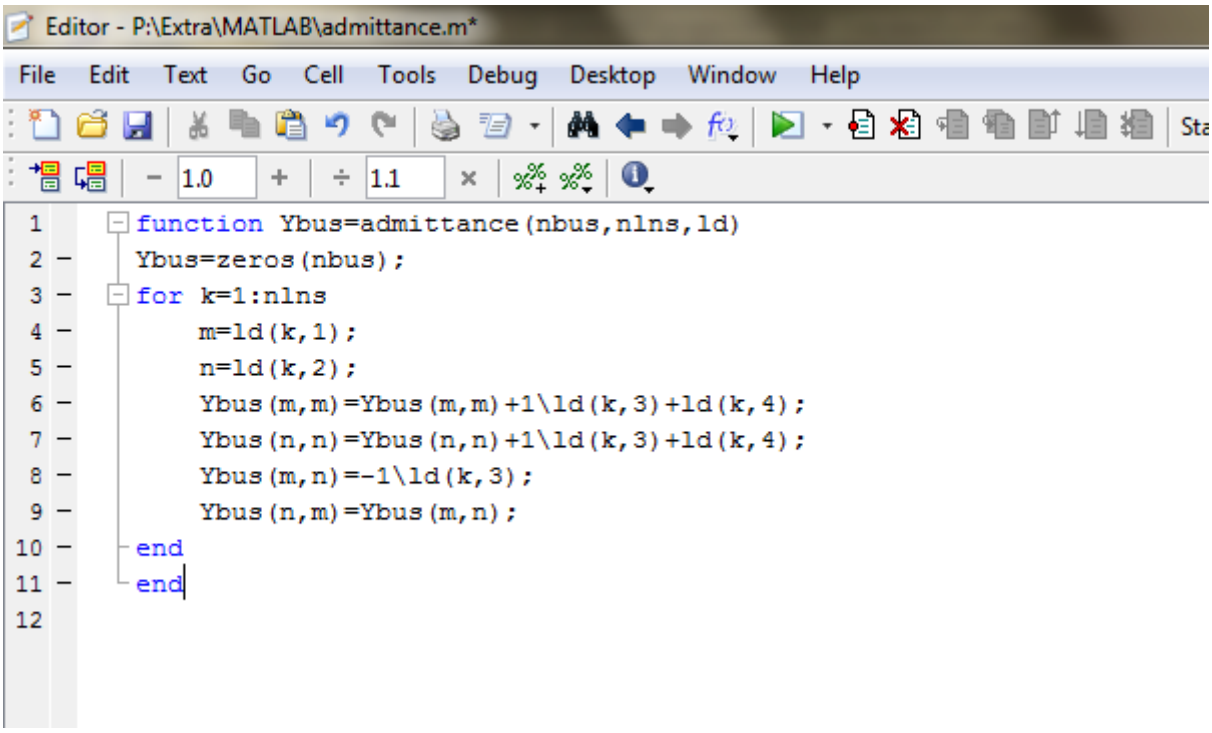


Figure 4.1: Generalized code for formulation of Ybus

A simple IEEE formulation of Ybus(Admittance)approach shown in this snap shot all the diagonal elements shown the self admittance and non-diagonal elements shown the transfer admittance.

b. Real numbers will be used to represent genes of the chromosome. Each chromosome consists of $2*nb-2-ng$ genes. The first np-1 gene represents the voltage angle of buses from bus 2 to bus np. The remaining genes represent voltage magnitude from bus 2 to bus np excluding ng voltage controlled buses. So a single chromosome will be as shown in fig. It is assumed that voltage controlled buses will be located at the last ng numbers from the nb numbers of the system buses.

Voltage angles genes						Voltages magnitudes genes					
δ_1	δ_2		δ_l		δ_{nb}	E_2	E_3		E_i		E_{nb-ng}

Figure 4.2: Chromosome representation

To generate the chromosome with in the precision bound the following concept are used.

$$\mathbf{b}=(b_{hi}-b_{lo})b_{norm}+b_{lo}$$

b_{hi} : highest number in the variable range.

b_{lo}:lowest number in the variable range.

b_{norm} : normalized value of variable.

At first defining the maximum range and minimum range of the Voltage and delta

```
% Create the initial population
 iga=0; % generation counter initialized
 V=(Emax-Emin)*rand(popsize,4)+Emin;
 DEL=-((Dmax-Dmin)*rand(popsize,4)+Dmin);% random
 par=[V DEL];
```

Figure 4.3: Chromosome generation

c. Choose the values of population size,selection rate, mutation probability, and maximum generation numbers.

```
% II Stopping criteria
maxit=1000; % max number of iterations
minmismatch=1.1; % minimum mismatch
%_____________________________________________________
% III GA parameters
popsize=100; % set population size
mutrate=0.2; % set mutation rate
selection=0.5; % fraction of population kept
```

Figure 4.4: Declaring the parameter

d. The main task is to assign the fitness to each chromosome,in load flow analysis at PQ buses the real and reactive power mismatch is minimised while in the case of Voltage controlled buses the real power and voltage magnitude is to be minimised, its indicate the final task should be minimizing P,Q at PQ buses and P,$|V|$ at PV buses.This is the known as the objective function as follow. Objective function can be defined as the sum of square of the power mismatches and the voltage mismatches as in.

$$H = \sum_{i=2}^{nb} \Delta P_i^2 + \sum_{i=2}^{nb-ng} \Delta Q_i^2 + \sum_{i=2}^{ng} \Delta V_i^2$$

Figure 4.5: Objective function

Where,

nb= represents the total number of system buses

ng=represents the number of voltage controlled buses

The objective function results from the summation of squares of active and reactive power mismatch. It is obvious that the optimal value for this function is 0. The unknown variables are E_p for all buses except the slack bus and voltage controlled buses in which E_p is known.δ_p for all buses except the slack bus.

e. Fitness value as described in this section.

```
for i=1:100
P(i)=p2(i,1)^2+p3(i,1)^2+p4(i,1)^2+p5(i,1)^2;%real power mismatch
Q(i)=q2(i,1)^2+q3(i,1)^2+q4(i,1)^2;%reactive power mismatch
V(i)=v(i,1)^2;%mismatch in magnitude of voltaage
H(i)=P(i)+Q(i)+V(i);%objective function

S(i)=100\(10+H(i));%fittness function
end
```

Figure 4.6: Objective function

f. Evaluate the each chromosome based on this fitness value.

```
mismatch=feval(ff,par); % calculates population mismatch using ff
[mismatch,ind]=sort(mismatch); % min mismatch in element 1
par=par(ind,:); % sort continuous
minc(1)=min(mismatch); % minc contains min of
meanc(1)=mean(mismatch); % meanc contains mean of population
```

Figure 4.7: Fitness function calling

After calculating the fitness of the each chromosome sorting the chromosome such that the chromosome having worst cost can be discarded from the population.After it the basic genetic operator are used to maintain the diversity in the population.

g. **Crossover:**Two parents are chosen,and the offspring are some combination of these parents.The simplest methods choose one or more points in the chromosome to mark as the crossover points. Then the variables between these points are merely swapped between the two parents. For example purposes, consider the two parents to be.

$$offspring_1 = [\, p_{m1}, p_{m2}, \uparrow p_{d3}, p_{d4}, \uparrow p_{m5}, p_{m6}, \ldots, p_{mN_{var}} \,]$$
$$offspring_2 = [\, p_{d1}, p_{d2}, \uparrow p_{m3}, p_{m4}, \uparrow p_{d5}, p_{d6}, \ldots, p_{dN_{var}} \,]$$

Figure 4.8: parent

Crossover points are randomly selected, and then the variables in between are exchanged:

$$offspring_1 = [\, p_{m1}, p_{m2}, \uparrow p_{d3}, p_{d4}, \uparrow p_{m5}, p_{m6}, \ldots, p_{mN_{var}} \,]$$
$$offspring_2 = [\, p_{d1}, p_{d2}, \uparrow p_{m3}, p_{m4}, \uparrow p_{d5}, p_{d6}, \ldots, p_{dN_{var}} \,]$$

Figure 4.9: offspring

Selection of mating

```
pick1=rand(1,M); % pate #1
pick2=rand(1,M); % pate #2
ic=1;
 while ic<=M
 for id=2:keep+1
 if pick1(ic)<=odds(id) && pick1(ic)>odds(id-1)
 pa(ic)=id-1;
 end
 if pick2(ic)<=odds(id) && pick2(ic)>odds(id-1)
 ma(ic)=id-1;
```

Figure 4.10: offspring

h. Single point cross over is implemented as follow;

```
%  Performs pating using single point crossover
  ix=1:2:keep; % index of mate #1
  xp=[ceil(rand(1,M)*Nt) ceil(rand(1,M)*Nt)]; % crossover point
  r=rand(1,M); % mixing parameter
for ic=1:M
  xy=par(pa(ic),xp(ic))-par(ma(ic),xp(ic)); % pa and ma mate
  par(keep+ix(ic),:)=par(pa(ic),:); % 1st offspring
  par(keep+ix(ic)+1,:)=par(ma(ic),:); % 2nd offspring
  par(keep+ix(ic),xp(ic))=par(pa(ic),xp(ic))-r(ic).*xy ;%1st
  par(keep+ix(ic)+1,xp(ic))=par(ma(ic),xp(ic))+r(ic).*xy; %2nd

  if xp(ic)<npar % crossover when last variable not selected
  par(keep+ix(ic),:)=[par(keep+ix(ic),1:xp(ic))  par(keep+ix(ic)+1,xp(ic)+1:npar)];
  par(keep+ix(ic)+1,:)=[par(keep+ix(ic)+1,1:xp(ic)) par(keep+ix(ic),xp(ic)+1:npar)];
  end%
```

Figure 4.11: Single point Cross over

i. It can sometimes find that method working too well. If care is not taken, the GA can converge too quickly into one region of the cost surface. If this area is in the region of the global minimum, that is good. However, some functions, such as the one we are modeling, have many local minima. If nothing to do solve this tendency to converge quickly, it would end up in a local rather than a global minimum. To avoid this problem of overly fast convergence,the technique force the routine to explore other areas of the cost surface by randomly introducing changes, or mutations, in some of the variables.The mutate value enter into the chromosome as follow

```
%Mutate the population
mrow=sort(ceil(rand(1,nmut)*(popsize-1))+1);
mcol=ceil(rand(1,nmut)*Nt/2);
 abc=par(:,[1 2 3 4]);
 bcd=par(:,[5 6 7 8]);
for ii=1:nmut
abc(mrow(ii),mcol(ii))=    (Emax-Emin)*rand+Emin;
bcd(mrow(ii),mcol(ii))=   ((Dmax-Dmin)*rand+Dmin);
end % ii
 par=[abc bcd];
%
```

Figure 4.12: Mutation the Voltage and Delta

j. To stop the genetic algorithm certain criteria should be needed in this case as the fitness reaches near to one or equal to one then GA stop.

```
% Stopping criteria
if iga>maxit || mismatch(1)==1.0
break
end
[iga mismatch(1)];
end%iga
```

Figure 4.13: Stopping criteria

Chapter 5

Coding Implementation And Results

The examples of Fig5.1 and Fig5.2 are shown in that the solution of the N-R method can be diverged when the power system is loaded extremely. The reason is that the Jacobian of the power flow equation tends to be singular.Table presents the result and determinant of normal case.

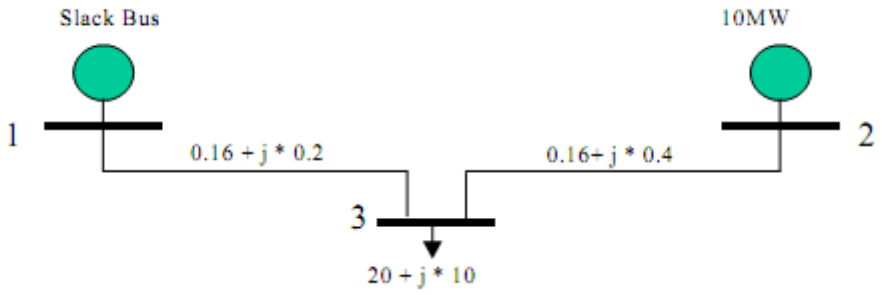


Figure 5.1: Simple System

However, the next example shows the change of determinant of the Jacobian matrix. When load is stressed, the Jacobian of the power flow equation tends to be singular.

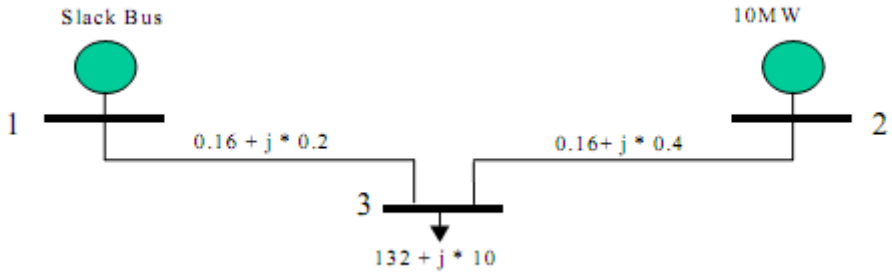


Figure 5.2: The change of determinant of Jacobian matrix

Bus	Voltage		Generation		Load	
	Mag[p.u]	ang(degree)	P[MW]	Q[MVAR]	P[MW]	Q[MVAR]
1	1.000	0.00	11.67	-14.47	-	-
2	1.1249	-2.871	10.00	27.94	-	-
3	1.0113	-2.634	-	-	20.00	10.00

Table I: The result and determinant of normal case

Bus	Voltage		Generation		Load	
	Mag[p.u]	ang(degree)	P[MW]	Q[MVAR]	P[MW]	Q[MVAR]
1	3	11.39	-14.81	0.553	0.69	-
2	3	9.44	26.55	1.114	2.78	-

Table II: The result and determinant of Heavy load

Determinant of Jacobian = 0.0084

When the load was over 133MW, the solution of N-R power flow did not converge.

A MATLAB program has been developed based on the proposed GA. The proposed method has been tested by applying it to the 5 bus system shown as below

5.1 Testing of method on 5 Bus system

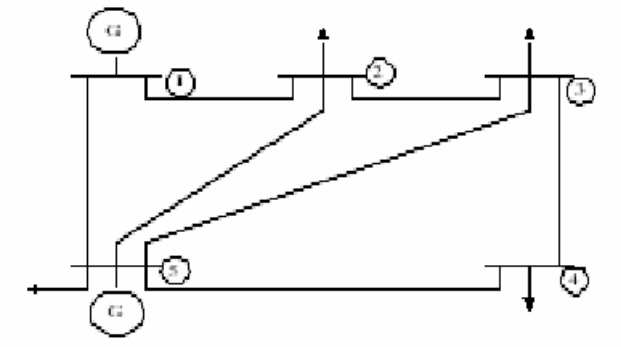


Figure 5.3: 5 bus system

Bus 1 is the slack bus with $V_1 = 1.06, \delta_1 = 0$

with a voltage controlled bus 5, $V_5 = 1$, and $P_5 = 0.4$

Buses $2, 3$, and 4are load buses as presented in following fig. The GA parameter

P_2	Q_2	P_3	Q_3	P_4	Q_4
0.45	0.15	0.4	0.05	0.6	0.1

Figure 5.4: Real and reactive power at load buses

setting for this system is as shown below Popsize=200

Selection rate=.5

Mutation rate=.3

$V_{max} = 1.1, V_{min} = 0.9,$

$\delta_{max} = 0.1743; \delta_{min} = -0.1743;$

Impedance By using the Admittance algorithm the Ybus is formed as follow

Bus p-q	Impedance(pu)
1-2	0.08+j0.24
1-5	0.02+j0.06
2-3	0.01+j0.03
2-5	0.06+j0.18
3-4	0.08+j0.24
3-5	0.06+j0.18
4-5	0.04+j0.12

Figure 5.5: Line Impedances

The real and Reactive power mismatch for load buses 2 3 and 4 are formed as below.

```
Ybus=[6.25-18.75i  -1.25+3.75i          0            0          -5.+15.00i;
     -1.25+3.75i    12.917-38.75i    -10+30i         0           -1.67+5i;
        0           -10+30i        12.9167-38.75i  -1.25+3.75i    -1.67+5i;
        0              0           -1.25+3.75i     3.75-11.25i    -2.5 + 7.5i;
     -5+15i        -1.67+5i         -1.67+5i       -2.5+7.5i      10.83-32.5];
```

Figure 5.6: Admittance Matrix

```
function S1=final_main1(x)

Ybus=[6.25-18.75i  -1.25+3.75i           0            0          -5.+15.00i;
     -1.25+3.75i    12.917-38.75i    -10+30i          0           -1.67+5i;
        0           -10+30i        12.9167-38.75i  -1.25+3.75i    -1.67+5i;
        0              0          -1.25+3.75i     3.75-11.25i    -2.5 + 7.5i;
     -5+15i       -1.67+5i        -1.67+5i       -2.5+7.5i      10.83-32.5];
Ymag=abs(Ybus);
Theta=angle(Ybus);
V2=x(:,1);D2=x(:,4);
V3=x(:,2);D3=x(:,5);
V4=x(:,3);D4=x(:,6);
V5=1;D5=x(:,7);
V1=1.06 ;D1=0;
for i=1:100
   g1(i)=0.45-V2(i,1)*(V1*Ymag(2,1)*cos(D2(i,1)-D1-Theta(2,1))+V2(i,1)*Ymag
   g2(i)=0.15-V2(i,1)*(V1*Ymag(2,1)*sin(D2(i,1)-D1-Theta(2,1))+V2(i,1)*Ymag
end

% g1=sort(g1);
% g2=sort(g2);
S1=[g1' g2'];
end
```

Figure 5.7: P and Q mismatch at bus2

```
function S2=final_main2(x)

Ybus=[6.25-18.75i  -1.25+3.75i           0            0          -5.+15.00i
     -1.25+3.75i    12.917-38.75i    -10+30i          0           -1.67+5i;
        0           -10+30i        12.9167-38.75i  -1.25+3.75i    -1.67+5i;
        0              0          -1.25+3.75i     3.75-11.25i    -2.5 + 7.5
     -5+15i       -1.67+5i        -1.67+5i       -2.5+7.5i      10.83-32.5]
Ymag=abs(Ybus);
Theta=angle(Ybus);
V2=x(:,1);D2=x(:,4);
V3=x(:,2);D3=x(:,5);
V4=x(:,3);D4=x(:,6);
V5=1;D5=x(:,7);
V1=1.06 ;D1=0;
for i=1:100
   g1(i)=0.5-V2(i,1)*(V1*Ymag(2,1)*cos(D2(i,1)-D1-Theta(2,1))+V2(i,1)*Ymag(
   g2(i)=0.5-V2(i,1)*(V1*Ymag(2,1)*sin(D2(i,1)-D1-Theta(2,1))+V2(i,1)*Ymag(
end
% g1=sort(g1);
% g2=sort(g2);
  S2=[g1' g2'];
end
```

Figure 5.8: P and Q mismatch at bus3

```
function S3=final_main3(x)

Ybus=[6.25-18.75i  -1.25+3.75i           0            0          -5.+15.00i;
     -1.25+3.75i    12.917-38.75i    -10+30i          0           -1.67+5i;
        0           -10+30i        12.9167-38.75i  -1.25+3.75i    -1.67+5i;
        0              0          -1.25+3.75i     3.75-11.25i    -2.5 + 7.5i;
     -5+15i       -1.67+5i        -1.67+5i       -2.5+7.5i      10.83-32.5];
Ymag=abs(Ybus);
Theta=angle(Ybus);
V2=x(:,1);D2=x(:,4);
V3=x(:,2);D3=x(:,5);
V4=x(:,3);D4=x(:,6);
V5=1;D5=x(:,7);
V1=1.06 ;D1=0;
for i=1:100
   g1(i)=0.6-V2(i,1)*(V1*Ymag(2,1)*cos(D2(i,1)-D1-Theta(2,1))+V2(i,1)*Ymag(2,
   g2(i)=0.1-V2(i,1)*(V1*Ymag(2,1)*sin(D2(i,1)-D1-Theta(2,1))+V2(i,1)*Ymag(2,
end

% g1=sort(g1);
% g2=sort(g2);
S3=[g1' g2'];
end
```

Figure 5.9: P and Q mismatch at bus4

```
function S4=final_main4(x)

Ybus=[6.25-18.75i  -1.25+3.75i           0            0          -5.+15.00i;
     -1.25+3.75i    12.917-38.75i    -10+30i          0           -1.67+5i;
        0           -10+30i        12.9167-38.75i  -1.25+3.75i    -1.67+5i;
        0              0          -1.25+3.75i     3.75-11.25i    -2.5 + 7.5i;
     -5+15i       -1.67+5i        -1.67+5i       -2.5+7.5i      10.83-32.5];
Ymag=abs(Ybus);
Theta=angle(Ybus);
V2=x(:,1);D2=x(:,4);
V3=x(:,2);D3=x(:,5);
V4=x(:,3);D4=x(:,6);
V5=1;D5=x(:,7);
V1=1.06 ;D1=0;
for i=1:100
   g1(i)=0.2-V2(i,1)*(V1*Ymag(2,1)*cos(D2(i,1)-D1-Theta(2,1))+V2(i,1)*Ymag(2,
%     g2(i)=0.1-V2(i,1)*(V1*Ymag(2,1)*sin(D2(i,1)-D1-Theta(2,1))+V2(i,1)*Ymag
    g2(i)=1.0;
end
% g1=sort(g1);
% g2=sort(g2);
  S4=[g1' g2'];
end
```

Figure 5.10: P mismatch at 5

```
function F=final(x)
S1=final_main1(x);
S2=final_main2(x);
S3=final_main3(x);
S4=final_main4(x);
p2=S1(:,1);
p3=S2(:,1);
p4=S3(:,1);
p5=S4(:,1);
q2=S1(:,2);
q3=S2(:,2);
q4=S3(:,2);
% v=S4(:,2);
for i=1:100
P(i)=p2(i,1)^2+p3(i,1)^2+p4(i,1)^2+p5(i,1)^2;
Q(i)=q2(i,1)^2+q3(i,1)^2+q4(i,1)^2;
% V(i)=v(i,1)^2;
H(i)=sum(P(i)+Q(i));
S(i)=100\(1+H(i));
end
 F=S;

end
```

objective function

fitness function

Figure 5.11: Objective And fitness function

population = Voltage mag Del

1.0197	0.9097	1.0572	0.1217	0.1713	-0.1388	0.1475
1.0933	0.9870	0.9616	-0.0678	0.0634	0.1348	-0.0905
0.9716	0.9731	1.0479	0.1473	0.1345	0.0245	-0.0988
0.9138	1.0467	0.9796	0.0305	-0.0744	0.0409	-0.1310
0.9649	0.9192	1.0528	-0.1366	-0.0668	-0.0882	0.0656
1.0742	1.0726	1.0682	-0.1417	-0.0366	0.1529	-0.0073
0.9274	1.0720	0.9074	-0.0041	0.1403	-0.0848	0.0466
0.9887	1.0840	0.9390	-0.0764	0.0523	0.1139	0.0231
0.9638	1.0515	1.0567	-0.1508	-0.0984	-0.0800	0.1441
0.9000	1.0595	1.0692	-0.0960	0.0662	0.0832	0.0216
1.0332	1.0832	0.9711	0.0589	-0.1302	0.1181	0.1271
1.0333	0.9097	1.0117	0.0978	0.0337	-0.1392	-0.1719
1.0799	0.9013	0.9312	-0.1298	0.0839	-0.0315	0.1478
1.0984	1.0644	0.9201	0.0161	0.1035	-0.0790	-0.0459
0.9001	0.9350	1.0215	-0.1740	0.0722	-0.0520	-0.1617
0.9226	1.0897	1.0411	0.1215	0.0777	0.0210	0.0820
1.0924	1.0067	1.0169	-0.1046	-0.1484	0.1122	0.1477

Figure 5.12: Chromosome

The result obtained from the GALF is compare with the conventional NRLF method and the percentage error being calculated as tabulated below

Variable	GA solution	NR solution	Per Error
δ_2	-0.078621	-0.079164	0.69
δ_3	-.081025	-.084709	4.3
δ_4	-.096325	-.09935	3.04
δ_5	-.034895	-.036085	3.29
V_2	.97328	0.9805	.73
V_3	0.9422	0.9771	3.57
V_4	0.93786	0.96617	2.9

Table III: Comparison between GA Solution and NR solution

5.2 IEEE 14 Bus System

Line impedance data taken from the MATPOWER(open source) using it admittance matrix is formed.Bus 2,3,6 and 8 are Voltage Controlled buses,bus no 1 is slack bus and other all buses are the load buses4,5,7,9,10,11,12,13 and 14.The data presented in the appendix

GA parameter setting Popsize=200

Selection rate=.5

Mutation rate=.3

$V_{max} = 1.3, V_{min} = 1.3,$

$Del_{max} = 0; Del_{min} = -0.34889;$

objective function defined as

```
function F=finally(x)
k2=bus2(x);
k3=bus3(x);
k4=bus4(x);
k5=bus5(x);
k6=bus6(x);
k7=bus7(x);
k8=bus8(x);
k9=bus9(x);
k10=bus10(x);
k11=bus11(x);
k12=bus12(x);
k13=bus13(x);
k14=bus14(x);

%_____________________________Real power calculation
P2=k2(:,1);P3=k3(:,1);P4=k4(:,1);P5=k5(:,1);P6=k6(:,1);P7=k7(:,1);P8=k8(
%_____________________________Reactive power calculation
Q4=k4(:,2);Q5=k5(:,2);Q7=k7(:,2);Q9=k9(:,2);Q10=k10(:,2);Q11=k11(:,2);Q1
%_____________________________ VOLTAGE CALCULATION
V2=k2(:,2);V3=k3(:,2); V6=k6(:,2); V8=k8(:,2);

for i=1:100
    P(i)=P2(i,1)^2+P3(i,1)^2+P4(i,1)^2+P5(i,1)^2+P6(i,1)^2+P7(i,1)^2+P8(
    Q(i)=Q4(i,1)^2+Q5(i,1)^2+Q7(i,1)^2+Q9(i,1)^2+Q10(i,1)^2+Q11(i,1)^2+Q
    V(i)=V2(i,1)^2+V3(i,1)^2+V6(i,1)^2+V8(i,1)^2;
    H(i)=sum(P(i)+Q(i));
end
F=1000000000\(1+H);
end
```

Figure 5.13: Objective Function

Variable voltage mag	GA solution	NR solution	Per Error
2	1.0450	0.9987	4.4306
3	1.0100	1.0942	8.366
4	1.0180	1.0366	1.82
5	1.0200	1.0090	1.07
6	1.0700	1.0731	0.289
7	1.0629	1.0960	3.11
8	1.0900	1.0008	8.1
9	1.0566	1.0055	4.83
10	1.0516	1.0760	2.30
11	1.0570	1.0427	1.35
12	1.0550	1.0572	0.29
13	1.0500	1.0224	2.89
14	1.0360	1.0509	1.43

variable	GA Solution
δ_2	-6.4605
δ_3	-13.3656
δ_4	-12.4987
δ_5	-5.6223
δ_6	-13.5287
δ_7	-10.8860
δ_8	-13.2191
δ_9	-12.3019
δ_{10}	-15.9210
δ_{11}	-13.3478
δ_{12}	-17.2611
δ_{13}	-17.3764
δ_{14}	-15.3764

Objective function H=0.00070909

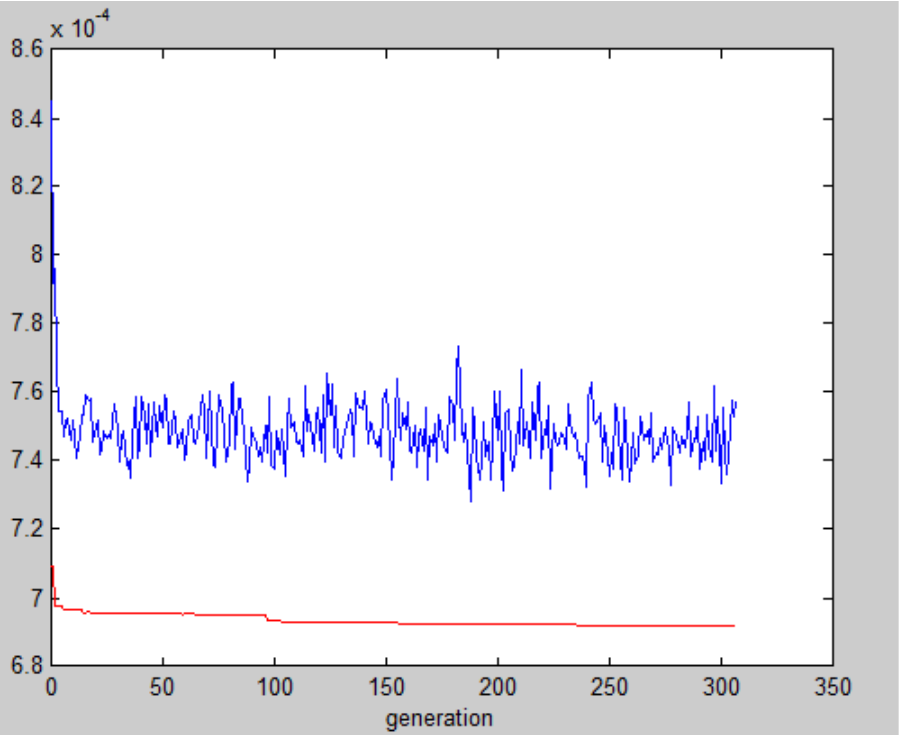


Figure 5.14: Objective Function V/s Generation

Chapter 6

Conclusion and Future work

6.1 Conclusion

- By the implementation of the GA for 5 bus and IEEE 14 bus has been concluded The genetic operators are one point crossover, non-uniform mutation, and Roult-wheel selection. To test the algorithm accuracy it has been applied to 14bus to find its power flow solution. In the 5-bus system of case 1, obtained results have been compared to those of the conventional NR method. It has been shown that the proposed GA method has reached an accurate solution with in the error limit 10 percent. It is believed that this method will be superior over other conventional methods in the case of very large systems.

- Although binary-coded genetic algorithm has been successfully applied to a wide range of optimization problems, they suffer from disadvantage when applied to the real-world problems involving a large number of variables. Thus, in my problem the real-coded genetic algorithm, where all decision variables (unknowns) are expressed as real numbers. Explicit conversion to binary does not take place. A reduction of computational effort is an obvious advantage of a real-coded genetic algorithm. Another advantage is that, an absolute precision is now attainable by making it possible to overcome the crucial decision of how

many bits are needed to represent potential solutions.

- Blending models are used in the crossover operator to remedy the problem of the crossover in the real-coded genetic algorithm which is, no new information is introduced each continuous value that was randomly initiated in the initial population is propagated to the next generation, only in different combinations.

6.2 Future work

- It need to develop methods to accelerate the convergence of the GA method further so that a computation speed close to that of the NR method can be achieved and to develop methods for GA to deal with the violation of generator reactive power limits.

- Enhancing the mutation scheme such that reach the solution nearer to the desired accuracy.

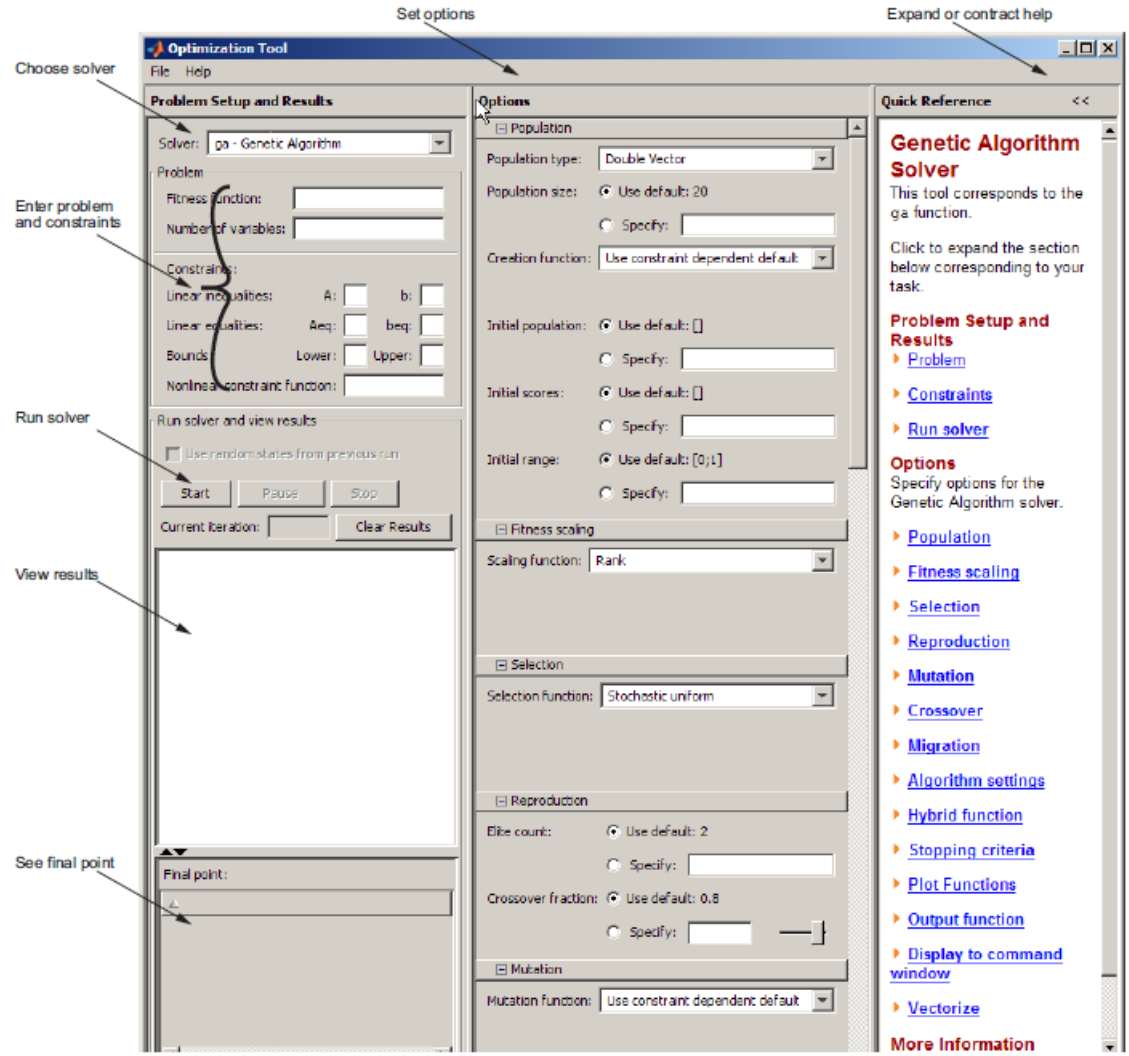
- Applying the Multi-Objective Optimization technique to reduces the concept of fitness function in the load Flow

Appendix A

Appendix A

Introduction to MATLAB7.9.0 R2009a GA Tool Box

Use the Optimization Tool To open the Optimization Tool, enter optimtool('ga')at the command line on MATLAB, or enter optimtool and then choose ga from the Solver menu.To use the Optimization Tool,first enter the following information:Fitness Function:The objective function that want to minimize. Enter the fitness function in the form@fitnessfun, where fitnessfun.m is a file that computes the fitness function .The @ sign creates a function handle to fitness function. Number of variables:The length of the input vector to the fitness function. we can enter constraints or a nonlinear constraint function for the problem in the Constraints pane. If the problem is unconstrained, leave these fields blank. To run the genetic algorithm, click the Start button. The tool displays the results of the optimization in the Run solver and view results pane.we can change the options for the genetic algorithm in the Options pane.To view the options in one of the categories


Set options
Expand or contract help
Choose solver
Enter problem and constraints
Run solver
View results
See final point
Optimization Tool
File Help
Problem Setup and Results
Solver: ga - Genetic Algorithm
Problem
Fitness function:
Number of variables:
Constraints:
Linear inequalities: A: b:
Linear equalities: Aeq: beq:
Bounds Lower: Upper:
Nonlinear constraint function:
Run solver and view results
Use random states from previous run
Start Pause Stop
Current iteration:
Clear Results
Final point:
Options
Population
Population type: Double Vector
Population size: Use default: 20
Specify:
Creation function: Use constraint dependent default
Initial population: Use default: []
Specify:
Initial scores: Use default: []
Specify:
Initial range: Use default: [0;1]
Specify:
Fitness scaling
Scaling function: Rank
Selection
Selection function: Stochastic uniform
Reproduction
Elite count: Use default: 2
Specify:
Crossover fraction: Use default: 0.8
Specify:
Mutation
Mutation function: Use constraint dependent default
Quick Reference
Genetic Algorithm Solver
This tool corresponds to the ga function.
Click to expand the section below corresponding to your task.
Problem Setup and Results
Problem
Constraints
Run solver
Options
Specify options for the Genetic Algorithm solver.
Population
Fitness scaling
Selection
Reproduction
Mutation
Crossover
Migration
Algorithm settings
Hybrid function
Stopping criteria
Plot Functions
Output function
Display to command window
Vectorize
More Information

AppendixA

A simple GA code in MATLAB

```
LIND = 15;                                  % Length of individual vars.
NVAR = 2;                                   % No. of decision variables
NIND = 40;                                  % No. of individuals
GGAP = 0.9;                                 % Generation gap
XOV = 0.7;                                  % Crossover rate
MUTR = 0.0175;                              % Mutation rate
MAXGEN = 30;                                % No. of generations
                                            % Binary representation scheme
FieldD = [LIND LIND; 1 1; 1000 1000; 1 1; 0 0; 0 0; 0 0];
% Initialise population
Chrom = crtbp(Nind, Lind*NVAR);             % Create binary population
ObjV = objfun(bs2rv(Chrom, FieldD));        % Evaluate objective fn.
Gen = 0;                                    % Counter
% Begin generational loop
while Gen < MAXGEN
    % Assign fitness values to entire population
    FitnV = ranking(ObjV);
    % Visualisation
    plotgraphics
    % Select individuals for breeding
    SelCh = select('sus', Chrom, FitnV, GGAP);
    % Recombine individuals (crossover)
    SelCh = recombin('xovsp', SelCh, XOV);
    % Apply mutation
    SelCh = mut(SelCh, MUTR);
    % Evaluate offspring, call objective function
    ObjVSel = objfun(bs2rv(SelCh, FieldD));
    % Reinsert offspring into population
    [Chrom ObjV]=reins(Chrom, SelCh, 1, 1, ObjV, ObjVSel);
    % Increment counter
    Gen = Gen+1;
end
% Convert Chrom to real-values
Phen = bs2rv(Chrom, FieldD);
```

Appendix B

GA programming

$ff =' finally';$

$npar = 26;$

$half = npar/2;$

$Emax = 1.1; Emin = .99;$

$Dmax = 0; Dmin = -0.34889;$

$maxit = 1000;$

$minmismatch = .999;$

$popsize = 100;$

$mutrate = 0.2;$

$selection = 0.5;$

$Nt = npar;$

$keep = floor(selection * popsize);$

$nmut = ceil((popsize - 1) * Nt * mutrate);$

$M = ceil((popsize - keep)/2);$

$iga = 0;$

$V = (Emax - Emin) * rand(popsize, 13) + Emin;$

$DEL = ((Dmax - Dmin) * rand(popsize, 13) + Dmin);$

$par = [VDEL];$

$mismatch = feval(ff, par);$

$par = par(ind, :);$

$\underline{\mathbf{A}}$*ppendixB*

$minc(1) = min(mismatch);$

$meanc(1) = mean(mismatch);$

$whileiga < maxit$

$iga = iga + 1;$

$M = ceil((popsize - keep)/2);$

$numberofmatingsprob = flipud([1 : keep]'/sum([1 : keep]));$

$weightschromosomesodds = [0cumsum(prob(1 : keep))'];$

$pick1 = rand(1, M);$

$pick2 = rand(1, M);$

$ic = 1;$

$whileic <= M$

$forid = 2 : keep + 1$

$ifpick1(ic) <= odds(id)andpick1(ic) > odds(id - 1)$

$pa(ic) = id - 1;$

end

$ifpick2(ic) <= odds(id)andpick2(ic) > odds(id - 1)ma(ic) = id - 1;$

end

end

$ic = ic + 1;$

end

$ix = 1 : 2 : keep;$

$xp = [ceil(rand(1, M) * Nt)ceil(rand(1, M) * Nt)];$

$r = rand(1, M);$

$foric = 1 : M$

$xy = par(pa(ic), xp(ic)) - par(ma(ic), xp(ic));$

$par(keep + ix(ic), :) = par(pa(ic), :);$

$par(keep + ix(ic) + 1, :) = par(ma(ic), :);$

$par(keep + ix(ic), xp(ic)) = par(pa(ic), xp(ic)) - r(ic). * xy;$

$par(keep + ix(ic) + 1, xp(ic)) = par(ma(ic), xp(ic)) + r(ic). * xy;$

$ifxp(ic) < npar$

AppendixB

$par(keep+ix(ic),:) = [par(keep+ix(ic),1:xp(ic))par(keep+ix(ic)+1,xp(ic)+1:$
$npar)];$
$par(keep+ix(ic)+1,:) = [par(keep+ix(ic)+1,1:xp(ic))par(keep+ix(ic),xp(ic)+1:$
$npar)];$
end
end
$mrow = sort(ceil(rand(1,nmut) * (popsize-1)) + 1);$
$mcol = ceil(rand(1,nmut) * Nt/2);$
$abc = par(:,[12345678910111213]);$
$bcd = par(:,[151617181920212223242526]);$
$forii = 1:nmutabc(mrow(ii),mcol(ii)) = (Emax-Emin) * rand + Emin;$
$bcd(mrow(ii),mcol(ii)) = ((Dmax-Dmin) * rand + Dmin);$
$endpar = [abcbcd];$
$mismatch = feval(ff,par);$
$(mismatch,ind) = sort(mismatch);$
$par = par(ind,:);$
$minc(iga+1) = min(mismatch);$
$meanc(iga+1) = mean(mismatch);$
$ifiga > maxit||mismatch(1) < 1break$
end
$(igamismatch(1));$
end
$disp(['optimizedfunctionis'ff])$
$disp(['popsize =' num2str(popsize)'mutrate =' num2str(mutrate)'par =' num2str(npar)])$
$disp(['generations =' num2str(iga)'bestmismatch =' num2str(mismatch(1))])$
$disp(['bestsolution']);$
$disp([num2str(par(1,:))]);$
$iters = 0:length(minc)-1;$
$plot(iters,meanc,iters,minc,'r');$

References

[1] STOTT BRIAN."Review of Load-Flow Calculation Methods". IEEE transection page VOL62, JULY 1974.

[2] A Klos and J. Wojcicka."Physical aspects of the nonuniqueness of load flow solutions " Electrical Power and Energy Systems,Vol 13,No 5,October 1991.

[3] K.F.Man,K.S.Tang,and S.Kwong,"Genetic Algorithms: Concepts and Applications" IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS, VOL.43, NO.5,OCTOBER1996

[4] Xiaodong YIN"Application of Genetic Algorithms Problem in Electrical Power Systems to Multiple Load Flow Solution"Proceding of tho 32nd Confrence on Decision and control ,2001

[5] Hassan A. Kubba and Samir Sami Mahmood,“ Genetic Algorithm Based Load Flow Solution Problem In Electrical Power Systems",Journal of Engineering ,Vol 4,15 December,2009.

[6] Muhammad Tami Al-Hajri,M. A. Abido."Assessment of Genetic Algorithm Selection, Crossover and Mutation Techniques in Reactive Power Optimization", IEEE Congress on Evolutionary Computation,2009.

[7] Mustafa Bagriya,Zehra Alif Aygen,“Minimizing Power Transmission Losses using Genetic Algorithm", International Conference of Electrical And Electronics Engineering,Vol67,1998.

[8] F.D.Galiana, ZC. Zeng,“Analysis Of The Load Flow Behaviour Near A Jacobian Singularity",IEEE transection,1991

[9] Federico Milano. "Documentation for PSAT version ". IEEE, July 14,2005.

[10] Bhadra S N, Kastha D , Banerjee S. "Optimal Power Flow Based on Hybrid Genetic Algorithm". Information Science And Engineering, 2007.

[11] STAGG G. W. and AL-ABIAD A., 1968, Computer Methods in Power System Analysis, Mc-Graw Hill Publishing Company.

[12] Hadi Sadat,Power system analysis,2nd edition.

[13] Pohlheim H.,2005, GEATBx: Genetic and Evolutionary Algorithm Toolbox for Use with Matlab, available at http://www.geatbx.com/.

Index

Abstract, vi

Background, 1

Basic techniques for power-flow studies, 10

Basic Terminology In GA, 18

Binary V/S Floating Coding Approach, 23

Certificate, iv

Comparison of load flow methods, 16

Components Of a Continuous Genetic Algorithm, 23

FDLF method, 15

GA in LF, 29

GA In LF details, 30

GA Programming, 54

General power Formulation, 12

GS Iterative Method, 13

IEEE 14 Bus System, 44

Introduction To GA, 17

N-R Method, 13

Objective Of LF study, 9

Out line of the Basic GA, 19

Testing of method on 5 Bus system, 39

Types Of Buses, 11

Types of GA Operator, 21