OPTIMISATION METHODS APPLIED TO COMPENSATOR PLACEMENT

Ignatius Burger

Thesis presented in partial fulfilment of the requirements for the degree of Master of Science of Engineering at the University of Stellenbosch.

Supervisor: Dr H.J. Beukes

September 2004

DECLARATION

I, the undersigned, hereby declare that the work contained in this thesis is my own original work and that I have not previously in its entirety or in part submitted it at any university for a degree.

Signature: _____

Date: _____

SUMMARY

The optimal placement of different types of compensators on electrical networks is a complex task faced by network planners and operations engineers. The successful placement of these devices normally involves a large number of power flow studies and relies heavily on the experience of the engineer. Firstly the operation and application of the different types of compensators must be clearly understood. Secondly the application of combinations of different compensators on a specific network must be investigated. Then the dynamics of the network and interaction between the network and the compensator/s must be studied under a wide variety of network conditions and load levels. This task is further complicated by the non-linear nature of the mathematical equations that govern the power flow and voltage distribution on an electrical network. Yet another complication is the fact that some of the variables that describe an electrical network can be non smooth or discrete. For instance, the discrete value of a tap position of a power transformer can only assume an integer value. To simplify the problem of compensator placement, advanced software tools are available that are capable of solving power flows of networks containing compensators. To a large degree, however, these tools still rely on the user to make intelligent decisions as to the configuration of networks and the placement of compensators. In many cases trial and error is the only way to find a good solution.

The purpose of this thesis is to show the different techniques available to implement intelligent algorithms capable of finding optimal solutions specific to the placement of voltage regulators. State of the art algorithms are implemented in Matlab that can place voltage regulators on sub transmission, reticulation and low voltage networks. The sub transmission and reticulation placement algorithm is a combination of an SQP technique and a simple combinatorial algorithm. The low voltage placement program is based on a simple genetic algorithm with a few customized features that has been developed to ensure fast convergence. The programs developed were used to do optimal voltage regulator placement on a number of networks. As far as possible real world networks were used. Where real world networks were not available test networks were used that closely resemble real networks, as they exist on typical networks owned by Eskom Distribution.

It was found that SQP is a very efficient algorithm for optimising large non-linear problems such as the placement of a Step Voltage Regulator on a large electrical network. This algorithm however does not handle discrete variables very well and is also limited in handling

any reconfiguration of the network due to the placement of series devices such as voltage regulators. To cater for reconfiguration, it is necessary to combine the SQP algorithm with a combinatorial algorithm.

The genetic algorithm used to do optimal placement of multiple Electronic Voltage Regulators on low voltage networks was found to be very efficient and robust. This can be attributed to the simplicity of the algorithm as well as the fact that it does not rely on the availability of derivative and second derivative information to move towards an optimal solution. Instead, it only uses fitness values obtained from function evaluations to optimise the placement problem. Another useful feature of using a genetic algorithm is that the algorithm does not get stuck in sub optimal areas in the solution space. Both the placement programs developed are relatively simple and do not consider all the factors involved in the placement of voltage regulators. However, the addition of any number of factors is however possible with further development of the programs as presented in this thesis.

OPSOMMING

Die optimale plasing van verskillende kompenseerders op elektriese kragstelsels is ´n moeilike probleem vir beplanners en operasionele personeel. Die plasing van kompenseerders gaan gewoonlik gepaard met ´n groot hoeveelheid netwerk studies en die sukses daarvan berus gewoonlik op die ondervinding van die ingenieur. Eerstens moet die werking en toepassing van elke kompenseerder behoorlik verstaan word. Tweedes moet die plasing van ´n enkele asook kombinasies van verskillende kompenseerders ondersoek word. Dan moet die dinamika van die netwerk en interaksie met die kompenseerder/s bestudeer word vir al die moontlike netwerk konfigurasies en belasting vlakke. Die taak word verder bemoeilik deur die nie-liniêre vorm van die wiskundige vergelykings wat die netwerk vrag en spanning verspeiding beskryf. Nog ´n komplikasie is die feit dat van die veranderlikes wat die probleem beskryf, diskreet is. Byvoorbeeld die tap posisie van ´n transformator kan slegs ´n heel getal aanneem. Om die plasing van kompenseerders te vergemaklik is gevorderde sagteware beskikbaar wat simulasies kan doen van netwerke wat kompenseerders bevat. Tot ´n groot mate is die sagteware nog steeds afhanklik van intellegente besluitneming deur die gebruiker. In die algemeen moet ´n groot hoeveelheid studies nog steeds gedoen work om ´n goeie oplossing te vind.

Die doel van hierdie tesis is om die verskillende tegnieke te wys wat beskikbaar is om intelligente algoritmes te implementeer wat optimale oplossings kan vind vir spesifiek die plasing van spanning reguleerders. Moderne algoritmes is in Matlab geimplementeer wat spanning reguleerders op sub transmissie, retikulasie en laag spanning netwerke kan plaas. Die sub transmissie en retikulasie plasings algoritme is gebaseer op ´n kombinasie van ´n sekwensieële kwadratiese programmering metode en ´n eenvoudige kombinatoriese metode. Die laag spanning plasings program is gebaseer op ´n eenvoudige genetiese algoritme met ´n paar unieke verstellings om vinnige konvergensie te verseker. Die twee programme wat ontwikkel is word dan gebruik on spanning reguleerders te plaas op ´n paar netwerke. So ver moontlik is bestaande netwerke gebruik. Waar bestaande netwerke nie beskikbaar was nie is toets netwerke saamgestel wat gebaseer is op bestaande Eskom netwerke.

Daar is gevind dat sekwensieële kwadratiese programmering ´n effektiewe algoritme is om groot nie liniêre optimerings probleme, soos die plasing van spanning reguleerders, op te los. Hierdie algoritme is egter nie geskik om diskrete veranderlikes te hanteer nie. Dit is verder ook nie geskik om enige netwerk rekonfigurasie te hanteer tydens die plasing van series

geskakelde kompenseerders soos spanning reguleerders nie. Om die rekonfigurasie moontlik te maak is dit nodig om die sekwensieële kwadratiese programmering te kombineer met ´n kombinatoriese algoritme. Daar is verder gevind dat die genetiese algoritme wat gebruik is om elektroniese spanning reguleerders te plaas op laag spanning netwerke baie effektief en robuust is. Dit is as gevolg van die eenvoudigheid van die algoritme en die feit dat dit nie afhanklik is van afgeleide en tweede afgeleide informasie om na die optimale oplossing te beweeg nie. Die algoritme gebruik slegs fiksheid waardes bereken van funksie evaluasies om die probleem te optimeer. Nog ´n voordeel van genetiese algoritmes is dat dit nie in sub optimale gebiede van die oplossing ruimte stil staan nie.

Beide die programme wat ontwikkel is, is redelik eenvoudig en neem nie al die faktore in ag wat gepaard gaan met die plasing van spanning reguleerders nie. Addisionele faktore kan egter maklik ingesluit word deur verdere ontwikkeling van die bestaande programme.

TABLE OF CONTENTS

List of figures

LIST OF TABLES

LIST OF ABBREVIATIONS

ADMD	After Diversified Maximum Demand
FACTS	Flexible ac Transmission Systems
GUI	graphical user interface
PCC	point of common coupling
PCCL	point of common coupling at load
PCCR	point of common coupling at receiving end
PCCS	point of common coupling at sending end
PSAT	Power System Analysis Tool
PSST	Power System Simulation Tool
PST	Phase Shifting Transformer
PAR	Phase Angle Regulator
PSSE	Power System Simulation for Engineers
SVC	Static Var Compensator
UPFC	Unified Power-Flow Controller
DPL	DigSilent Programming Language
SVR	Step Voltage Regulator
EVR	Electronic Voltage Regulator
SGAVR	Simple Genetic Algorithm for Voltage Regulators
δ	phase angle of the sending-end voltage
ac	alternating current
A	Matrix of constraint normals, Jacobian matrix
a	vector
$a_i\ i = 1,2,..$	Set of vectors (columns of A)
A^T, a^T	transpose
x	Variables in an optimization problem
$x^{(k)}, k = 1,2,..$	Iterates in an iterative method
$\delta, \delta^{(k)}$	Correction to $x^{(k)}$
$f(x)$	Objective function
$s, s^{(k)}$	Search direction (on iteration k)
$\alpha, \alpha^{(k)}$	Step length (on iteration k)

∇	First derivative operator (elements $\frac{\partial}{\partial x_i}$)
$g(x) = \nabla f(x)$	Gradient vector
∇^2	Second derivative operator (elements $\frac{\partial^2}{\partial x_i \partial x_j}$)
$G(x) = \nabla^2 f(x)$	Hessian matrix (second derivative matrix)
C^k	Set of k times continuously differentiable functions
$[a,b]$	Closed interval
(a,b)	Open interval
L	Lagrangian function
A	Set of active constraints
$c(x)$	Vector of constraint functions
$a_i(x) = \nabla c_i(x)$	Constraint gradient vector
λ	Lagrangian multiplier
$q^{(k)}(\delta)$	Model quadratic function approximated by Taylor expansion
ρ	Weighting parameters in penalty function
∇_x, ∇_λ	Partial derivative operators w.r.t x and λ respectively
$W = \nabla_x^2 L(x,\lambda)$	Hessian of Lagrangian function
$\psi(x)$	Penalty function
E	Set of equality constraint
I	Set of inequality constraints
$\Re^n$	n-dimensional space
$\in$	Element of a set

ACKNOWLEDGEMENTS

I would like to convey my gratitude to the people and institutions that made this project and thesis possible:

Dr Johan Beukes for his invaluable support and guidance;

ESKOM for their financial support.

CHAPTER 1 INTRODUCTION

Mathematical optimisation can be defined as the process of finding the best way to achieve certain objectives while maintaining certain constraints. This 'best way' is called the optimal solution to the optimisation problem and can be either in the form of either a maximum or a minimum. In the field of power flow studies this objective can be in the form of maximising certain voltage levels or power transfer of a power network, or to minimise the cost of a compensator or minimise technical losses on a power network. Optimisation techniques refer to all the available methods that are used to find a set of design parameters that can, in some way, be defined as optimal. One approach to finding the optimal solution to a problem is to evaluate the objective function for all possible combinations of solutions that make up the solution space and then, by a simple process of comparing, identify the optimal solution. The purpose of optimisation techniques is to limit this massive search through the entire solution space to only certain parts of it in order to increase the efficiency of the search.

Advances in computer development since the seventies have led to an increase in popularity of using optimisation techniques to solve complex mathematical problems. The reason for this increase in popularity was that, for the first time, it enabled real world problems in many (thousands of) variables to be solved in a reasonable time. These capabilities were largely due to the increase in computation speed and capacity of computers at the time. In turn, this revolution in computer technology led to a resurgence in development work on new optimisation techniques. Today, mathematical optimisation is a vast field of study with many applications in finance, science and engineering. Different optimisation classes and methods exist, which solve many kinds of problems in industry. These optimisation classes are well structured and each display unique characteristics, features, advantages and disadvantages. The efficiency of solving an optimisation problem depends very much on the solution approach followed. To make an informed decision on how to solve a complex problem with optimisation techniques, it is important to understand the underlying optimisation theory as well as to have an up-to-date knowledge of the different optimisation methods available. It is further important to fully understand the characteristics of the problem at hand as well as the requirements of the solution with respect to speed and accuracy. It is also important to recognise that different optimisation methods can be combined to solve a particular problem or alternatively, the problem can be broken down into sub problems and each sub problem can be solved separately using different optimisation techniques.

In the field of power flow studies, optimisation has found only a handful of applications. Most of the applications have been aimed at high voltage transmission networks where the aim is to minimise generation cost and/or technical losses. Recently developed combinatorial optimisation techniques, such as genetic algorithms and the Tabu search method, have been applied to solve the optimal placement of shunt capacitors on radial distribution networks. The University of Stellenbosch has developed software that uses non-linear optimisation techniques to evaluate the application of different compensator technologies on power networks. The aim of this software is to help the network planner to quickly evaluate a number of possible network solutions and thereby minimise the number of detailed and time consuming studies that have to be done. This software, which is MatLab based and makes use of the MathWorks Optimisation Toolbox, has been further improved upon and used in this thesis to evaluate certain compensator placements. This thesis further explores the underlying optimisation principles in order to apply them more efficiently.

The study of optimisation, as with many other disciplines in mathematics, is accompanied by a large amount of complex notation and extended mathematical proofs, which can make the subject very difficult to understand and time consuming. It is, however, very important to have a good theoretical background of optimisation in order to understand and apply the different optimisation techniques efficiently. Chapter 2 of this thesis aims at illustrating the necessary optimisation principles used in later chapters. Where possible, this is done by supplying illustrations in one or two-dimensional examples. By doing this, the related objective and constraint functions can be visualized as three-dimensional surfaces, which add insight to the mathematical formulations. Along with the objective and constraint surfaces, the gradients, contours and surface normals can also be visualized. These are the tools that are used in many optimisation algorithms and once visualized, simplify the understanding of more complex optimisation algorithms.

To structure the study of optimisation techniques, it is useful to catagorise the different methods of solving optimisation problems according to certain features of the problem to be solved. This leads to the development of standard techniques that are used to solve problems in standard form. To solve a problem as an optimisation problem, it is necessary to first convert the original problem into one of the available standard optimisation problem forms and then decide on one of the available methods to solve that type of standard problem. This ‘standard form’ will largely be determined by the mathematical equations that describe the

objective function as well as that describing the constraints. The method that is decided upon to solve a specific problem might also depend on other factors such as the size of the problem, the accuracy of the solution required and the speed of the solution required. In some cases, different methods might perform similarly, while in other cases there might be vast differences in accuracy and performance. It is, therefore, important to understand the problem at hand as well as the theory and application of the solution methods applied to the problem to ensure that the end result is satisfactory.

Chapter 3 describes the application of optimisation techniques in the field of power flow studies. Important issues are handled such as the classic optimal power flow problem as well as other related work and tools available for further development work. Existing software developed within this research, as well as the MathWorks Optimization Toolbox, is analysed in detail to gain insight into its operation and the optimisation algorithms used.

Chapter 4 deals specifically with the optimal placement of voltage regulators. The chapter describes in detail the two programs that were developed to solve the placement of voltage regulators on medium voltage reticulation networks and single phase low voltage radial networks. Both programs make use of modern optimisation techniques to solve the respective problems.

Chapter 5 gives a number of specific applications of optimisation techniques applied to the placement of compensators on existing electrical networks. The first case study is the placement of a Phase Angle Regulator to control active power transfer on a sub transmission network. This is a real network on the Eskom Distribution grid in the Eastern Cape that suffers from thermal overloading. The second and third case studies cover the placement of Step Voltage Regulators on typical Eskom reticulation networks. The fourth case study is the placement of multiple Electronic Voltage Regulators on a single phase low voltage electrification network.

Chapters 6 and 7 conclude the thesis by proposing further work and making closing remarks. Much of the work done in this thesis was done from fundamental concepts to prove certain concepts of applying optimisation techniques. Although the two programs developed are functional, they still need further development work to be of commercial use. With further work, as well as co-operation with commercial software suppliers, the ideas presented can be implemented in the industry.

CHAPTER 2 OPTIMISATION THEORY

2.1 OVERVIEW

The purpose of this chapter is to analyse the optimisation methods used in the software developed as part of this thesis. The first program developed uses Sequential Quadratic Programming to place Step Voltage Regulators on medium voltage radial networks and the second program uses a genetic algorithm to place Electronic Voltage Regulators on radial low voltage networks.

Optimisation is a vast field of study and algorithms are divided into different classes according to the characteristics of the problems they can solve. This chapter first explain general optimisation principles and then discuss unconstraint and constraint optimisation separately in more detail. Generally, gradient search methods are used to solve unconstraint optimisation problems and the Lagrange methods are used to solve constraint problems. Non smooth optimisation is then covered and the genetic algorithm is discussed specifically and in detail.

2.2 SMOOTH OPTIMISATION FUNDAMENTALS

Smooth optimisation refers to the optimisation of problems involving variables that are continuous and differentiable. This type of optimisation theory has been developed over many years, either by improving existing methods or by the development of new methods. However, even the state of the art smooth optimisation techniques, for example, Sequential Quadratic Programming (SQP) is based on the same fundamental mathematical concepts originating from classical Calculus. It is extremely important to understand these concepts in order to solve real world problems with smooth optimisation methods. This section explains some of the fundamental concepts of smooth optimisation that will provide a good basis on which to solve smooth optimisation problems as well as to understand more complex theory and algorithms discussed in later chapters. The rest of this section will refer to smooth optimisation as just optimisation.

Optimisation was needed because the use of classical Calculus to solve optimisation problems is restricted to functions that are piecewise continuous and differentiable. Even in these cases, solutions found using Calculus are restricted to problems of low dimensionality [B23]. There was thus a need to develop methods to solve complex problems efficiently and this has led to the development of what is now known as mathematical optimisation.

Optimisation problems can be catagorised into different types according to many different characteristics. One of the most basic differentiations is between constraint and unconstraint problems. As the name indicates, a constraint problem is one where the allowable variable space of the objective function is restricted to a specific region by another set of functions called the constraint functions. A general constraint problem can be stated in standard form as follows:

$$\text{Extremise } f(x) \text{ with respect to } x \quad x \in \Re^n \qquad \text{Objective function} \qquad (2.1)$$

$$\text{with } x = (x_1, x_2, ..., x_n)$$

Subject to

$$c_i(x) = 0, \quad i \in E \qquad \text{Equality constraint functions} \qquad (2.2)$$

$$c_i(x) \geq 0, \quad i \in I \qquad \text{Inequality constraint functions} \qquad (2.3)$$

where E is the set of equality constraints and I is the set of inequality constraints. The number of equality constraints in an optimisation problem must be less than the number of variables x, otherwise the problem will be either uniquely determined by the equality constraints or the problem will be over specified. In comparison, an unconstraint optimisation problem is one where the variable space of the objective function is not restricted to a certain area. Due to this uncomplicated structure the unconstraint optimisation problem is generally easier to solve.

The region in the solution space defined by the objective function and the constraint functions in a constraint problem is called the feasible region. Any point within this region is called a feasible solution to the optimisation problem. In an unconstraint problem all points are feasible solutions. A particular feasible solution that extremises the objective function is called the optimal feasible solution. Optimisation techniques can also be classified as global optimisation techniques or as local optimisation techniques according to their ability to find a global optimal solution or not.

When optimising an objective function it is important to realize that a function might possess more than one extremum within a certain region. Each of these extrema can be defined as the extremum of a sub region of the original region and are called relative or local extrema. The largest of these points is called the absolute or global extremum. This concept is illustrated for an objective function in two variables in Figure 2-1.

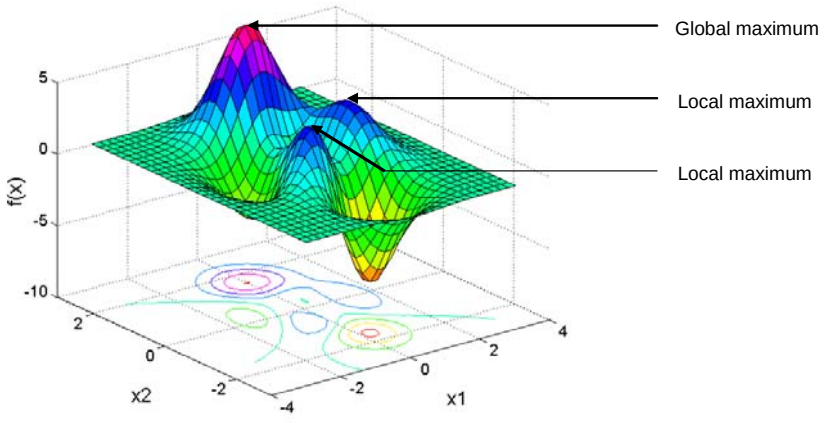


Figure 2-1: Illustration of local and global maxima

In general non-linear optimisation, it is often very difficult to establish whether a certain optimal point is a relative optimal point or a global optimum point and in some cases, an

intelligent starting point is necessary to ensure a globally optimal solution. In other cases the structure of the problem guarantees a given optimal solution to be global. For instance a unimodal optimisation problem has only one relative maximum in a specified region. This is a very useful property to have since it implies that a local maximum is necessarily also a global maximum. An example of a one dimensional unimodal function is shown in Figure 2-2. As shown in this figure a unimodal function does not have to be continuous or differentiable [B23].

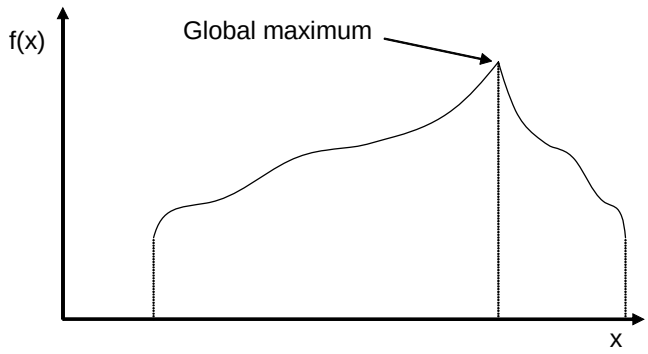


Figure 2-2: One dimensional unimodal function

A closely related concept to that of a unimodal function is that of convexity and concavity. A strictly convex function has the useful characteristic that it takes on one and only one absolute minimum in a specified region. The value of such a function $f(x)$, at some intermediate point, $x_1 < x < x_2$, is equal or less than the weighted average of $f(x_1)$ and $f(x_2)$. Convexity can be stated mathematically as follows:

$$f\{ax_1 + (1-a)x_2\} \le af(x_1) + (1-a)f(x_2) \tag{2.4}$$

with $0 < a < 1$. This concept is shown in Figure 2-3. A convex function must be continuous but does not have to be differentiable.

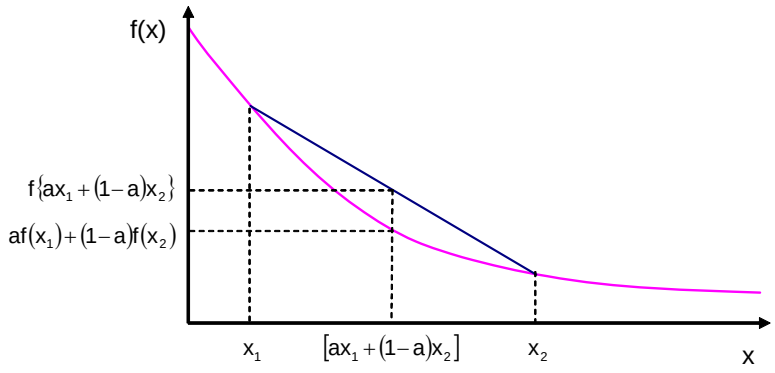


Figure 2-3: A typical convex function

Analogous to convexity, a concave function has one and only one relative maximum in a specified region. Mathematically this can be stated as follows:

$$f\{ax_1 + (1-a)x_2\} \geq af(x_1) + (1-a)f(x_2) \tag{2.5}$$

with $0 < a < 1$. This can be seen in Figure 2-4. Similar to a convex function, a concave function must be continuous but does not have to be differentiable.

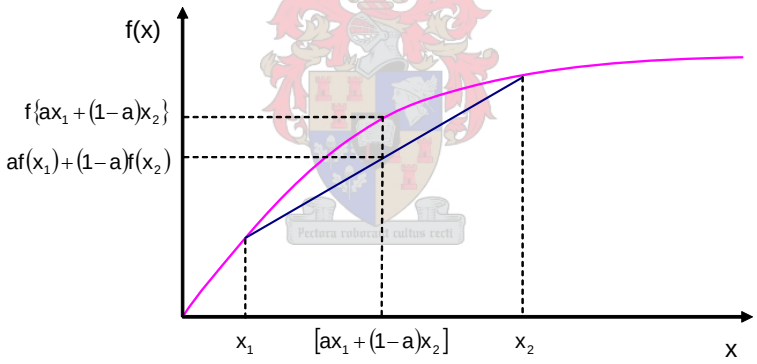


Figure 2-4: A typical concave function

If convexity or concavity cannot be used to determine if a solution is a global optimal solution another method must be used to prove the solution to be global. One way to do this is to restart the optimisation process from different starting points and then compare the results. Other general techniques used are to search for all the relative extreme points within the region of interest and then to compare the objective function at all these points and systematically determine the global optimal solution.

Optimisation techniques can also be classified according to the way in which they determine its search points. Techniques that use historic values to determine the next search points in an effort to improve the objective function are referred to as sequential optimisation whereas

techniques that do not use historic values to determine the next search points are referred to as simultaneous optimisation techniques [B23].

From the above discussion it is clear that optimisation techniques can be classified according to many different characteristics. A very useful way is to categorise the different techniques according to the type of problem they can solve. Figure 2-5 shows a functional classification of optimisation techniques that can be used to decide on a technique to solve a particular problem.

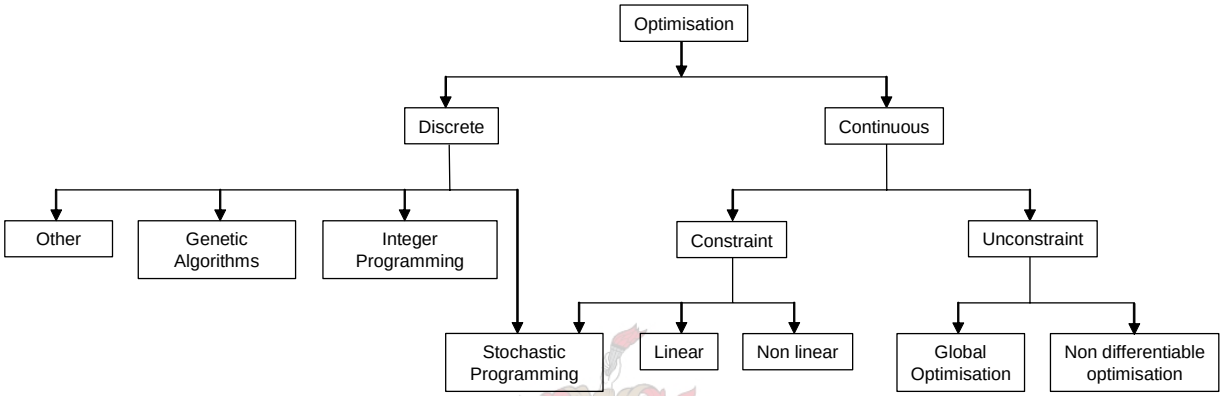


Figure 2-5: Optimisation tree

One of the very basic and very important theorems in the study of optimisation is the Weierstrass Theorem. This theorem states that every function that is continuous in a closed region possesses a largest and a smallest value within the interior or on the boundary of that region [B23]. This theorem implies that when searching for an optimal solution, it is important to search both the interior of the specific region as well as the boundary of the region. If a function is piece-wise continuous it can be divided into sections in which the Weierstrass Theorem can be applied. Although this theorem is a good starting point in the search for optimality it does not give any useful information about the location or the nature of the extreme points.

Another important feature of optimisation theory is that it can be applied to either a minimisation problem or a maximisation problem. This is because of the simple relationship that exists between a minimisation problem and a maximisation problem. This relationship can be mathematically stated as follows: $\min\{f(x)\} = -\max\{-f(x)\}$ and is illustrated in Figure 2-6.

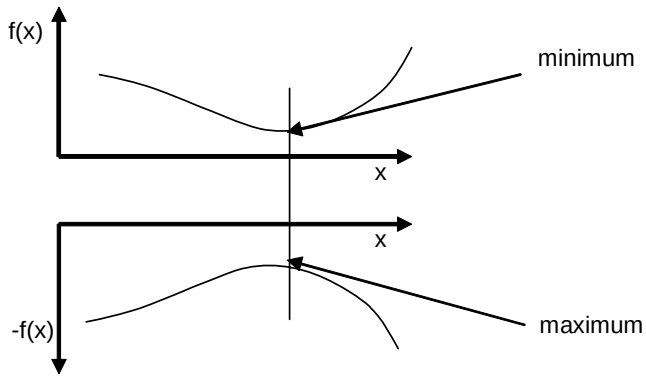


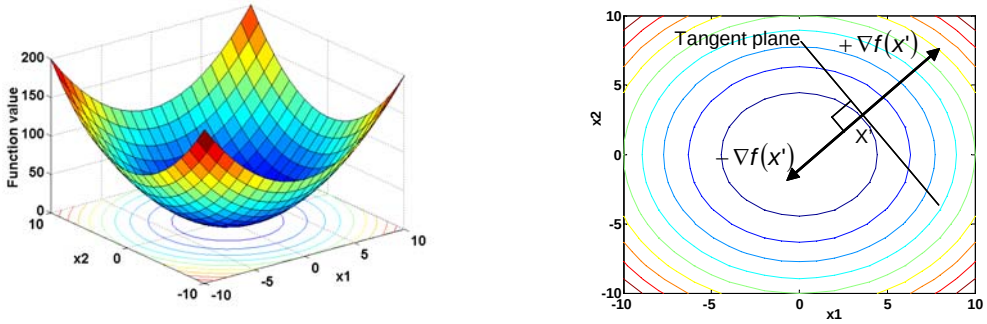
Figure 2-6: Equivalence of minimum and maximum problem

Stated differently a minimisation problem can be converted into an equivalent maximisation problem and vice versa. It is therefore only necessary to derive theory for one of the two.

In some problems it might be necessary to optimise more than one objective function. Such a problem can be solved by using a multi-objective approach, which is a separate field of optimisation theory and is not covered in this thesis. Alternatively the different objective functions can be combined into a single objective function. The method is much simpler and is implemented by a strategy called value theory. Value theory is the method used to incorporate multiple objectives into a single objective function and is especially useful when there is no mathematical relationship between the different variables. For example in the case of finding the optimal configuration of a power system it might be necessary to consider factors such as cost, network security, quality of supply and reliability into a single objective function. This difficult problem can be handled by specifying an objective function of the form: $U = w_1U_1 + w_2U_2 + w_3U_3 + w_4U_4$ with w_i a weighting factor that assigns a relative importance to the corresponding factor and with $0 < w_i \leq 1$. As an example U_1 may be the cost variable, U_2 the network security, U_3 the quality of supply and U_4 the reliability variable. This formulation of a multi-variable objective function has been used successfully in the optimisation software developed in this thesis.

In solving optimisation problems two of the most important concepts are that of the gradient vector and a line. A general strategy in optimisation methods is to use the gradient vector to determine a search direction and then to use the concept of a line to move a certain distance in that direction towards the next search point. The gradient vector, $\nabla f(x')$, has the property that it points in the direction of greatest accent of the function $f(x')$ at the point x' [B25]. Analogous to this, the negative gradient vector $-\nabla f(x')$, points in the direction of greatest

decent of the function $f(x')$. It is obvious that this is very useful information to have in the search for either a minimum or a maximum of an objective function. The gradient vector indicates a good direction to move into but does not say anything about how far must be moved. Other properties of the gradient vector are that it is orthogonal to the contours of $f(x')$ as well as to the tangent plane at the point of evaluation x' of the function $f(x')$. These properties of the gradient vector can be seen in Figure 2-7.



(a) (b)

Figure 2-7: (a) 2 dimensional function (b) Contours, gradient and tangent plane

With the gradient defined, the concept of a line must also be introduced. A line can be defined as a set of points $x(\alpha) = x' + \alpha s$ for all α with x' a fixed point and s the direction of the line. In optimisation it is often necessary to calculate the gradient of a function along a line. If the gradient of the function is indicated by ∇f, the gradient of this function along a line in direction s is given by $\nabla f \cdot s$.

The Taylor series is another very important tool in optimisation and forms the basis of the Newton type methods. The Taylor series expansion of a multi-variable function can be stated as follows:

$$f(\alpha) = f(0) + \alpha f'(0) + \frac{1}{2}\alpha f''(0) + \ldots \tag{2.6}$$

If the point around which the series is to be calculated is other than the origin, the Taylor series can be expressed as:

$$f(x' + \alpha s) = f(x') + \alpha s^T \nabla f(x') + \frac{1}{2}\alpha^2 s^T \left[\nabla^2 f(x')\right] \cdot s \tag{2.7}$$

To illustrate the use of the Taylor series in optimisation, a simple function (2.8) will be minimised by approximation as a quadratic function.

$$f(x) = x^3 - 2x - 5 \tag{2.8}$$

The starting point is chosen as $f(6) = 199$. In this example $q(x)$ is the Taylor approximation of $f(x)$. The result after three iterations can be seen in Figure 2-8.

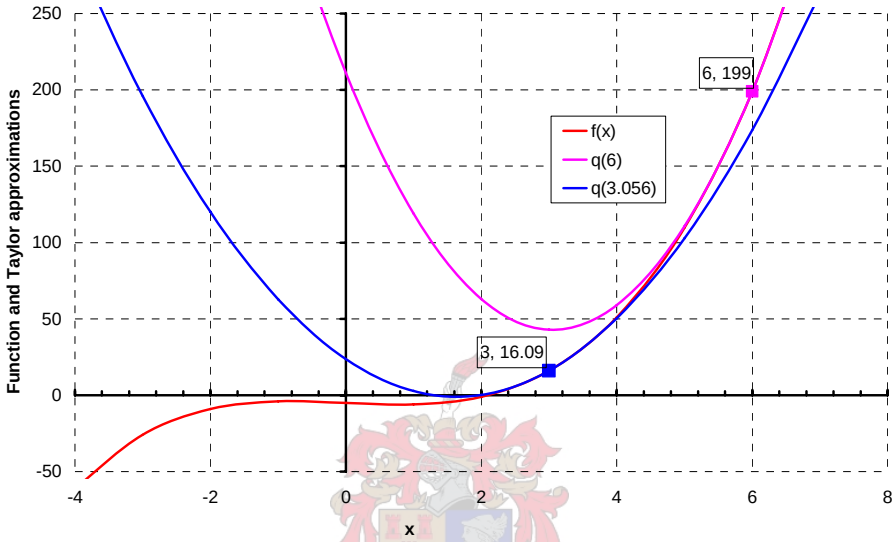


Figure 2-8: Illustration of the Taylor series expansion

By using the Taylor expansion as explained above general non-linear functions can be approximated as linear or quadratic functions. Most techniques for unconstraint optimisation are based on a model of the objective function that is based on this approximation. Of these, the quadratic model has been proven to be the most successful [B25]. Unconstraint optimisation techniques are also based on a prototype algorithm. The two prominent algorithm types are the trust region type and the line search type of which only the latter will be discussed in this thesis. The main motivation for using the latter is to prevent non positive Hessian matrices when approximating objective functions with the Taylor expansion.

The rest of Chapter 2 discusses in detail smooth unconstraint and constraint optimisation in Sections 2.3 and 2.4 respectively and then discusses non smooth optimisation in Section 2.5. The focus of the smooth optimisation methods is the SQP techniques used in the Step Voltage Regulator placement program developed in this thesis. The focus of the non smooth optimisation methods are the genetic algorithm that is used in the Electronic Voltage Regulator placement program developed in this thesis.

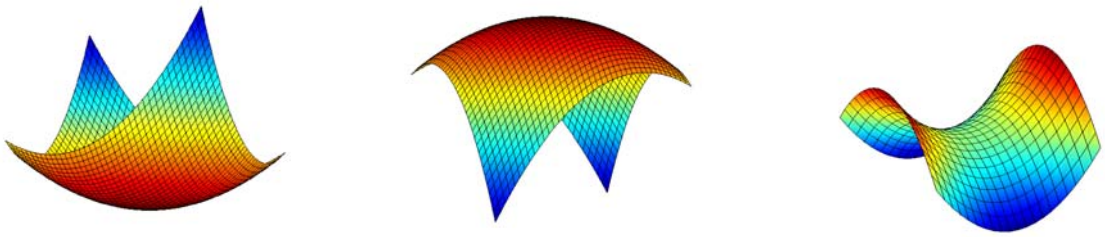
2.3 UNCONSTRAINT OPTIMISATION

2.3.1 Overview

Unconstraint optimisation refers to the group of numerical searching techniques used to find the extremum of an unconstrained objective function. It is important to notice that most of the searching techniques can only locate local extrema. Although this in itself is very useful, the methods developed for unconstraint optimisation are also used in constraint optimisation. Often a constraint problem can be solved by solving a series of unconstraint sub problems.

2.3.2 Fundamentals of unconstraint optimisation

A function that varies continuously over an open interval with its first derivative equal to zero and its second derivative smaller than zero has a relative maximum at that point. The first derivative of a function is a measure of the slope of the function and the second derivative is a measure of the curvature of the function. If the second derivative is greater than zero, the function has a relative minimum at that point. If the second derivative is equal to zero, the behaviour at this point is unspecified [B23]. In such a case, a Taylor expansion can be used at that point to evaluate further. Figure 2-9 illustrates the three main types of extreme points that a function can assume.



(a) (b) (c)

Figure 2-9: (a) Minimum (b) Maximum (c) Saddle point

The saddle point clearly shows that a vanishing gradient does not guarantee a minimum or a maximum. Once all the extrema have been evaluated within the region of interest, the function must also be evaluated on its boundaries. In summary, it can be said that a necessary condition for a function $f(x')$to have a local extremum at point x' is $\nabla f(x') = 0$. Such a point might be a local maximum, a local minimum or a saddle point. A sufficient condition for a local minimum is $\nabla f(x') = 0$ and $\nabla^2 f(x')$ is positive definite. In the case of multi-dimensional functions, the extrema is first determined inside the region of interest. This is

done by evaluating the Hessian matrix of the objective function at all the stationary points inside the region for positive definiteness [B23]. The Hessian matrix is a matrix of all the second partial derivatives of the function. The Hessian matrix's form is shown here:

$$G = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_{\frac{2}{2}}^{2}} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \cdot & \cdot & & \cdot \\ \\ \cdot & \cdot & & \cdot \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_{\frac{2}{n}}^{2}} \end{bmatrix} \tag{2.9}$$

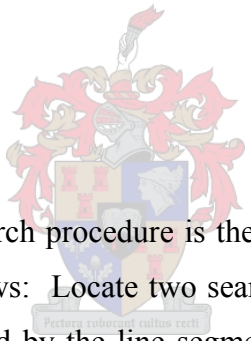
Thereafter, the function is evaluated on the region boundaries for extrema. These calculated local boundary extrema are then compared with the extrema obtained inside the region and the absolute or global extremum can be calculated.

Multi-dimensional functions that are continuous and differentiable and whose extrema lie at an interior point of a region can be optimised by using gradient techniques. These techniques make use of derivative information about the function to determine optimal search directions. Some of the most efficient gradient techniques available are the steepest decent method, the conjugate gradient method and quasi Newton methods. The optimal solution of a multi-dimensional function that is not differentiable is found by using direct search methods. In these methods, only function evaluations are used to determine search directions to obtain extrema. An early example of a direct search method is the *pattern search* developed by Hooke and Jeeves (1961) [B23]. Another class of optimisation techniques that do not rely on derivative information being available are the *random search* techniques. Unlike the gradient type methods, the random search techniques are not sequential searching techniques. They fall into the class of simultaneous programming techniques as the search points are not determined from historical values. A good example of a simultaneous programming technique is the genetic algorithm that is used in the Electronic Voltage Regulator placement program for low voltage networks.

The line search is one of the most important concepts in optimisation and is used in a wide variety of practical algorithms. The power of the line search is due to its simplicity. For

example, finding the maximum of a one dimensional, unimodal function can be done using the following simple method. Select a few search points, evaluate the function at these points and determine the maximum. From this information, a smaller interval can be defined within which the true maximum will be. This process is repeated until this smaller interval, called the interval of uncertainty, is small enough. All line search procedures consist of two distinctive parts. The first is the bracketing phase, which determines an interval (or bracket) that contains the extremum. The second part of the procedure is the sectioning phase. In this phase, the bracket is sequentially divided into brackets with decreased length. In the limit, the bracket length tends to zero and the solution is thus found. More advanced line search algorithms also include a form of interpolation whereby typically a quadratic polynomial is fitted. Subsequently, the minimum of this polynomial is calculated. Many different line search techniques exist and can be catagorised according to whether derivative information is available or not. Different search procedures that do not use derivative information exist, of which the following are a few:

- Symmetrical two point search
- Fibonacci search
- Golden-ratio search



The most basic one-dimensional search procedure is the Symmetric Two Point search. This procedure can be described as follows: Locate two search points within the given region of interest. Let this region be indicated by the line segment *a*-*b* in Figure 2-10. Initially, the interval of uncertainty is *a*-*b*. Now let the two initial search points be x_1 and x_2. Let e indicate the spacing between x_1 and x_2. Also let x_1 and x_2 be spaced symmetrically around the centre of the region. Now the function is evaluated at x_1 and x_2. From this information, the new interval of uncertainty is chosen. If e is small, this new interval of uncertainty will be approximately half the size of the original interval. It can easily be shown that increasing the size of e or having x_1 and x_2 unsymmetrical leads to an increased size of the interval of uncertainty [B23]. The size of the interval of uncertainty is calculated as follows:

$$L_{2k} = \frac{b-a}{2^k} + \left(1 - \frac{1}{2^k}\right)e \tag{2.10}$$

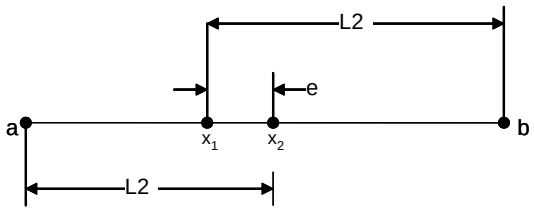


Figure 2-10: Illustration of the symmetrical two point search

One dimensional search procedures as explained above are used in more complex optimisation algorithms to find optimal solutions to multi-dimensional problems.

2.3.3 Multi dimensional optimisation techniques for unconstraint functions

Methods to solve unconstraint optimisation problems can be categorized according to the derivative information used. Methods that only use function evaluations are best suited to optimise functions that are very non-linear or have discontinuities. Gradient methods are best suited for optimising functions that are continuous in the first derivative. Methods that use higher order derivative information are only effective when the second order derivatives are given or easily calculated without the need for numerical differentiation. Gradient methods use the slope of the objective function to determine effective search directions. Two of the most powerful gradient techniques available for the optimisation of unconstraint problems are the method of steepest descent and the conjugate gradient method. Both methods are iterative and have the general form:

$$x_{k+1} = x_k + \alpha_k s_k \tag{2.11}$$

with α_k a scalar quantity that gives the distance along a specific search direction s_k and with x_k the present evaluation and x_{k+1} the next evaluation. The rest of this section is dedicated to the theory behind both these methods. The application of both methods will also be illustrated by examining a simple problem. In both cases, the method will be explained for minimising an objective function although exactly the same method can be used for maximising an objective function. The section then concludes with an overview of another class of unconstraint optimisation technique used for multi-dimensional functions called the Quasi-Newton method.

Steepest decent method: The steepest decent method minimises a multi-dimensional unconstraint function *f(x)* by following a path of decent starting at an initial point. Such a path is characterised by having a slope along the path that is negative and defined by (2.12) where *s* is defined as a smooth curve on the objective surface.

$$\frac{\partial f}{\partial s} < 0 \tag{2.12}$$

It is obvious that a good path to follow will be the path of steepest decent starting from some initial point. The path of steepest decent is found to be that defined as follows:

$$\frac{\partial x_i}{\partial s} = -\left\{ \sum_{i=1}^{n} \left(\frac{\partial f}{\partial x_i} \right)^2 \right\}^{-\frac{1}{2}} \frac{\partial f}{\partial x_i}, \qquad i = 1,2,...n \tag{2.13}$$

This formula for the path of steepest decent can now be used in a minimisation algorithm as in (2.14) with ∇f defined by (2.15).

$$x_{k+1} = x_k + \{(\nabla f)'(\nabla f)\}^{-\frac{1}{2}} \cdot \nabla f \cdot \Delta S \tag{2.14}$$

$$\nabla f = \frac{\partial f}{\partial x_i}, \qquad i = 1,2,..n \tag{2.15}$$

and

ΔS is a user specified scalar value

As can be seen, this algorithm takes on the general form:

$$x_{k+1} = x_k + \alpha_k s_k \tag{2.16}$$

with α_k a scalar quantity that gives the distance along a specific search direction s_k. The $\{(\nabla f)'(\nabla f)\}^{-\frac{1}{2}}$ term in (2.14)(2.17) is a scalar quantity and can therefore be incorporated with ΔS into a single term $\Delta\tau$. This leads to a simplified form of the algorithm that can be expressed as follows:

$$x_{k+1} = x_k + \nabla f \Delta\tau \tag{2.17}$$

With $\Delta\tau$ defined as follows:

$$\Delta\tau = \{(\nabla f)'(\nabla f)\}^{-\frac{1}{2}}\Delta S \tag{2.18}$$

The algorithm consists of the following steps:

1. Choose a starting point
2. Calculate the direction of steepest decent at this point as: $-\nabla f$
3. Move in the direction of steepest decent a distance specified as follows:

$$\Delta\tau = \{(\nabla f)'(\nabla f)\}^{-\frac{1}{2}}\Delta S \tag{2.19}$$

4. If this new point is not the minimum, go back to step 2. If the point is in fact the minimum, the algorithm is terminated. The distance (2.20) is the product of the calculated value, $\{(\nabla f)'(\nabla f)\}^{-\frac{1}{2}}$, and a user specified value ΔS.

$$\Delta\tau = \{(\nabla f)'(\nabla f)\}^{-\frac{1}{2}}\Delta S \tag{2.20}$$

As it is, the step length will be proportional to the gradient of the objective function, scaled by the user-specified value ΔS. The effect of this is that big steps are taken where the object function is steep and small steps are taken on flatter parts of the objective surface. This can lead to undesirable oscillations around the extreme point. Alternatively, the step size can be kept constant and equal to a user specified value namely $\Delta\tau = \Delta S$.

Because the steepest decent method forms such an important part of smooth optimisation theory, an example has been included to illustrate exactly how it works. The following example shows the important steps of the steepest decent method applied to a simple non-linear problem.

Example 1: Illustration of the method of steepest descent.

Problem:

Minimise

$$f(x) = (x_1 - 3)^2 + 9(x_2 - 5)^2 \tag{2.21}$$

starting at $x_0 = (1,1)$

$\Delta\tau = \Delta S$ chosen as 0.1

The gradient is calculated at the starting point: $\nabla f|_{x0} = -\begin{bmatrix} 4 \\ 72 \end{bmatrix}$

Therefore $x_1 = (1,1) - 0.1 * -\begin{bmatrix} 4 \\ 72 \end{bmatrix} = (1.4,8.2)$

At this point, the gradient is recalculated: $\nabla f_{x1} = -\begin{bmatrix} -3.2 \\ 57.6 \end{bmatrix}$

And x_2 is calculated as $(1.72, 2.44)$

At x_2 the gradient is calculated as: $\nabla f|_{x2} = -\begin{bmatrix} 2.56 \\ 46.08 \end{bmatrix}$

And x_3 is calculated as $(1.976, 7.04)$

The minimisation process after two steps is shown in Figure 2-11. In the figure, the sequence of one-dimensional searches is shown by vertical planes. Although still a distance away from the minimum point, it can be seen that movement is made towards the minimum. It can also be seen that convergence is slow due to the zig zag steps that are taken. This kind of oscillatory behavior is typical of the steepest descent method and obviously slows down the rate of convergence. To illustrate another very important method of smooth optimisation, the same problem will be solved using the conjugate gradient method.

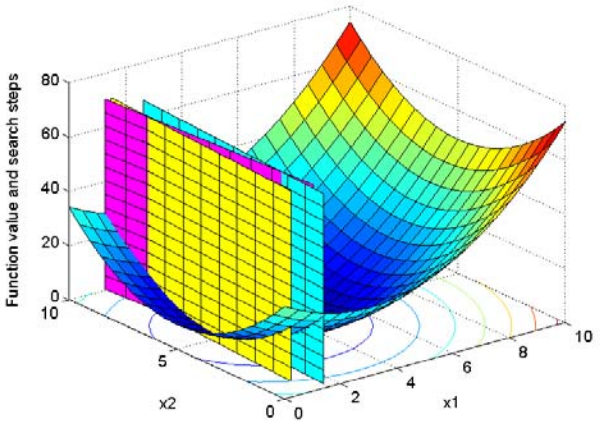


Figure 2-11: Three steps of the steepest descent method

Conjugate gradient method: Like the steepest descent method, the conjugate gradient method uses a sequence of one-dimensional searches towards the minimum of the objective function. In the case of the conjugate gradient method, the direction of each search is determined by a formula that is based on the partial derivative information of the objective function. This formula is a combination of the current gradient vector and the previous search

vector. As with the steepest descent method, this method also takes on the general form given by (2.22) with α_k a scalar quantity that gives the distance along a specific search direction s_k.

$$x_{k+1} = x_k + \alpha_k s_k \tag{2.22}$$

As stated earlier, the only difference comes in with the calculation of the search direction s_k. The method of conjugate gradients is based on the fact that at the minimum along the search direction s_k, the search direction will be perpendicular to the gradient of the objective function. This is necessarily the case because the dot product of two perpendicular vectors is zero as shown by (2.23).

$$\nabla f_{k+1} \cdot s_k = 0 \tag{2.23}$$

At this point, which is the minimum along the current search direction, a new search direction is calculated and used for the next search towards the minimum of the objective function. The formula to determine the search direction s_k is now determined. For the first search, the search direction is simply taken as the negative of the gradient as follows:

$$s_0 = -\nabla f_0 \tag{2.24}$$

Similar to the steepest decent method, this is done because no history is available about previous search directions. The subsequent search directions are calculated as follows:

$$s_{k+1} = -\nabla f_{k+1} + \beta_k s_k \tag{2.25}$$

with $k = 0,1,2,...,n-1$. It is shown here without proof that the choice of β_k equal to:

$$\beta_k = \frac{\nabla f_{k+1}' \nabla f_{k+1}}{\nabla f_k' \nabla f_k} \tag{2.26}$$

will ensure that a quadratic objective function in *n* variables is minimised in no more than *n* iterations. The above statement only guarantees good convergence for a quadratic objective function and does not say anything about objective functions of other forms. To cater for general non-linear functions, it is necessary to make use of the Taylor expansion of such

functions. It can easily be shown that for the general case, the method will closely approximate the quadratic form near the solution point. The Taylor expansion of a function *f(x)* can be expressed as follows:

$$f(x) = f(x_0) + \nabla f\Big|_{x_0}(x - x_0) + \frac{1}{2}(x - x_0)' G(x - x_0) + \ldots \tag{2.27}$$

Where G is defined as follows:

$$G = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x \frac{2}{2}} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \cdot & \cdot & & \cdot \\ & & & \\ \cdot & \cdot & & \cdot \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x \frac{2}{n}} \end{bmatrix} \tag{2.28}$$

At the solution point all the partial derivatives will be zero i.e. $\nabla f\Big|_{x_0} = 0$ so that equation (2.27) above becomes (2.29) if the higher order terms are neglected. This is proof that the method will be well-behaved close to the solution point.

$$f(x) - f(x_0) = \frac{1}{2}(x - x_0)' G(x - x_0) \tag{2.29}$$

The following example shows the important steps of the conjugate gradient method applied to a simple non-linear problem.

Example 2: Illustration of the conjugate gradient method.

Problem:

Minimise (2.30) starting at $x_0 = (1,1)$.

$$f(x) = (x_1 - 3)^2 + 9(x_2 - 5)^2 \tag{2.30}$$

Because the objective function is quadratic, the minimum of any one-dimensional move can be calculated by solving the following:

$$f(\alpha) = a\alpha^2 + b\alpha + c \tag{2.31}$$

This function is minimized as follows:

$$\alpha = \frac{-b}{2a} \tag{2.32}$$

Solution:

The gradient is calculated at the starting point as:

$$\nabla f|_{x0} = -\begin{bmatrix} 4 \\ 72 \end{bmatrix} \tag{2.33}$$

Using (2.22) the next point is calculated as:

$$x_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} + \alpha_0 \begin{bmatrix} 4 \\ 72 \end{bmatrix} \tag{2.34}$$

So α_0 is easily calculated using (2.32) to obtain the next point as:

$$x_1 = \begin{bmatrix} 1.223 \\ 5.011 \end{bmatrix} \tag{2.35}$$

The gradient is recalculated at this point as:

$$\nabla f|_{x1} = -\begin{bmatrix} -3.55 \\ 0.197 \end{bmatrix} \tag{2.36}$$

From (2.26) β_0 is calculated as:

$$\beta_0 = \frac{(3.55)^2 + (0.197)^2}{(4)^2 + (72)^2} = 0.00244 \tag{2.37}$$

The new search direction can now be calculated as:

$$S_1 = \begin{bmatrix} 3.554 \\ -0.197 \end{bmatrix} + 0.00244 \begin{bmatrix} 4 \\ 72 \end{bmatrix} = \begin{bmatrix} 3.564 \\ -0.022 \end{bmatrix} \tag{2.38}$$

and X_2 is calculated using (2.22) again,

$$x_2 = \begin{bmatrix} 1.223 \\ 5.011 \end{bmatrix} + \alpha_1 \begin{bmatrix} 3.564 \\ -0.022 \end{bmatrix} \tag{2.39}$$

α_1 is easily calculated using (2.32) to obtain $x_2 = \begin{bmatrix} 3 \\ 5 \end{bmatrix}$.

This is the minimum point of the objective function. Figure 2-12 shows these two steps from the initial point to the solution point. As can be seen, this is a vast improvement on the result obtained from the steepest descent method.

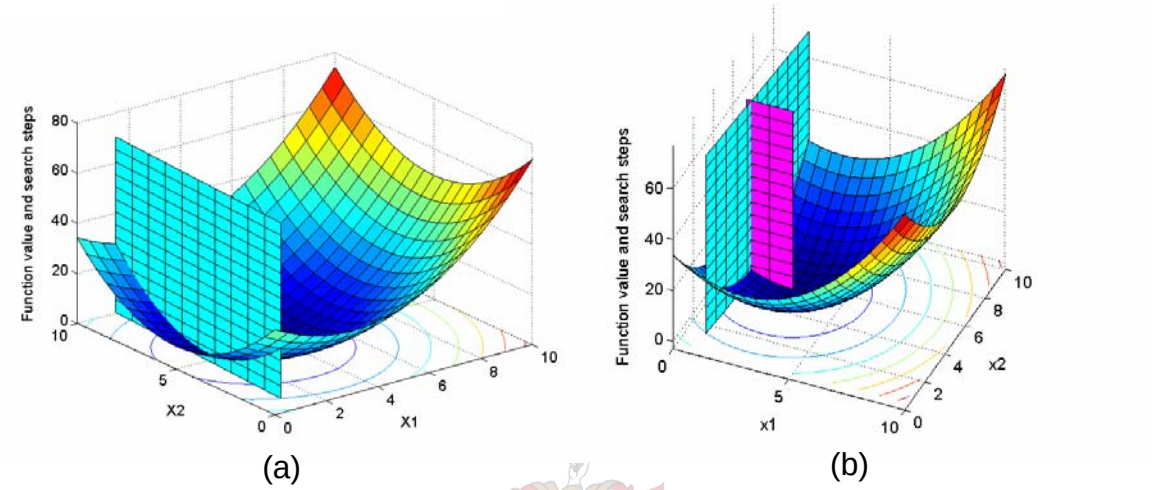


Figure 2-12: (a) First and (b) second step of the conjugate gradient method

Quasi-Newton method: The Quasi-Newton method uses gradient information of the objective function to build up a quadratic model at each iteration in the form given by (2.40) where G is the Hessian matrix and ^{T} indicates the transposition of a matrix and where c is a constant vector and b is a constant scalar.

$$\min_x \frac{1}{2} x^T G x + c^T x + b \tag{2.40}$$

Furthermore G is a positive definite matrix. From normal calculus, the optimal solution is calculated by setting the partial derivatives of x equal to zero as follows:

$$\nabla f(x^*) = Gx^* + c = 0 \tag{2.41}$$

The solution can then easily be found as $x^* = -G^{-1}$. It is clear that the inverse of the Hessian matrix needs to be calculated in order to solve for the optimal point. Two methods are available to obtain the Hessian matrix namely the Newton method and the Quasi Newton method. The Newton method calculates the Hessian directly at each iteration of the move towards the minimum point. This numerical calculation of the Hessian matrix is

computationally very intensive and therefore other methods have been developed to avoid it. The Quasi-Newton method avoids this calculation by approximation of the Hessian matrix. Many different methods exist for updating the Hessian matrix. For general problems, one of the most effective methods is the BFGS method by Broyden, Fletcher, Goldfarb and Shanno [B26]. This method is formulated as given by (2.42) [B26]. Here H indicates an estimation of the Hessian matrix G.

$$H_{k+1} = H_k + \frac{q_k q_k^T}{q_k^T s_k} - \frac{H_k^T s_k^T s_k H_k}{s_k^T H_k s_k} \tag{2.42}$$

Where s_k and q_k are defined as follows:

$$s_k = x_k + 1^{-x_k} \tag{2.43}$$

$$q_k = \nabla f(x_{k+1}) - \nabla f(x_k) \tag{2.44}$$

Using this formula, an approximation of the Hessian is made at each iteration of the search, starting from an initial point H_0. From the previous assumption that H is a positive definite matrix, the starting point H_0 must be set to a positive definite matrix. Because the inverse of the Hessian matrix will actually be used to calculate the search direction, it will be useful to calculate this inverse directly. A similar method as described above is available for this approximation namely the DFP formula of Davidson, Fletcher and Powell [B26]. The gradient information needed in the above methods is obtained though analytical calculation or by deriving it numerically. The second part of the quasi-Newton method is to do a line search in the direction specified by: $d = -H_k^{-1} \nabla f(x_k)$. The line search is done in order to find the minimum along the line formed by the search direction that was calculated above. This is normally done by approximation using a search technique or by using inter- or extrapolation.

This concludes the discussion on unconstraint optimisation techniques. Section 2.4 continues the discussion of smooth optimisation by looking specifically at smooth optimisation problems that are constrained by equality and inequality constraints.

2.4 CONSTRAINT OPTIMISATION

2.4.1 Overview

Constraint optimisation refers to the group of numerical searching techniques used to find the extremum of a constrained objective function. This section discusses the properties of such problems as well as the general approach of how to solve them. A specific method to solve general constraint optimisation problems called Sequential Quadratic Programming is also discussed.

2.4.2 Fundamentals of constraint optimisation

A general constraint optimisation problem in standard form can be expressed as follows:

$$\begin{gathered}\text{Extremise } f(x) \text{ with respect to } x, \quad x \in \Re^n \\ \text{with } x = (x_1, x_2, ..., x_n) \\ \text{subject to} \\ c_i(x) = 0, \quad i \in E \\ c_i(x) \geq 0, \quad i \in I\end{gathered} \tag{2.45}$$

Constraints in optimisation can either be in the form of equality constraints or inequality constraints. If only inequality constraints are present, the number of degrees of freedom of the objective variables equals the number of variables. If equality constraints are present, the number of degrees of freedom reduces by one for each equality constraint present.

An objective function that is constrained by a set of equalities can be converted to an unconstraint problem by the application of the Lagrangian function [B23]. This method states that the constraint optimisation problem is equivalent to finding the stationary values of the unconstraint Lagrangian function. A necessary condition is that the Lagrangian function must have the necessary concavity and convexity in the vicinity of the stationary point. The Lagrangian function is constructed as:

$$\begin{gathered}L(x_1, x_2, ..., x_n, \lambda_1, \lambda_2, ..., \lambda_m) = f(x_1, x_2, ..., x_n) + \sum_{i=1}^{m} \lambda_i c_i(x_1, x_2, ..., x_n) \\ \text{with } i \in E\end{gathered} \tag{2.46}$$

The above method for equality constraint optimisation can be extended to include inequalities. This is done by converting the inequality constraints to equality constraints by

the application of slack and/or surplus variables. This method is described by the Kuhn-Tucker conditions [B23]. In practice, inequality constraints can also be handled by treating them as equality constraints in areas where they are active. To implement this, a list of active inequality constraints must be maintained.

A special kind of constraint optimisation problem is the linear program. In the linear program both the objective function and the constraint functions are linear, and in standard form must have all variables positive and the number of equality constraints must be less than the number of variables. The linear programming problem can be stated in standard form as in (2.47) where A is an $m \times n$ matrix and $m \le n$.

$$\min_{x} \quad f(x) \qquad (2.47)$$

$$\text{with } f(x) = c^T(x)$$

$$\text{subject to } Ax = b, \quad x \ge 0$$

Inequalities can be converted to equalities by the use of slack and/or surplus variables. After this has been done, the linear programming problem can be stated in standard form. Values of x that satisfy the constraint functions and bounds, $Ax = b, \quad x \ge 0$, are called feasible solutions. Feasible solutions that extremise the objective function $f(x) = c^T(x)$ are called maximum or minimum feasible solutions. Since the objective function is linear, it does not contain any curvature that can generate an extreme point. Thus an extreme point can only occur on the boundaries of the feasible region. At this point, at least $m - n$ variables will have the value of zero and the other m variables will be uniquely determined by the equations $Ax = b, \quad x \ge 0$. This is illustrated in Figure 2-13.

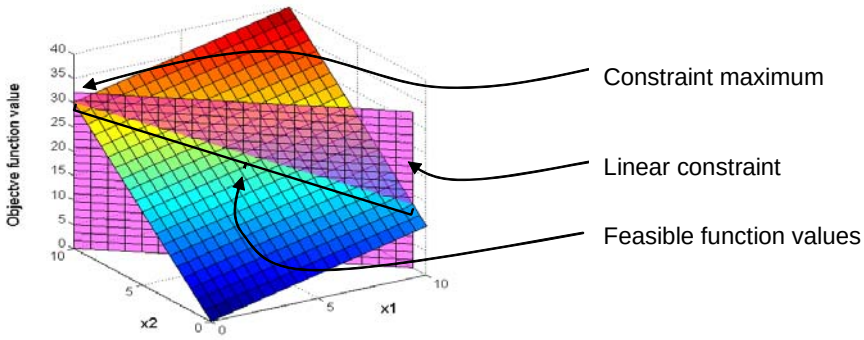


Figure 2-13: Illustration of a linear program

This means that for a linear programming problem in standard form, the solution will always exist at a particular extremum point of the feasible region. The basic method of solving linear programming problems is by an algorithm known as the simplex algorithm. The operation of this algorithm falls outside the scope of this thesis and will therefore not be discussed. A good reference for more information on this algorithm is [B24].

One of the most important concepts in constraint optimisation is that of the Lagrangian function and the associated Lagrangian multipliers. In unconstraint problems, the gradient vector and Hessian matrix of the objective functions are used to define conditions for optimality. The Lagrangian concept enables the definition of similar optimality conditions for constraint functions. The basic idea is that at the constraint extremum, the gradient of the objective function will be parallel to the gradient of the constraint functions [B25]. This concept is illustrated in Figure 2-14.

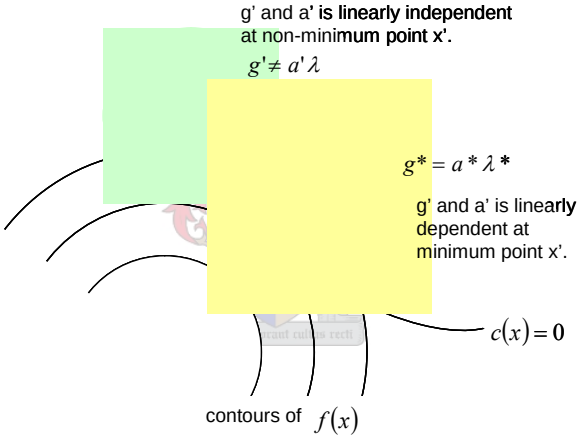


Figure 2-14: Illustration of the Lagrange multiplier

The governing equation becomes:

$$g^* = \sum_{i \in E} a_i^* \lambda_i^* = A^* \lambda^* \tag{2.48}$$

where

g^* is the gradient vector of the objective function at the local extremum x^*,

a^* is the gradient vector of the constraint functions at the local extremum x^* and

λ^* is the vector of Lagrangian multipliers at the local extremum x^*. The vectors x^* and λ^* are obtained by solving equations (2.49) and (2.50).

$$\begin{aligned} g(x) &= \sum_{i \in E} a_i(x)\lambda_i \\ c_i(x) &= 0, \quad i \in E \end{aligned} \tag{2.49}$$

$$L(x,\lambda) = f(x) - \sum_i \lambda_i c_i(x) \tag{2.50}$$

From these equations the Kuhn – Tucker conditions for optimality can be derived as follows:

$$\nabla_x L(x,\lambda) = 0 \tag{2.51}$$

$$c_i(x) = 0, \quad i \in E \tag{2.52}$$

$$c_i(x) \geq 0, \quad i \in I \tag{2.53}$$

$$\lambda_i \geq 0, \quad i \in I \tag{2.54}$$

$$\lambda_i c_i(x) = 0 \quad \forall i \tag{2.55}$$

Just as the quadratic form plays a very important role in unconstraint optimisation it also plays an important role in constraint optimisation. With the use of the Taylor series expansion, methods to optimise quadratic functions can be applied to more general functions. The following is an overview of how the Lagrangian function is used to optimise a constraint quadratic function of the form given by:

$$f(x) = \frac{1}{2} x^T G x + gx \tag{2.56}$$

subject to the following constraints:

$$\begin{aligned} Ax - b &= 0 \\ Cx - d &\leq 0 \end{aligned} \tag{2.57}$$

For simplicity, only equality constraints will be considered. Given the optimisation problem above, the Lagrangian function can be written as:

$$L(x,\lambda) = \frac{1}{2} x^T G x + g^T x - \lambda^T \left(A^T x - b \right) \tag{2.58}$$

The gradient of the Lagrangian is calculated as:

$$\nabla L = Gx + g + A^T \lambda \tag{2.59}$$

Together with the original linear equality constraint $Ax - b = 0$ the following set of linear equations are defined:

$$\begin{bmatrix} G & -A \\ -A^T & 0 \end{bmatrix} \begin{pmatrix} x \\ \lambda \end{pmatrix} = -\begin{pmatrix} g \\ b \end{pmatrix} \tag{2.60}$$

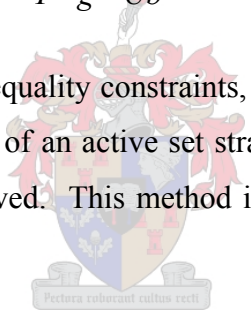
The coefficient matrix is called the Lagrangian matrix. This is a symmetric matrix and it is convenient to calculate the inverse of this matrix as:

$$\begin{bmatrix} G & -A \\ -A^T & 0 \end{bmatrix}^{-1} = \begin{bmatrix} H & -T \\ -T^T & U \end{bmatrix} \tag{2.61}$$

The solution is then as follows:

$$\begin{aligned} x^* &= -Hg + Tb \\ \lambda^* &= T^T g - Ub \end{aligned} \tag{2.62}$$

If the quadratic problem includes inequality constraints, a more advanced approach has to be followed. One possibility is the use of an active set strategy. In this method, a sequence of equality constrained problems is solved. This method is discussed and illustrated in section 2.4.3.



2.4.3 Minimising a function with inequality constraints

The purpose of this section is to show how the active set strategy is used to solve constraint optimisation problems involving inequality constraints. This method forms an integral part of more complex optimisation algorithms such as SQP as used in the MathWorks Optimization Toolbox. The active set strategy is explained for a quadratic objective function and linear constraints. For the more general case where both the objective function and the constraints can be non-linear, it is necessary to approximate such non linearities by using the Taylor series expansion. As stated earlier, the active set method makes it possible to handle inequality constraints in optimisation. It does this by solving a sequence of equality constraint sub problems with the equalities chosen as a sub set of the original set of inequality constraints. This active set of equality constraints is constructed by the algorithm at each iteration according to a certain set of rules. Each equality constraint sub problem is solved by using the Lagrange conditions for optimality. This leads to a set of linear equations that can

easily be solved. The Lagrange conditions for optimality can be stated as follows [B25]: The Lagrangian function must be solved with all of the Lagrange multipliers being non negative and all the constraints must be satisfied. The algorithm will be given within the framework of the following assumptions:

- Let x_i indicate the solution to the current iterate.
- Let A indicate the active set of constraints in the current iteration.
- The index p is used for a constraint not currently in the active set, but entering the active set in the next iteration.
- The index q is used for a constraint currently in the active set, but leaving the active set in the next iteration.
- The index δ is used to indicate the correction from one iteration to the next. For example, the correction between $x_3 = (2,6)$ and $x_4 = (5,-10)$ is $\delta = (3,16)$.

The quadratic sub problem will be solved in terms of this correction variable. The effect of solving the problem in terms of the correction variable δ instead of the original variables x_k is a shift of the origin to the new location x_k. The effect of this shift in origin is that the constraints simplifies from $Ax \geq b$ to $A\delta \geq 0$. Equations (2.63) to (2.65) will be used in the algorithm.

$$\min \frac{1}{2}\delta^T H\delta + \delta^T g_k$$

$$\text{subject to} \qquad (2.63)$$

$$a_i^T \delta = 0, \quad i \varepsilon A$$

$$\alpha_k = \min\left(1, \min_{\substack{i:i\notin A \\ a_i^T s_k < 0}} \frac{b_i - a_i^T x_k}{a_i^T s_k}\right) \qquad (2.64)$$

$$\min_{i \in A \cap I} \lambda_{ik} \qquad (2.65)$$

The algorithm to implement the active set strategy is given by the following seven steps:

1. Given x_0 and A, set $k = 0$

2. If $\delta = 0$ does not solve (2.63), go to step 5

3. Calculate Lagrange multipliers λ_k and solve (2.65)

4. If $\lambda_{qk} \geq 0$ terminate with $x^* = x_k$, otherwise remove q from A

5. Solve (2.63) for s_k

6. Find α_k to solve (2.64) and set $x_{k+1} = x_k + \alpha_k s_k$

7. If $\alpha_k < 1$, add p to A where p is a previously inactive constraint. Set $k = k + 1$ and go to step 2

Although this algorithm might seem complex, it is a very logical sequence of steps, which is most easily understood when applied to a sample problem. Such a sample problem is worked through in detail in Addendum D. The problem was taken from literature [B2].

2.4.4 Sequential quadratic programming

Sequential Quadratic Programming is currently the most widely used algorithm to solve smooth non-linear constraint optimisation problems [B29], [B25]. The basic idea behind the algorithm is to apply the Khun-Tucker conditions for optimality in an iterative manner on an approximation of the original problem. At each iteration, an approximate quadratic programming problem is solved. The solution to this sub problem is used to determine a search direction. This search direction is then used to do a line search. The Hessian matrix is approximated at each iteration. The quadratic programming problem that is solved at each iteration is an approximation of the original problem by a quadratic form of the objective function and the linearisation of the constraint functions. These approximations lead to an optimisation problem of form given as follows:

$$\begin{gathered} \min_x f(x_k) + \nabla f(x_k)^T d + \frac{1}{2} d^T \nabla^2 L(x_k) d \\ \text{subject to} \\ c_i(x_k) + \nabla c_i(x_k)^T d = 0, \; i \in E \\ c_i(x_k) + \nabla c_i(x_k)^T d \leq 0, \; i \in I \end{gathered} \tag{2.66}$$

with the Lagrangian given by (2.67) and where λ is the approximation of the Lagrange multipliers of the equality constraints and μ is the approximation of the Lagrange multipliers of the active inequality constraints. Equation (2.66) uses a substitution of the Lagrangian

instead of the objective function in the second order term. This is done to include curvature information about the constraint functions into the quadratic sub problem.

$$L(x) \stackrel{\Delta}{=} f(x) + \lambda^T c_{i,i\in E}(x) + \mu^T c_{i,i\in I}(x) \tag{2.67}$$

In the Lagrangian (2.67), the Hessian is approximated by applying an approximation algorithm such as the BFGS method. The solution of the above quadratic sub problem gives a search direction $s_k = d_k$. This search direction is then used in a line search to calculate the next iterate as in (2.68)

$$x_{k+1} = x_k + \alpha s_k \tag{2.68}$$

where α is a scalar quantity that specifies the length of the search in the direction s_k. The calculation of the search length parameter, α, presents another complication in the algorithm because of the constraints on the objective function. Because of the constraints, the search length must be calculated to maintain feasibility in addition to minimising the objective function. This can be seen as a multi-objective problem that can be expressed as a single objective problem by applying value theory in the form of a penalty function. This search length, α, is calculated by solving the following penalty function:

$$\psi(x) = f(x) + \rho\left[\sum\left|c_{i,i\in E}(x)\right| + \sum\max\left\{0, c_{i,i\in I}(x)\right\}\right] \tag{2.69}$$

where ρ is a penalty factor. As can be seen from (2.69) the penalty function will decrease in value if the objective function value, $f(x)$, decreases or if the penalty function decreases. A decrease in penalty function means a movement towards the feasible region. Intuitively this makes sense and can be proven mathematically. The mathematical proof can be found in books covering modern non-linear optimisation techniques [B25].

2.5 NON SMOOTH OPTIMISATION

2.5.1 Overview

Sections 2.2 and 2.3 discussed in detail the basic techniques used to deal with smooth optimisation problems. In general, these techniques rely heavily on gradient and curvature information to determine search directions, which are used to move towards the optimal point. A prerequisite of using such methods is that both the objective and constraint functions must be continuous and differentiable throughout the solution space. However, practical problems however often have variables that are not continuous and differentiable. An example of this in power systems is the discrete nature of transformer tap positions or the effect that the reconfiguration of a power network has on the load flow equations that describe it. There are various ways of handling discrete variables and discontinuities in optimisation and the method used is largely dependant on the structure of the problem. The following are a few methods available to handle discontinuities.

- Approximations/curve fitting
- Rounding off of variables to the closest discrete value
- Non smooth optimisation methods

Non smooth type algorithms have become very popular because of the fact that many real world problems contain discrete variables and/or other discontinuities in the solution space. Other advantages of non smooth optimisation methods are that no gradient or Hessian information is needed. This is a major advantage as the calculation of these quantities can be computationally very intensive or may not exist at all. Another advantage is their inherent ability to find global optimal solutions and to not get trapped in sub optimal parts of the solution space. Although many non smooth optimisation techniques are available, only the genetic algorithm will be discussed in detail in this thesis. The main reason for this decision is the success of the algorithm applied to the placement of shunt capacitor banks [B14] as well as the availability of a Matlab Toolbox for genetic algorithms. This toolbox was a great help in the development of the placement program for LV Electronic Voltage Regulators (EVRs). Another reason for the use of genetic algorithms is the successful application in other fields of electrical engineering. The genetic algorithm is in its basic form a very simple algorithm and good results have been obtained when applied to the optimal placement of EVRs on complex radial low voltage networks. This section discusses genetic algorithms in more detail.

2.5.2 Genetic algorithms

Although optimisation problems can be solved by a simple iterative trial and error method, the number of possible combinations quickly grows and it becomes impractical to try all combinations to arrive at a solution within a reasonable time. A simple trial and error method can be enhanced by adding some heuristic rules to narrow down the number of combinations. In general, it is difficult to structure such rules especially when the algorithm is expected to solve different types of problems. One way to avoid this problem is to implement a genetic algorithm. A genetic algorithm is a probabilistic search technique, which is based on the principles of biological evolution. In biological evolution, living organisms continually change over generations to produce better offspring. Equivalently genetic algorithms follow paths in the solution space, which improve a specific solution based on the performance of the individuals of previous generations.

Darwin first described the process of evolution in his book [B32]. According to Darwin's theory, nature selects the best genetic settings of an organism to ensure survival in future generations. The offspring are then optimised further to produce successively better offspring. An analogy of this biological process was first applied to the theory of optimisation by Holland [B33] in the form of genetic algorithms. Similar to natural evolution, a genetic algorithm selects the most optimal solutions from a set of solutions, and uses the genetic operators of crossover and mutation to generate further solutions. Each such solution is more optimal than the solutions of the previous generations.

Genetic algorithms have been used successfully in electronics where the concept of embryonic circuits is used to synthesis analogue circuits such as filters and amplifiers [B19]. Embryonic circuits are very minimal circuits designed with a specific problem in mind. A two-component LC circuit, for example, could serve as an embryonic circuit. Using the embryonic circuit as a starting point, the genetic algorithm operates on it with functions that either add a circuit component or chance the configuration of existing circuit components. In the field of power systems, genetic algorithms have been used to do optimal shunt capacitor placement on transmission and distribution networks [B12], [B13] and [B14].

Genetic algorithms are stochastic search methods that simulate the process of natural evolution. This process of evolution is repeated until certain optimisation criteria are met and the evolution is stopped. The structure of a simple genetic algorithm is shown in Figure 2-15.

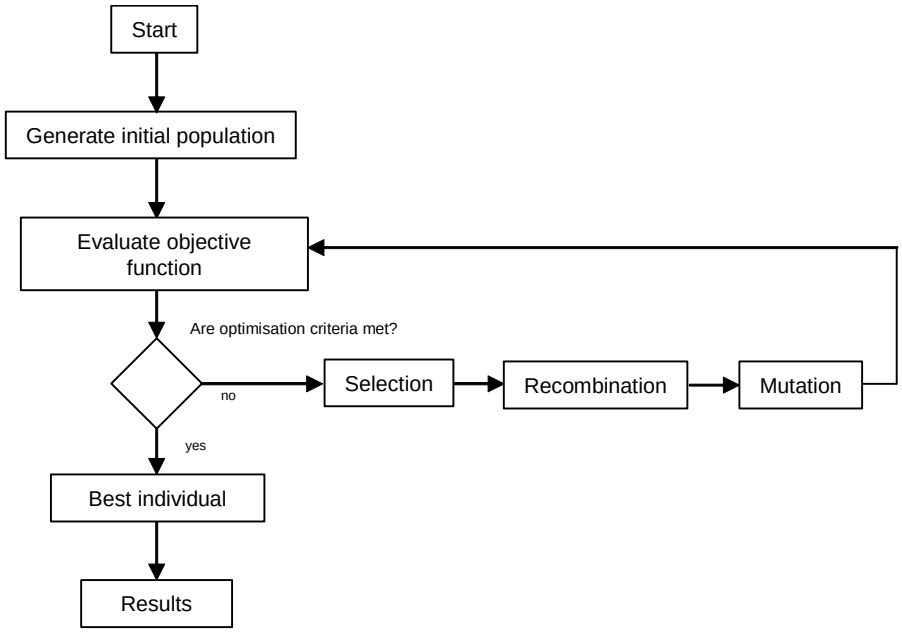


Figure 2-15: Simple genetic algorithm

The genetic algorithm starts by initialising a number of random solutions. A set of solutions like this is called a population. Next, the objective function of the initial population is evaluated, which is tested against the optimisation criteria. If the optimisation criteria are met, the best individual is chosen and the required results are obtained. If the optimisation criteria are not met, a new generation is created. The rest of this section explains the basic mechanisms involved in the creation of the new generations.

Terminology used in genetic algorithms has been adopted from biological genetics. In biological genetics the cells of living organisms contains chromosomes that contain all the information about the whole organism. Chromosomes consist of blocks of DNA called genes and each gene encodes a particular characteristic of the organism. For example, the eye colour of a human. In genetic algorithms chromosomes and genes also exist with each chromosome a possible solution and each gene within the chromosome encoding a specific characteristic of the solution. In the case of compensator placement, the chromosome might represent the placement of compensators on a power network and each gene represents the presence of a compensator at a specific node. Possible settings for a characteristic are called alleles. Each gene also has its own position in the chromosome, which is called the locus. A complete set of genetic material (all chromosomes) is called a genome and a particular set of genes in a genome is called a genotype.

In a genetic algorithm a chromosome is a mathematical representation of the physical solution it represents [B33]. The method by which this representation is implemented is referred to as encoding. Many types of encoding exist, of which binary encoding is one of the most simplistic methods. In binary encoding every chromosome is a string of bits with each being either a zero or a one. Figure 2-16 shows the structure of a binary encoded chromosome.

Chromosome 1	1101100100110110
Chromosome 2	1101111000011110

Figure 2-16: Binary encoding

Each chromosome is represented by a binary string and each bit in the string can represent some characteristics of the solution. The type of encoding used depends mainly on the structure of the problem to be solved. In the case of the EVR placement program, binary encoding is ideal because each bit can represent the presence of an EVR at a specific node. Binary encoding gives many possible chromosomes even with a small number of alleles. Another method of encoding is permutation encoding. In permutation encoding, every chromosome is a string of numbers that represents a position in a sequence. The chromosome will then have the form shown in Figure 2-17.

Chromosome A	1 5 3 2 6 4 7 9 8
Chromosome B	8 5 6 7 2 3 1 4 9

Figure 2-17: Chromosomes with permutation encoding

A third type of encoding is value encoding. Direct value encoding can be used in problems where more complicated values are used such as real numbers. The use of binary encoding for this type of problems would be difficult. In value encoding, every chromosome is a sequence of some values. These values can be anything that is significant to the specific problem, such as (real) numbers, characters or any other objects. This type of encoding is illustrated in Figure 2-18.

Chromosome A	4.5675 2.8867 8.4569 12.3456 42.8522

or

Chromosome B	G B R X P Q D R A N G L J S E

Figure 2-18: Chromosomes with value encoding

Value encoding is a good choice for some special problems. However, for this encoding it is often necessary to develop special crossover and mutation strategies specific to the problem [B43].

The first step in creating a new generation is a process called selection. Individuals within a generation are selected for reproduction or mating according to a specific set of rules. Typically, selection is done according to the individual's fitness. The selected individuals are called the parents and the process of producing offspring is called recombination. Stated differently, chromosomes are selected from the population to be parents for producing offspring by the processes of crossover and mutation. According to Darwin's theory of evolution the best chromosomes survive to create new offspring. Several philosophies are available to implement the process of selection. Examples are Roulette Wheel Selection, Tournament Selection, Rank Selection, Steady State Selection and some others [B43]. The following is a short description of some commonly used strategies.

The simplest type of selection strategy is Roulette Wheel Selection. In Roulette Wheel Selection parents are selected according to their fitness. The better the chromosomes, the greater the probability that they will be selected. As the name indicates, the method is like a roulette wheel where all the chromosomes in the population are placed. The size of the section in the roulette wheel is proportional to the value of the fitness function of each chromosome. The bigger the fitness value is, the bigger the section in the roulette wheel and the greater the chance of the individual being selected. This type of selection is shown in Figure 2-19.

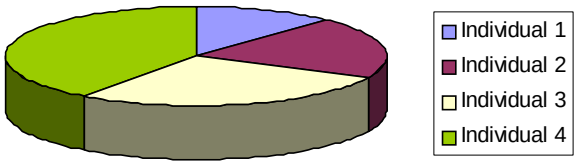


Figure 2-19: Roulette Wheel Selection

Clearly, the chromosomes with bigger fitness values will have a greater chance of being selected.

The problem with Roulette Wheel Selection is that it will be insufficient when individuals have big differences between their fitness values. For example, if the best chromosome's fitness is 90% of the sum of all fitnesses, the other chromosomes will each have a very small

chance of being selected. This problem is solved by a selection strategy called Rank Selection. In Rank Selection, the fitness of a chromosome in a population is first ranked and then each chromosome is assigned a fitness value according to its ranking. The worst chromosome will have a fitness of 1, the second worst a fitness of 2 and so on, and the best will have a fitness equal to the number of chromosomes in the population. The effect of employing rank based fitness instead of roulette based fitness is illustrated in Figure 2-20 and Figure 2-21.

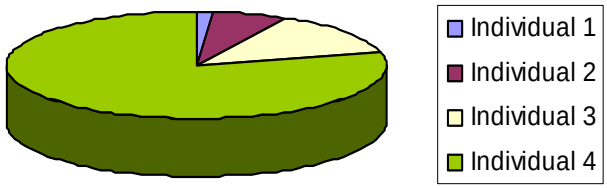


Figure 2-20: Situation before ranking (graph of fitnesses)

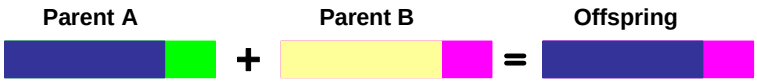
Figure 2-21: Situation after ranking (graph of order numbers)

Rank Based Selection acts as an equaliser and ensures that all the chromosomes have a fair chance of being selected. One possible problem with this strategy is that it can lead to slower convergence, because the best chromosomes do not differ as much from other ones.

Over and above specific selection strategies there are also some general philosophies that can be applied to selection. A good general philosophy in selection is to let a big part of the current generation survive to the next generation [B43]. This is called Steady-State Selection and is implemented as follows: In every generation, a few individuals with high fitness are automatically selected for creating new offspring. Also, some individuals with low fitness are automatically removed and replaced with new offspring. The rest of the population survives to the next generation. Another general procedure to create a new population is to force the best chromosomes to survive to the next generation without crossover and mutation performed on them. This method is called Elitism and it prevents the best chromosomes from

getting lost by first copying the best chromosomes to the new population [B43]. The rest of the population is constructed by the processes of mutation and crossover. Elitism has successfully been implemented in the program to do optimal placement of EVRs on low voltage networks discussed in this thesis.

After encoding and selection, the genetic algorithm continues with the process of crossover. Crossover creates new offspring from parent chromosomes by mixing selected genes from the parents. The simplest way this is done is to randomly choose a crossover point and copy the genes before this point from the one parent and then copy the genes after the crossover point from the other parent. This simple crossover method is illustrated in Figure 2-22 (the color difference shows the crossover point).



11001011+11011**111** = **11001111**

Figure 2-22: Crossover

There are other ways to implement crossover. For example, more crossover points can be selected. Two Point Crossover is illustrated in Figure 2-23.

11001**011** + 11**011**11 = **11011111**

Figure 2-23: Two Point Crossover

Crossover can be quite complicated and depends mainly on the type of encoding of chromosomes. Specific crossover strategies designed for specific problems can significantly improve the performance of genetic algorithms. In the case of the program to do optimal placement of EVRs on low voltage networks, Single Point Crossover has been used.

Crossover and mutation are the two basic operators of genetic algorithms and the performance of a genetic algorithm depends very much on the effectiveness of these two operators. After crossover is performed, mutation takes place. The mutation process is a random process where the genetic makeup of an individual is slightly changed according to certain probabilities. Mutation is intended to prevent all solutions in the population from getting trapped in a local optimum. The type and implementation of mutation depends on the type of encoding used as well as the characteristics of the problem. In the case of binary encoding a

few randomly chosen genes can be changed from zero to one or from one to zero. This simple type of mutation is shown in Figure 2-24.

Original offspring 1	110**1**111000011**1**10
Original offspring 2	110110**0**100110110
Mutated offspring 1	110**0**111000011**0**10
Mutated offspring 2	110110**1**100110111

Figure 2-24: Simple mutation

Another type of mutation can be implemented by changing the order of genes in a chromosome. This is illustrated in Figure 2-25.

Original offspring 1	1 **2** 3 4 5 6 **8** 9 7
Original offspring 2	**6** 8 **9** 3 4 1 2 3 5
Mutated offspring 1	1 **8** 3 4 5 6 **2** 9 7
Mutated offspring 2	**9** 8 **6** 3 4 1 2 3 5

Figure 2-25: Mutation by order changing

When real valued encoding is used mutation can be implemented by adding or subtracting a small number to or from selected values. This type of mutation is illustrated in Figure 2-26.

Original offspring 1	1.29 5.68 **2.86 4.11**
Original offspring 2	**7.89** 3.28 **2.77**..4.48
Mutated offspring 1	1.29 5.68 **2.73 4.22**
Mutated offspring 2	**7.53** 3.28 **2.98**..4.48

Figure 2-26: Mutation by adding

Once the offspring have been generated by the processes described, the offspring can be reinserted into the original population. In this process some or all of the parents might be replaced by the offspring. Depending on the number of offspring produced, it might be necessary to use a reinsertion strategy to control the way in which the offspring enter the

population. A number of reinsertion strategies are available and are dependant on the type of selection algorithm used. For example, uniform insertion or fitness - based insertion.

The effectiveness of a genetic algorithm depends very much on the choice of certain parameters. The following is a short description of some of the important parameters of genetic algorithms.

Crossover probability is a measure of how often crossover will be performed. If there is no crossover, offspring are exact copies of parents. If there is crossover, offspring are made from parts of both parents' chromosomes. If the crossover probability is 100%, then all offspring are made by crossover. If it is 0%, a whole new generation is made from exact copies of chromosomes from the old population. Crossover is done in the hope that the new chromosomes will contain good parts of old chromosomes and therefore the new chromosomes will be better. However, it is good to leave some part of the old population to survive to the next generation [B43].

Mutation probability is how often parts of a chromosome will be mutated. If there is no mutation, offspring are generated immediately after crossover (or directly copied) without any change. If mutation is performed, one or more parts of the chromosome are changed. If the mutation probability is 100%, the whole chromosome is changed, if it is 0%, nothing is changed. Mutation generally prevents the genetic algorithm from falling into local extremes. Mutation should not occur very often, because then the genetic algorithm will in fact change to a random search [B43].

The population size of a genetic algorithm is the number of chromosomes in a population in one generation. If there are too few chromosomes, the genetic algorithm has few possibilities to perform crossover and only a small part of search space is explored. On the other hand, if there are too many chromosomes, the genetic algorithm slows down. Research shows that after some limit (which depends mainly on the encoding and the problem) it is not useful to use very large populations because it does not solve the problem faster than moderate sized populations [B43].

Constraints in genetic optimisation can be handled in two ways. The first approach is to ignore solutions that are not feasible and to replace them with new individuals that are feasible. The second approach is to make use of a suitable penalty function that penalises the

fitness of individuals that e not feasible. This in effect simplifies a constraint optimisation problem to an unconstraint optimisation problem.

Some basic recommendations for tuning parameters of genetic algorithms do exist. These recommendations are very general as there is no specific theory available that can be used to tune genetic algorithm parameters for all problems. The crossover rate of genetic algorithms should generally be high, about 80% - 95%. (However, some results show that for some problems a crossover rate of about 60% is the best). On the other hand, the mutation rate should be very low. The best rates have been found to be about 0.5% - 1% [B43].

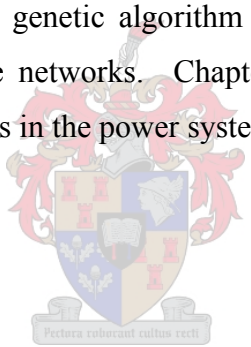
It has been found that very big population sizes do not usually improve the performance of genetic algorithms (in terms of speed of solution). Good population sizes are about 20 - 30, however sometimes sizes of 50 - 100 are reported as the best. Some research also shows that the best population size depends on the size of the encoded string (chromosomes) [B43]. For example, in chromosomes with 32 bits, the population size should be higher than for chromosomes with 16 bits.

Basic Roulette Wheel Selection can be used, but sometimes Rank Selection can be better. Elitism should be used if another method for saving the best solution is not used. Steady State Selection could also be attempted. The type of encoding depends on the problem and also on the size of the problem. Genetic operators depend on the chosen encoding and on the problem.

This concludes a very brief description of a simple genetic algorithm as used in the EVR placement program developed in this thesis. As can be seen the genetic algorithm is much simpler in principle than the smooth optimisation algorithms. This simplicity makes the genetic algorithm both practical and robust.

2.6 CONCLUSION

This chapter discussed the optimisation theory that is necessary to solve a wide variety of problems encountered in the field of power system studies. Once such an application has been identified, the first step is to describe the problem in such a way that it fits one of the standard optimisation problem structures. Standard optimisation techniques can then be used to solve these problems. Based on the characteristics of the different problems, the most effective solution method must be selected and then applied to solve the problems. In smooth optimisation, SQP is currently one of the most effective methods to solve general smooth non-linear optimisation problems. The method sequentially approximates the problem as a linear constraint quadratic problem and manages the inequality constraints with an active set strategy. Non smooth optimisation problems cannot be solved by using gradient methods such as SQP. Instead, methods that only use function evaluations are used, such as the genetic algorithm and the tabu search. Because of its simplicity as well as good results obtained with similar problems, the genetic algorithm has been used to solve the optimal placement of EVRs on low voltage networks. Chapter 3 will discuss in more detail the application of optimisation techniques in the power system environment.



CHAPTER 3 REVIEW OF OPTIMAL POWER FLOW

3.1 OVERVIEW

The research in this thesis has been done within the framework of finding new applications for optimisation algorithms in power system studies with the long term goal of automating tasks in the planning and operation of electrical power networks. Isolated work is being done by many researchers to automate certain tasks and breakthroughs are made regularly. With more and more applications being realised, there is a need to integrate some of these optimisation routines to solve larger and more complex problems. Thus far, optimisation theory has found application in power system simulations in limited areas aimed mainly at generation and interconnected high voltage transmission networks with very little work being done on the medium voltage (132 kV-3.3 kV) and low voltage (400/230 V) side. This chapter reviews the existing applications of optimisation methods in power system studies.

Based on the structure of the specific optimisation problem, many problems can be solved by the optimisation methods described in Chapter 2. The main consideration in selecting an optimisation method is the type of variables involved. If the variables are continuous and differentiable, the Newton type algorithms can be used. If the variables are piecewise continuous, the problem can be broken down into separate continuous problems and the same Newton type methods can be used. These types of algorithms are also very powerful for solving constraint problems, mainly by the use of the Lagrangian. Problems that are solved with these types of methods include the optimal power flow problem where generation dispatch is used to minimise generation cost and/or network technical losses. Other control variables include transformer tap changing and reactive power device outputs.

Another problem solved by optimisation methods is the shunt capacitor placement problem. Shunt capacitors are commonly used on transmission, distribution and reticulation networks to regulate voltage, reactive power flow, power factor and to minimise losses. The aim of optimal capacitor placement is to minimise cost while regulating all nodal voltages within certain limits. The total cost is made up of the annualised cost of the capacitor installations minus the savings due to the reduction in technical losses. The capacitor placement problem is solved as a combinatorial problem by one of a number of discrete optimisation methods like the genetic algorithm, tabu search, reactive tabu search and simulated annealing [B14],[B15], [B16].

3.2 THE ORDINARY POWER FLOW PROBLEM

The ordinary power flow is the problem of solving all the complex nodal voltages of a power network, given the complex loads and power generated at each node. Normally, at least one reference node is supplied where the complex voltage is known. Because of the non-linear characteristic of the nodal power flow equations, an iterative method is needed to solve the power flow equations. The ordinary power flow is solved efficiently by a number of available algorithms. Today, the Newton-Raphson method is preferred due to its quadratic convergence and other advantageous properties [B28]. Many textbooks cover the development of these algorithms [B22], [B27] and will not be discussed in detail here. A variety of software packages are available that can efficiently solve networks of practically any size. The following is a short summary of the Newton-Raphson method applied to solving a power system load flow.

The non-linear power flow equations are a summation of complex power at each node of the power system. To solve these equations with the Newton-Raphson method, it is necessary to separate these equations into equations that only contain real quantities. These nodal equations can be stated as follows:

$$P_k = V_k \sum_{n=1}^{N} Y_{kn} V_n \cos(\delta_k - \delta_n - \theta_{kn})$$

$$Q_k = V_k \sum_{n=1}^{N} Y_{kn} V_n \sin(\delta_k - \delta_n - \theta_{kn}) \qquad (3.1)$$

$$k = 1,2,...,N$$

where P_k and Q_k are the active and reactive power injections at node k. Y_{kn} and θ_{kn} refer to the magnitude and angle of the admittance between node k and n.

The non-linear equations of (3.1) are then linearised around the operating point by forming the total differentials as given by (3.2). The matrix containing the sensitivity information is called the Jacobian matrix.

$$\begin{bmatrix} \Delta P \\ \Delta Q \end{bmatrix} = \left[\begin{array}{cc|cc} \frac{\partial P_1}{\partial \delta_1} & \frac{\partial P_1}{\partial \delta_N} & \frac{\partial P_1}{\partial V_1} & \frac{\partial P_1}{\partial V_N} \\ \frac{\partial P_N}{\partial \delta_1} & \frac{\partial P_N}{\partial \delta_N} & \frac{\partial P_N}{\partial V_1} & \frac{\partial P_N}{\partial V_N} \\ \hline \frac{\partial Q_1}{\partial \delta_1} & \frac{\partial Q_1}{\partial \delta_N} & \frac{\partial Q_1}{\partial V_1} & \frac{\partial Q_1}{\partial V_N} \\ \frac{\partial Q_N}{\partial \delta_1} & \frac{\partial Q_N}{\partial \delta_N} & \frac{\partial Q_N}{\partial V_1} & \frac{\partial Q_N}{\partial V_N} \end{array}\right] \bullet \begin{bmatrix} \Delta \delta \\ \Delta V \end{bmatrix} \tag{3.2}$$

The entries of the Jacobian matrix are calculated as given by (3.3) and (3.4). Equation (3.3) gives the formulae for the off diagonal entries and (3.4) gives the formulae for the diagonal entries.

$$n \neq k$$

$$\begin{aligned} J1_{kn} &= \frac{\partial P_k}{\partial \delta_n} = V_k Y_{kn} \sin(\delta_k - \delta_n - \theta_{kn}) \\ J2_{kn} &= \frac{\partial P_k}{\partial V_n} = V_k Y_{kn} \cos(\delta_k - \delta_n - \theta_{kn}) \\ J3_{kn} &= \frac{\partial Q_k}{\partial \delta_n} = V_k Y_{kn} \cos(\delta_k - \delta_n - \theta_{kn}) \\ J4_{kn} &= \frac{\partial Q_k}{\partial V_n} = V_k Y_{kn} \sin(\delta_k - \delta_n - \theta_{kn}) \end{aligned} \tag{3.3}$$

$$n = k$$

$$J1_{kk} = \frac{\partial P_k}{\partial \delta_k} = -V_k \sum_{\substack{n=1 \\ n \neq k}}^{N} Y_{kn} V_n \sin(\delta_k - \delta_n - \theta_{kn})$$

$$J2_{kk} = \frac{\partial P_k}{\partial V_k} = V_k Y_{kk} \cos\theta_{kk} + \sum_{n=1}^{N} Y_{kn} V_n \cos(\delta_k - \delta_n - \theta_{kn}) \tag{3.4}$$

$$J3_{kk} = \frac{\partial Q_k}{\partial \delta_k} = V_k \sum_{\substack{n=1 \\ n \neq k}}^{N} Y_{kn} V_n \cos(\delta_k - \delta_n - \theta_{kn})$$

$$J4_{kk} = \frac{\partial Q_k}{\partial V_k} = -V_k Y_{kk} \sin\theta_{kk} + \sum_{n=1}^{N} Y_{kn} V_n \sin(\delta_k - \delta_n - \theta_{kn})$$

$$k, n = 2,3,\ldots,N$$

The non-linear nodal equations are then linearised and can be solved iteratively. Although the ordinary power flow is not an optimisation method it does have a strong resembles to the Newton type optimisation algorithms. It will be shown in the next section how the ordinary power flow problem can be integrated into an intelligent optimisation algorithm capable of finding the optimal operating point of a large power network.

3.3 GENERATION DISPATCH

The application of optimisation theory in power systems dates back to the 1950's when the need was identified to minimise the operating cost of generating and supplying electricity to a given load. Since then, major contributions in the development of algorithms and methods have been made by Dommel [B4], Squires [B5], Carpentier [B6] and Tinney [B7]. Today, optimal power flow algorithms are available either as stand alone programs or as part of integrated power system simulation packages for example PowerWorld and PowerFactory. The rest of this section discusses the structure of the generation dispatch problem as an optimisation program as well as some general strategies on how to solve it.

The optimal power flow problem consists of three parts:

- The objective function
- The power flow equations, which are implemented as equality constraints
- The limiting constraints implemented as inequality constraints

The objective function for minimising the total generation cost and minimising technical losses are given by equations (3.5) and (3.6) [B3].

$$\min \sum_{i=1}^{N} F_{\cos t_i}(P_i) \tag{3.5}$$

with N is the number of generation nodes

$$\min \sum_{i=1}^{N} F_{Loss_i} \tag{3.6}$$

$$\text{with } F_{Loss_i} = P_{km} + P_{mk} \text{ and}$$

branch i is connected between nodes k and m

The power flow equations are implemented as equality constraints, as they describe the balance of power in the network and have to be satisfied at all times. The limiting constraints are normally implemented as inequality constraints. Equations (3.7) to (3.18) show these inequality constraints [B3]:

Limits on power of a PV node:

$$P_{low_i} \le P_{PV_i} \le P_{high_i} \tag{3.7}$$

Limits on voltage of a PQ or PV node:

$$|V|_{low_i} \leq |V|_i \leq |V|_{high_i} \tag{3.8}$$

Limits on tap position of a transformer:

$$t_{low_i} \leq t_i \leq t_{high_i} \tag{3.9}$$

Limits on phase shift angle of a transformer:

$$\theta_{low_i} \leq \theta_i \leq \theta_{high_i} \tag{3.10}$$

Limits on shunt capacitances or reactances:

$$S_{low_i} \leq S_i \leq S_{high_i} \tag{3.11}$$

Limits on reactive power generation of a PV node:

$$Q_{low_i} \leq Q_{PV_i} \leq Q_{high_i} \tag{3.12}$$

Upper limit on active power flow in transmission lines and transformers:

$$P_{ij} \leq P_{high_{ij}} \tag{3.13}$$

Upper limit on total (MVA) power flow in transmission lines and transformers:

$$P_{ij}^2 + Q_{ij}^2 = S_{high_{ij}}^2 \tag{3.14}$$

Upper limits on current magnitude in transmission lines and transformers:

$$|I|_{ij} \leq |I|_{high_{ij}} \tag{3.15}$$

Limits on voltage angles between nodes:

$$\Theta_{low_{ij}} \leq \Theta_i - \Theta_j \leq \Theta_{high_{ij}} \tag{3.16}$$

Limits on area active power flows:

$$P_{low_{area_a}} \leq \sum_{a\ to\ b} P_{ab} \leq P_{high_{area_a}} \tag{3.17}$$

Limits on area MVA flows:

$$S_{low_{area_a}} \leq \sum_{a\ to\ b} \left(P_{ab}^2 + Q_{ab}^2\right) \leq S_{high_{area_a}}^2 \tag{3.18}$$

In summary, the optimal power flow must satisfy the power flow equations as well as the inequality constraints (3.7) - (3.18), while minimising some objective function, for example (3.5) or (3.6).

To simplify the understanding of the optimisation problem, the variables are divided into four categories:

- Demand variables
- Control variables
- State variables
- Output variables

The demand variables represent the variables that, under normal conditions, have constant values. Typical demand variables are active and reactive power at load nodes and equipment ratings. In general, these are all the variables that could be control variables but for some reason are not allowed to change. For example, the voltage magnitude of a PV node where the voltage is not allowed to move because of an operational or other reason. The control variables are the variables that represent the physical quantities that are allowed to be modified to satisfy the generation – demand balance under given constraints. Typical control variables are:

- Active power of a PV node
- Reactive power generation of a PV node
- Tap position of a transformer
- Shunt capacitance or reactance
- Real and imaginary part of a tap position (applicable for transformers with phase shift)

The state variables are the variables that, as a set, describe the complete state of the power network. The state variables in the optimal power flow problem are:

- Real part of complex voltages at all nodes
- Imaginary part of complex voltages at all nodes

All other variables are classed as output variables. They must be expressed as functions of the control and state variables. Typical output variables are:

- Voltage magnitude at PQ and PV nodes
- Voltage angle at PQ and PV nodes
- Tap magnitude and phase shift of phase shift transformers
- Power flow in branches
- Reactive generation at PV nodes

Some of the variables mentioned above might be discrete. For example, the tap position of a transformer can only be an integer and the status of a shunt capacitor can only be in service or out of service, namely zero or full value. Although there are different ways of handling this complication, a widely used method is to treat them as continuous variables. When the optimisation is finished, these variables can be set to the nearest available value. Other constraints that have been considered in previous work on optimal power flow are system security [B8] and system transient stability [B10].

Algorithms that solve the optimal power flow problem can be divided into two distinct classes according to the programming approach followed. Generally, algorithms are classed as either Class A or Class B algorithms. The Class A algorithms make use of the fact that an ordinary power flow already gives the following sensitivity information:

- A set of complex nodal voltages
- The Jacobian matrix
- The incremental power flow

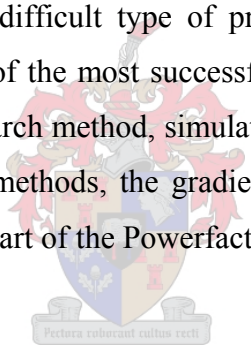
Another useful feature of the ordinary load flow that is taken advantage of is the fact that all the variables will already be within, or very close to, the constraints. Thus, the power flow is first solved and this solution is then taken as the starting point of the optimisation process. The optimisation process is then run. The output of the optimisation process is used to update the ordinary power flow, which is solved again. This process in continued until the optimal operating point is found. Standard algorithms can be used to solve the ordinary power flow

part of this iterative process. The optimisation part makes use of the Taylor approximation of the objective function, equality constraints and inequality constraints around the operating point obtained from the solution of the ordinary power flow. This leads to a linear or quadratic optimisation problem, which can be solved by standard techniques. From a programming point of view, the optimal power flow problem can be implemented using two separate software environments as long as the necessary variables can be transferred between the two. Thus, a Class A algorithm can be seen as an extension of the ordinary power flow solution.

In Class B type algorithms, the optimal power flow problem is solved by integrating the power flow equations into the optimisation problem and solving the problem as a whole. This is done by the direct application of the Kuhn–Tucker conditions for optimality to the optimisation problem and is effectively solved by application of the Newton type methods. An example of this application in PowerWorld is given in [B9]. Currently, the Class A type algorithms are preferred [B3].

3.4 OPTIMAL CAPACITOR PLACEMENT

The optimal placement of shunt capacitor banks on power networks has been the subject of much research. The reason for this is possibly because of the obvious financial benefit that can be obtained by reducing technical losses. Another possible reason is that capacitor banks, being shunt devices, are somewhat easier to handle within optimisation algorithms than series devices. The insertion of a series device means that an additional node must be inserted, which changes the data structure and power flow equations significantly. In general, the objective of the optimal shunt capacitor placement is to minimise the capital cost of installing the devices as well as to minimise technical losses dissipated on the network. The constraints of the capacitor placement optimisation problem are the upper and lower voltage limits on the network. The solution space is defined by the capacitor sizes available as well as the nodes available for placement. The capacitor banks may also be switchable. The capacitor placement problem, as explained, above is a constraint non-linear optimisation problem with discrete variables. This is a very difficult type of problem to solve and many different approaches have been tried. Some of the most successful methods are the gradient method, tabu search method, reactive tabu search method, simulated annealing, genetic algorithms and evolutionary algorithms. Of these methods, the gradient and the tabu search method have been implemented commercially as part of the Powerfactory software from DigSilent.



3.5 PSAT

3.5.1 Introduction

Power System Analysis Toolbox (PSAT) is a Matlab based program developed by the University of Stellenbosch that enables quick evaluation of power system compensators. This software indirectly uses optimisation techniques to solve the underlying mathematical equations governing compensator behaviour and is thus classed as an application of optimisation theory. It was decided to use this software as a starting point for the development of new optimisation applications. The development of PSAT was started by a previous student [B1] and completed as part of this thesis. This section examines in detail the software from an optimisation application point of view. See [B1] for more detail information on the mathematical formulation of the network and compensator models. Compensators in PSAT are modelled as one of the following types:

- Shunt devices e.g. shunt capacitor bank
- Series devices e.g. series capacitor
- Series-shunt devices e.g. unified power-flow controller
- In-line devices e.g. HVDC line

These topologies are illustrated in Figure 3-1.

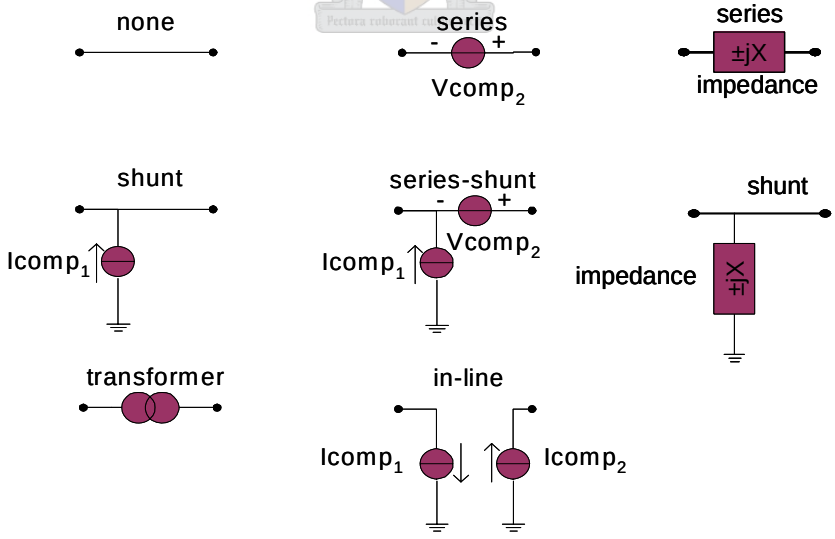


Figure 3-1: Compensator models

The circuit model for solving the power flow problem in the software is shown in Figure 3-2.

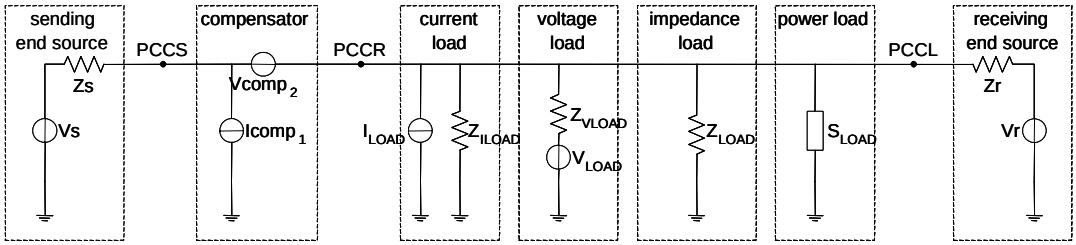


Figure 3-2: Circuit model for analysis of compensators

Five points of interest can be defined in this model:

- S: Sending end
- PCCS: Point of Common Coupling at the Sending end of the compensator
- PCCR: Point of Common Coupling at the Receiving end of the compensator
- PCCL: Point of Common Coupling for the loads
- R: Receiving end

This model caters for a combination of constant impedance, constant current and constant power loads, sending and receiving end voltage sources and any topology of the compensator. PSAT is based on the observation that in general, compensators are designed to regulate one or more of the following three properties in a power network:

1. The voltage level at PCCR ($|V_{PCCR}|$)
2. The phase angle at PCCR ($\theta_{V_{PCCR}}$)
3. The power factor at PCCS (pf_{PCCS})

From a design point of view, the aim is to set up and solve a network model so that the above three properties are as close as possible to certain set points. In practice, preference might be given to one or more of these three properties. To implement these preferences, value theory is used by introducing scaling factors as follows:

$$0 \le \lambda_{1,2,3} \le 1,\ \lambda_1 + \lambda_2 + \lambda_3 = 1 \tag{3.19}$$

Mathematically, this compensator design problem can be stated as follows:

$$\min_{\bar{x}} \left\{ \gamma_1 \left(\left| V_{PCCR}(\bar{x}) \right| - \left| V_{ref} \right| \right)^2 + \gamma_2 \left(\theta_{V_{PCCR}}(\bar{x}) - \theta_{V_{ref}} \right)^2 + \gamma_3 \left(pf_{PCCS}(\bar{x}) - pf_{ref} \right)^2 \right\} \quad (3.20)$$

This problem statement forms the objective function of the optimisation problem. At this stage, the following important observations can be made:

- The objective function is a scalar function. This is important since optimisation methods in general can only handle a scalar objective function.
- The objective function is non-linear. This is obvious since all three terms in the objective function are squared.
- The objective function is a quadratic function of the objective variables, but not necessarily a quadratic function of the state variables. The quadratic form is a very special non-linearity in optimisation theory and is easy to solve.
- Because of this non-linearity, the optimisation problem is classified as a non-linear optimisation problem.

The second part of the optimisation problem is called the constraint functions. In this specific problem, the constraints are manifested by the finite ratings of the physical compensator. In the PSAT software, these ratings are specified as follow:

- Separate positive and negative voltage regulation limits
- Separate positive and negative phase angle regulation limits
- Separate positive and negative real power set points
- Separate apparent power limits for positive and negative reactive power operation
- Separate positive and negative current ratings

The above compensator power limits are shown is Figure 3-3.

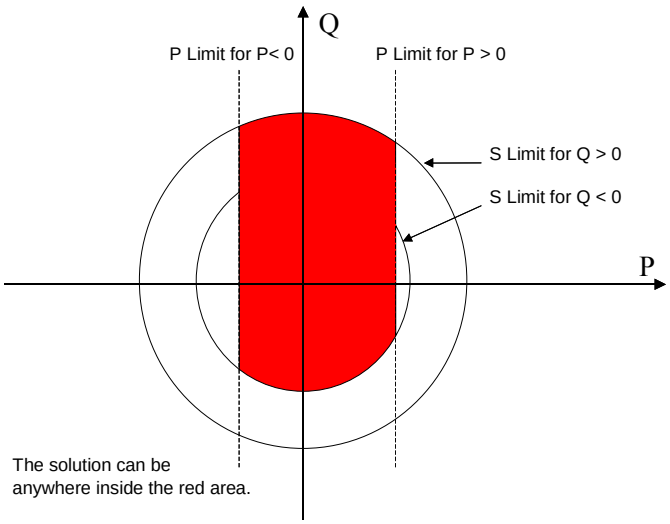


Figure 3-3: Compensator power limits

Because of these constraints, the optimisation problem is classified as a constraint non-linear optimisation problem. Mathematically, the problem can now be stated as follows:

$$\min_{\bar{x}} \left\{ \gamma_1 \left(\left|V_{PCCR}(\bar{x})\right| - \left|V_{ref}\right| \right)^2 + \gamma_2 \left(\theta_{V_{PCCR}}(\bar{x}) - \theta_{V_{ref}} \right)^2 + \gamma_3 \left(pf_{PCCS}(\bar{x}) - pf_{ref} \right)^2 \right\} \tag{3.21}$$

subject to

$$c_i(\bar{x}) = 0, \quad i \in E$$

$$c_i(\bar{x}) \geq 0, \quad i \in I$$

Here the constraints c_i are the voltage and phase angle regulation constraints as well as the power constraints of the compensator. The only variation in this problem is introduced by the compensator. The realisation that the behaviour of the compensator can be completely described by the compensator source and load side voltage phasors is used to formulate the mathematical equations. Since the problem is solved as an optimisation problem, it is convenient to define the four scalar variables given by (3.22). These quantities are the sending end voltage magnitude and phase angle and the receiving end voltage magnitude and phase angle.

$$\left|V_{PCCR}\right|, \ \theta_{V_{PCCR}}, \ \left|V_{PCCS}\right| \text{ and } \theta_{V_{PCCS}} \tag{3.22}$$

All other quantities in the network model can be calculated in terms of these four variables, called the state variables. This includes the objective variable pf_{PCCS}, which is the only variable in the objective function that is not a state variable. This means that the objective variables can be manipulated to minimise the objective function, subjected to the limits that the compensator ratings put on the state variables.

The configuration of the compensator can simplify the optimisation problem by reducing the number of state variables. For example, in the case of a shunt configuration, the sending and receiving end voltages is the same, which can be expressed as follows:

$$|V_{PCCR}| = |V_{PCCS}| \text{ and } \theta_{V_{PCCR}} = \theta_{V_{PCCS}} \tag{3.23}$$

Based on the compensator topology, five separate optimisation routines have been implemented.

- *Vbus_none* – Finds the operating point of a network with no compensator.
- *Vbus_series* - Finds the operating point of a network with a series configured compensators.
- *Vbus_shunt* - Finds the operating point of a network with a shunt configured compensator.
- *Vbus_angletransformer* - Finds the operating point of a network with an angle transformer as a compensator.
- *Vbus_transformer* - Finds the operating point of a network with a transformer as a compensator.

Solving the load flow equations as a constraint optimisation problem is done by the following steps:

- Define a set of states. States are the minimum number of variables that are sufficient to for solving the equations. In this case, the variables $|V_{PCCR}|$, $\theta_{V_{PCCR}}$, $|V_{PCCS}|$ and $\theta_{V_{PCCS}}$ are used as the states.
- Express the optimisation objective function in terms of the state variables.
- Express the equality and inequality constraint functions in terms of the state variables.
- Solve the above two sets of equations as a constraint optimisation problem.

Assignment of the states as above has the advantage that the states are all real numbers, which makes the application of standard optimisation techniques possible. These states are easily transformed back to complex quantities to form the voltage, current and power phasors of the solved network.

3.5.2 Solving the power flow problem

In order to decide on which optimisation technique to use to solve the power flow problem it is necessary to evaluate the structure of both the objective and constraint functions.

Convexity is a very useful feature to have in solving an optimisation problem and it is therefore worth checking both the objective and constraint functions for convexity. If the convexity of the objective function and the constraint functions can be proven, the optimisation problem becomes a convex programming problem. Mathematically, this means that a local minimum is necessarily also a global minimum and therefore a sufficient condition for the correct solution to the problem. This type of problem is solved by applying the Kuhn-Tucker conditions for optimality. The Kuhn-Tucker theorem is based on constructing the Lagrangian function, $L(X,\lambda)$ and then solving a set of equations resulting from (3.24).

$$L(x,\lambda) = f(x) - \sum_i \lambda_i c_i(x) \tag{3.24}$$

A problem with equality constraints that are non-linear is not a convex programming problem. It is, therefore, necessary to investigate the equality constraints of the power flow problem for linearity.

From (3.21) the objective function is:

$$\min_{\bar{x}} \left\{ \gamma_1 \left(\left|V_{PCCR}(\bar{x})\right| - \left|V_{ref}\right| \right)^2 + \gamma_2 \left(\theta_{V_{PCCR}}(\bar{x}) - \theta_{V_{ref}} \right)^2 + \gamma_3 \left(pf_{PCCS}(\bar{x}) - pf_{ref} \right)^2 \right\} \tag{3.25}$$

From the objective function in (3.25), it can be seen that the first two terms are quadratic functions of the state variables $\left|V_{PCCR}\right|$ and $\theta_{V_{PCCR}}$. The third term can be rewritten in terms of the state variables as:

$$\gamma_3\left(\cos\left(\theta_{VPCCS}-\frac{\sin\theta_{PCCS}}{V_s-|V_{PCCS}|\cos\theta_{PCCS}}\right)-pf_{ref}\right)^2 \tag{3.26}$$

Thus it can be seen that because of the third term in the objective function, $\gamma_3\left(pf_{PCCS}(\bar{x})pf_{ref}\right)^2$, the objective function is not an obvious convex function of the state variables. It is worth noting that if λ_3 is equal to zero, the objective function definitely becomes convex. The constraints must also be evaluated for convexity. Table 1 to Table 4 show the inequality and equality constraints.

Table 1: Voltage and Angle Regulation constraints

$\frac{-(V_{PCCR}-V_{PCCS})}{V_{PCCS}}+\frac{VR_{rating}(1)}{100}\le 0$	Voltage regulation upper limit	(3.27)
$\frac{(V_{PCCR}-V_{PCCS})}{V_{PCCS}}-\frac{VR_{rating}(1)}{100}\le 0$	Voltage regulation lower limit	(3.28)
$-(\theta_{PCCR}-\theta_{PCCS})+PAR_{rating}(1)\le 0$	Angle regulation upper limit	(3.29)
$(\theta_{PCCR}-\theta_{PCCS})-PAR_{rating}(1)\le 0$	Angle regulation lower limit	(3.30)

Table 2: Apparent power constraints

$-\lvert V_{PCCR}-V_{PCCS}\rvert * sign(\mathrm{Re}(S_{PCCS}))+S_{Rating}(1)\le 0$	Apparent power upper limit	(3.31)
$\lvert V_{PCCR}-V_{PCCS}\rvert * sign(\mathrm{Re}(S_{PCCS}))-S_{Rating}(2)\le 0$	Apparent power lower limit	(3.32)

with

$S_{PCCS}=\lvert V_{PCCR}\rvert(\cos\theta_{PCCR}+j\sin\theta_{PCCR})(c_a+jc_b)+\lvert V_{PCCR}\rvert^2(c_c+jc_d)$	(3.33)
$S_{PCCR}=\lvert V_{PCCR}\rvert(\cos\theta_{PCCR}+j\sin\theta_{PCCR})(c_e+jc_f)+\lvert V_{PCCR}\rvert^2(c_g+jc_h)$	(3.34)
$where \quad c_a=\mathrm{Re}\left(\frac{V_S^*}{Z_S^*}\right) \quad c_b=\mathrm{Im}\left(\frac{V_S^*}{Z_S^*}\right) \quad c_c=\mathrm{Re}\left(-\frac{1}{Z_S^*}\right) \quad c_d=\mathrm{Im}\left(-\frac{1}{Z_S^*}\right)$	(3.35)
$and \quad c_e=\mathrm{Re}\left(-\frac{V_R^*}{Z_R^*}\right) \quad c_f=\mathrm{Im}\left(-\frac{V_R^*}{Z_R^*}\right) \quad c_g=\mathrm{Re}\left(\frac{1}{Z_R^*}\right) \quad c_h=\mathrm{Im}\left(\frac{1}{Z_R^*}\right)$	(3.36)

Table 3: Current constraints

$-\lvert I_{PCCR} - I_{PCCS}\rvert * sign(\mathrm{Re}(I_{PCCS})) + I_{Rating}(1) \le 0$	Current upper limit	(3.37)
$\lvert I_{PCCR} - I_{PCCS}\rvert * sign(\mathrm{Re}(I_{PCCS})) - I_{Rating}(2) \le 0$	Current lower limit	(3.38)

with

$I_{PCCS} = (c_a + jc_b) - \lvert V_{PCCR}\rvert(\cos\theta_{PCCR} + j\sin\theta_{PCCR})(c_c + jc_d)$	(3.39)
$I_{PCCR} = \lvert V_{PCCR}\rvert(\cos\theta_{PCCR} + j\sin\theta_{PCCR})(c_g + jc_h) + (c_e + jc_f) + \ldots$ $\ldots + \frac{S_{Load}}{\lvert V_{PCCR}\rvert(\cos\theta_{PCCR} + j\sin\theta_{PCCR})}$	(3.40)

Table 4: Active power constraints

$\lvert V_{PCCR}\rvert^2(c_c - c_g) + \lvert V_{PCCR}\rvert\cos\theta_{PCCR}(c_a - c_e) - \lvert V_{PCCR}\rvert\sin\theta_{PCCR}(c_b - c_f)\cdots$ $\cdots - P_{load}' + P_{rating} = 0$	(3.41)

As can be seen from Table 1 to Table 4, all the inequality constraints, except the phase angle regulation constraints, are non-linear. Moreover, the apparent power and current inequality constraints are complex functions of the state variables. It can also be seen that the equality constraint of the active power is non-linear.

As stated earlier, any problem with non-linear equality constraints is not a convex programming problem. It is therefore concluded that the power flow problem cannot be solved as a convex programming problem. The consequence of this is that solving the problem using a gradient method such as SQP might only produce a local optimal solution. It is therefore important to use a starting point for the optimisation search that is sufficiently close to the global optimal solution.

The only equality constraints in the power flow problem arise from the way in which the active power of the compensator is specified. The active power is specified as a set point. If the active power is specified as an active power limit, this equality constraint will be transformed into an inequality constraint. Since the problem is not convex, it is worth considering other techniques to solve this non-linear problem. Because of the good linear programming techniques available, it will be advantageous to linearise the non-linearities around the operating point. This technique is called linearization and is very powerful. It can

handle objective and constraint functions of any shape, is quick and can handle large numbers of variables [B24]. Thus, it might thus be possible to solve the problem at hand by using a linearization technique. When the problem is highly non-linear, linearization techniques may not always result in convergence. In such a case, it may be necessary to resort to a feasible direction method or a penalty function method. The problem with the feasible direction method is that it can be difficult to locate a feasible starting point. In PSAT, a feasible starting point is calculated first by solving the network without any compensator installed. Therefore, it might be possible to solve the problem with a feasible direction technique. Another method available is the penalty function method. This method attempts to reformulate the non-linear programming problem as an unconstraint problem that does not require a feasible starting point. This method is very powerful and is recommended when highly non-linear problems are to be solved [B24]. From using the software, however, it has been proven that using an SQP technique works quite well and no problems are experienced with infeasible solutions. The reason for this is possibly the fact that a feasible starting point is always guaranteed.

3.5.3 Implementation in Matlab

This section describes how the PSAT optimisation program has been implemented in Matlab. The basic Matlab package only contains very basic optimisation functionality in the form of the following functions:

- *Fminbnd* – This function minimises a function in one variable on a fixed interval
- *Fminsearch* - This function can handle functions in several variables
- *Fzero* - This function can find a zero of a function in one variable
- *Lsqnonneg* - This function solves a linear least squares problem with non negative constraints

These are very basic functions that can solve reasonably simple optimisation problems. For more complex problem solving, an Optimisation Toolbox by MathWorks is available. The Optimization Toolbox solves optimisation problems of the following types [B29]:

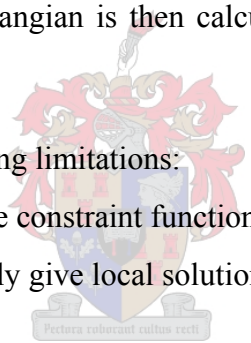
- Unconstrained non-linear minimisation
- Constrained non-linear minimisation, including goal attainment problems
- Minimax problems, and semi-infinite minimisation problems
- Quadratic and linear programming
- Non-linear least squares and curve-fitting

- Non-linear system of equation solving
- Constrained linear least squares
- Sparse and structured large-scale problems

The PSAT power flow problem as given by (3.25) has the following attributes:

- The object function is non-linear
- The inequality constraints are non-linear
- The equality constraints are non-linear

This type of problem is solved in the Optimisation Toolbox by the *fmincon* function. The following is a short description of this function for application to medium scale problems. The *fmincon* function uses an SQP method to minimise a generally non-linear constraint function. In this method, a Quadratic Programming (QP) sub problem is solved at each iteration. The QP sub problem is obtained by linearisation of the non-linear constraint functions. The Hessian of the Lagrangian is then calculated and updated at each iteration, using the BFGS formula.



The *fmincon* function has the following limitations:

- The objective function and the constraint functions must be continuous.
- The *fmincon* function may only give local solutions. A good choice of starting point is essential.
- The objective function and constraint functions must be real-valued, i.e. they cannot return complex values.

In the PSAT optimisation problem the above limitations are easily dealt with and do not provide any problems.

3.5.4 Example

To show how PSAT works, a simple example is analysed step by step. The purpose of the example is to verify results from PSAT as well as to give insight into the constraint optimisation techniques used. The following three analyses will be done:

- Hand calculation
- Graphical calculation
- Solution by PSAT

The circuit that will be analysed is shown in Figure 3-4.

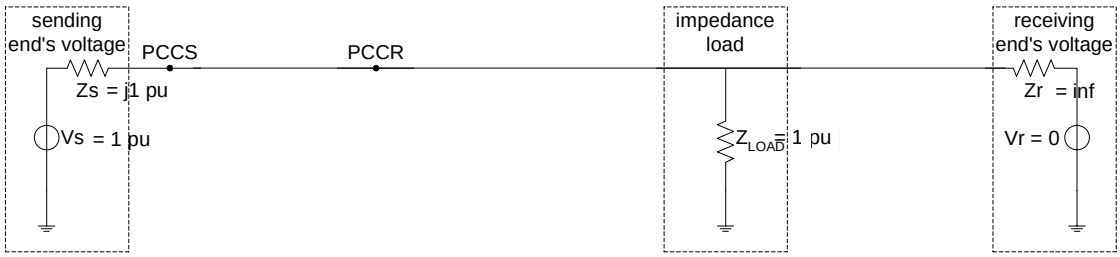


Figure 3-4: Simple circuit analysis

As can be seen from Figure 3-4, no compensator is present. For the purpose of this example, only $|Vpccr|$ will be optimised. Therefore $[\gamma_1 \gamma_2 \gamma_3]$ is set to [1 0 0]. That means that from (3.21) the objective function reduces to:

$$\min_{\bar{x}} \left\{ \left(\left| V_{PCCR}(\bar{x}) \right| - \left| V_{ref} \right| \right)^2 \right\} \tag{3.42}$$

This function is shown for values of V_{PCCR} in the neighborhood of the solution in Figure 3-5.

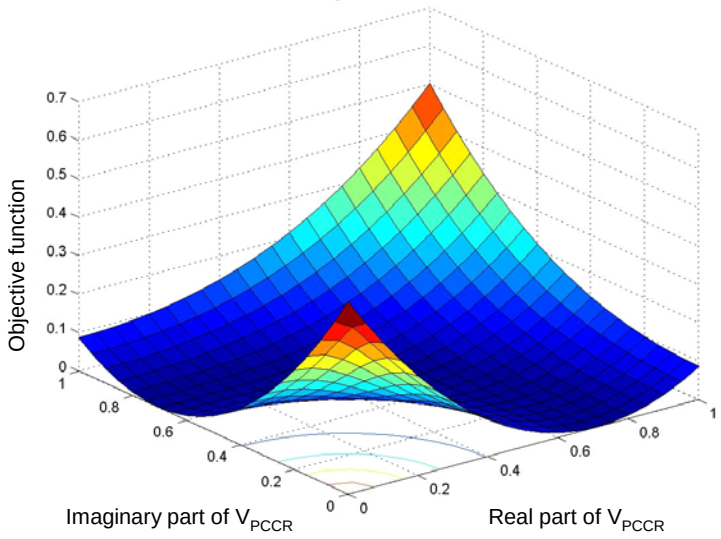


Figure 3-5: Plot of objective function

In terms of the PSAT model, the variables given in Table 5 have been set to zero or infinity.

Table 5: PSAT variable settings

$V_r = 0$	$I_L = 0$
$V_r = \infty$	$V_L = 0$
$S_{LOAD} = 0$	$S_L = 0$
$Z_{IL} = \infty$	$Z_L = 0$

In this circuit there is no compensator present, which means that $|Vpccr|$ cannot be altered. By simple calculation (voltage division), the voltage V_{PCCR} is be found to be 0.5 + j0.5 p.u. When solving the circuit according to the PSAT methodology of setting up the power constraint function as explained in 3.5.2 with (3.43),

$$S_{PCCR} = S_{PCCS} \tag{3.43}$$

the equation for complex power simplifies to:

$$j|V_{PCCR}|^2 - \overline{V}_{PCCR} + |V_{PCCR}|^2 = 0 \tag{3.44}$$

Function (3.44) is plotted in the complex plain in the region of the non-trivial solution, 0.5 + j0.5 p.u. in Figure 3-6. Only the magnitude of the function has been plotted. This is adequate to find the solution to the equation.

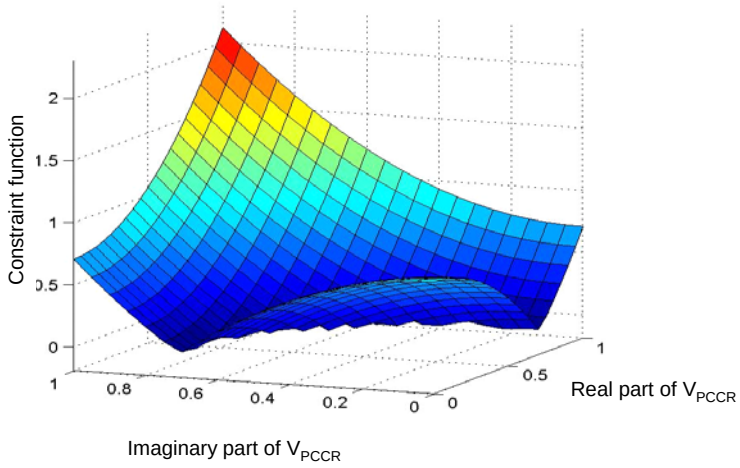


Figure 3-6: Constraint surface

In this case (where there is no compensator present), the solution of the constraint optimisation problem is found to be where the optimisation function and the constraint functions intersect. This intersection is shown in Figure 3-7.

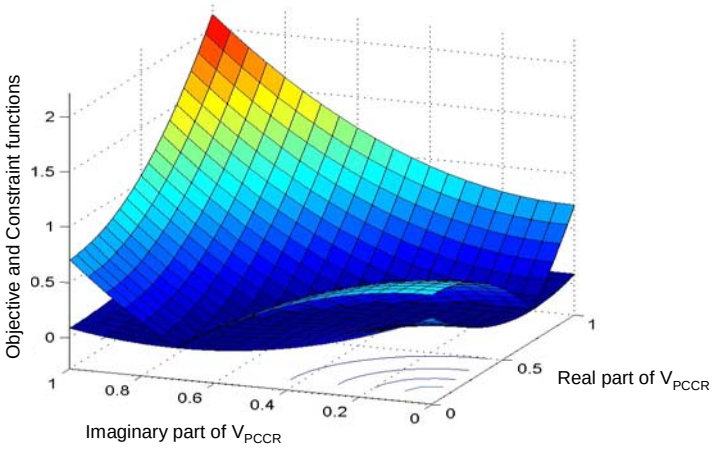


Figure 3-7: Objective and constraint surfaces

The intersection occurs at the point 0.5 + j0.5 p.u., as was expected. This result can be checked against the result from PSAT as shown in Figure 3-8.

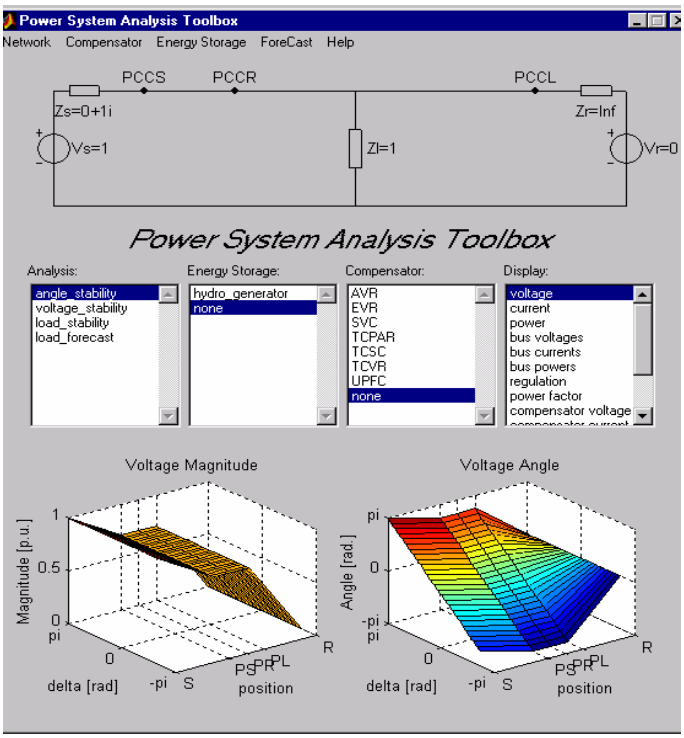


Figure 3-8: Plot of the PSAT circuit and simulation result

As can be seen, the result is the same as that obtained from both the hand calculation and the graphical solution.

This section has covered a wide variety of principles, which is essential for the understanding of the optimisation software used to solve compensator problems as well as understanding general smooth optimisation methods used in power system studies. It is seen that the given

problem is a constraint non-linear problem that can be efficiently solved with an SQP technique such as that implemented in the Optimization Toolbox.

3.6 SOFTWARE

In order to develop optimisation techniques to solve power flow problems it is necessary to decide on a suitable software environment to support the necessary program development. Preferably, the environment must be capable of solving a large set of non-linear power flow equations as well as supporting optimisation algorithms suitable for solving functions that are non-linear and contain discrete variables and other discontinuities. Alternatively, separate software environments can be used to solve the power flow equations and the optimisation algorithms. In such a case it is obviously necessary to have a suitable link between the two software environments to transfer variables and parameters. With the above-mentioned issues in mind, a thorough survey was done to decide on a suitable software environment for the work presented in this thesis. The survey was done with the knowledge of the software used in previously developed programs on this research, as well as what software is supported by Eskom Distribution and by the University of Stellenbosch. The following is a review of the available software packages that were evaluated.

DigSilent PowerFactory is an integrated power system simulation program capable of doing most types of power system related simulations. In 2002, Eskom Distribution acquired a corporate license for DigSilent Power Factory. This effectively replaced PSSE (Power System Simulation for Engineers) that was used up to that point. In August 2003, PowerFactory version 13.1 was rolled out in Distribution. This was the first version of PowerFactory that had some OPF capabilities. Table 6 gives a summary of the functionality of PowerFactory version 13.1.

Table 6: PowerFactory optimisation functionality

Objective function	Loss minimisation
	Production cost minimisation
Controls	Gen active power dispatch
	Gen reactive power dispatch
	Transformer tap positions
Constraints	Branch limits
	Bus voltage limits

DigSilent does have a built-in programming language called DigSilent Programming Language that is capable of developing optimisation algorithms that can interact with the load

flow and other functions. DigSilent is also working on incorporating a link between PowerFactory and Matlab [B42], which will further enhance the capabilities of PowerFactory to handle advanced optimisation procedures. Typically, the two packages can be used to develop Class A optimisation algorithms using PowerFactory to solve the network equations and the MathWorks Optimization Toolbox to do the optimisation part.

Matpower is a Matlab based power system simulation package that can solve ordinary, as well as optimal power flows. The program is freeware and is designed as a tool for research. The program makes use of the Mathworks Optimisation Toolbox and consists of a collection of Matlab M-files that make it easy to modify or incorporate into other Matlab programs. Because of these features, Matpower is a very good basis to develop further applications in power systems such as optimal compensator design. The software comes with a relatively complete manual [B30] and is very easy to use. Another feature of the software is that the data structure is based on the PTI (Power Technologies, Inc) data structure. The only disadvantage of Matpower is that it is based on an earlier version of the Optimization Toolbox. Since then the *constr* function of the Optimization Toolbox has been replaced by the *fmincon* function. These two functions have major differences in the structure of the optimisation constraints and are not compatible at all. Because of this problem, the optimal power flow facilities do not work with the latest version of the Mathworks Optimization Toolbox. The author of MatPower is, however, working on the necessary upgrades [B41].

PSSE has a completely integrated optimal power flow solver called *PSSE OPF*. This package can solve the optimal power flow problem with the general structure as given in Table 7.

Table 7: PSSE OPF optimisation functionality

Objective function	Loss minimisation
	Production cost minimisation
	Interface flows
Controls	Gen active power dispatch
	Gen reactive power dispatch
	Adjustable series and shunt reactive devices
Constraints	Branch flow limits
	Bus voltage limits
	Interface flow limits

PSSE OPF seems to be quite powerful with a number of more specific options available to the user as well as the flexibility to combine different objective and constraint functions. The package also comes with a special graphical user interface that makes programming easy.

PowerWorld is another load flow simulation program that solves optimal power flow. A limited bus version of the software is available as freeware. The advantage of PowerWorld is that it has an interactive graphical user interface. Namely, networks can be entered graphically and the results are shown graphically. Unique to this program is that power flows and other parameters can be changed and observed online. Results are shown by moving arrows and other interactive graphics. Another useful feature of the program is that the bus admittance matrix and Jacobian matrix are fully accessible.

PSAT is a Matlab program designed by the University of Stellenbosch specifically as a tool to understand and evaluate compensators to solve network problems. A very useful feature of the program is that results are displayed as three-dimensional graphs, which aids in the understanding of the operation of the compensators. The solver of the program is based on the *fmincon* function of the MathWorks Optimization Toolbox. The *fmincon* function is a function that optimises any multivariable constrained non-linear function. It can handle both non-linear equality and inequality constraints. The function uses state of the art algorithms based on the SQP technique. A general feature that is unique to PSAT is that it makes use of general optimisation techniques to solve power flow problems.

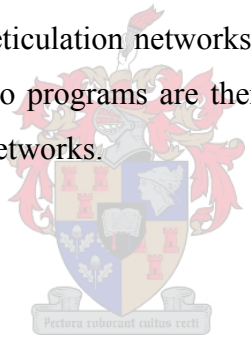
Matlab is a generally used package for engineering software development and is also used at the University of Stellenbosch. The advantage of Matlab is its handling of matrix calculations, flexibility and ease of use. Another advantage is its advanced graphing facilities and the numerous specialised toolboxes available. Two of these toolboxes specifically were used extensively in the software developed as part of this thesis. The two toolboxes used are the MatWorks Optimization Toolbox and the Genetic Algorithm Toolbox developed by the University of Sheffield [B31].

After evaluating the programs discussed, it was decided to use MatLab to handle the matrix computations necessary for solving power flow equations as well as the various optimisation toolboxes that already exist to implement different optimisation algorithms. Another advantage is that direct access to all code is possible and it is easy to integrate different functions, programs and toolboxes.

3.7 CONCLUSION

This chapter discussed a wide variety of subjects that serve as background for the rest of the thesis. It is seen that optimisation methods are used in a number of applications in power systems. Optimal generation dispatch and optimal capacitor placement are two of the well established applications of optimisation methods and are available in a number of commercial simulation programs. The third application covered in this chapter is that of optimal compensator design. The PSAT program is analysed in detail to show how it solves the compensator design problem as a constraint non-linear optimisation problem. Finally, all the available software packages are discussed in order to decided on a suitable environment for further development work on the subject of applying optimisation methods in compensator placement.

Chapter 4 describes two programs developed specifically for the placement of voltage regulators on different types of power networks. The one program places Step Voltage Regulators on sub transmission or reticulation networks and the other program places EVRs on low voltage networks. These two programs are then used in Chapter 5 to place voltage regulators on a number of practical networks.



CHAPTER 4 OPTIMAL PLACEMENT OF VOLTAGE REGULATORS

4.1 OVERVIEW

In Eskom Distribution one of the most common network problems is voltage regulation. On reticulation networks Step Voltage Regulators (SVR) and shunt capacitor banks are the two most commonly used methods to strengthen networks suffering from voltage regulation problems. SVRs and shunt capacitor banks can also be used as a combined solution on a particular network. On single phase, low voltage (230 V) networks, voltage can be regulated with an EVR. This is a relatively new device that is aimed at deep rural electrification applications.

This chapter focuses on the two programs that have been developed to optimally place SVRs and EVRs. The program developed to place SVRs is based on an SQP technique and the EVR placement program is based on a genetic algorithm. The former is described in Section 4.2 and the latter in Section 4.3. Chapter 5 presents a number of case studies to illustrate the use of both programs.

4.1.1 Types of voltage regulators

For the purpose of this thesis an SVR is modelled as a transformer with a certain boost capability. In practice, SVRs come in standard sizes and can be configured to be in either closed delta or open delta configuration. The closed delta configuration gives a maximum of 15% voltage boost and the open delta configuration gives a maximum of 10% voltage boost. The other consideration when placing an SVR is its current carrying capacity. [B34] is the Eskom standard for the application of SVRs.

The EVR is a single phase transformer fitted with a thyristor controlled tap changer that is used on single phase, low voltage networks to do automatic voltage regulation. The device was developed by the University of Stellenbosch and has been installed successfully on various Eskom networks. Typically, these devices are installed on electrification projects where low voltage problems are experienced. The motivation for developing the EVR is to stretch LV networks in electrification projects to make them more cost effective [B44].

4.1.2 Placement of Step Voltage Regulators

The placement of an SVR forms an integral part of the planning and management of these devices on Eskom reticulation networks. The process of installing a new SVR is shown in Figure 4-1. As can be seen the placement of an SVR is a very complex process.

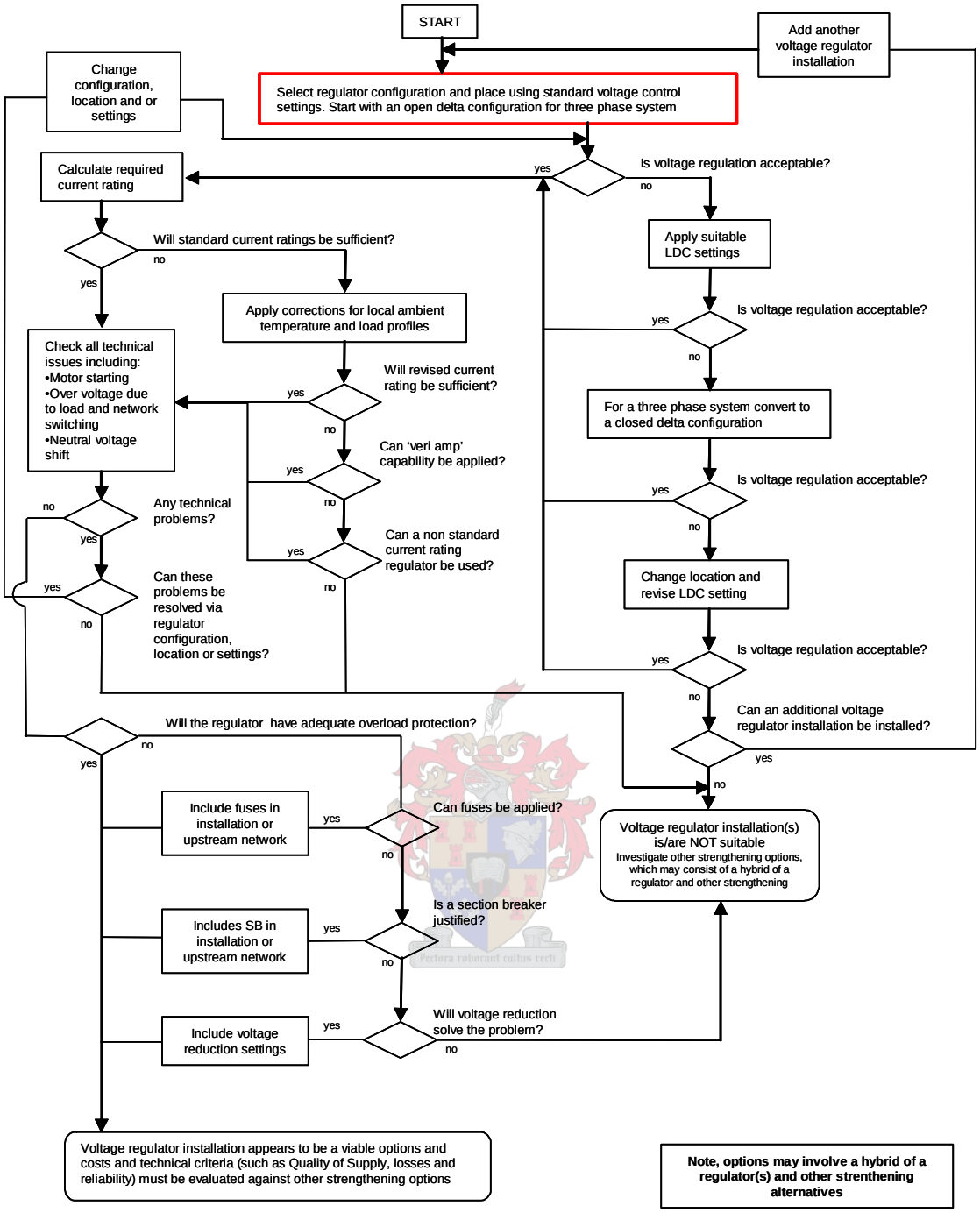


Figure 4-1: Process of installing a new SVR [B35]

Traditionally, the placement of SVRs is done by doing a large number of load flow studies while varying the placement, configuration and rating of the SVR to manually obtain the optimal position. This task is complicated by the non-linear nature of the power flow equations as well as network constraints. Because of these complications, it was decided to develop a program that can automatically find the optimal placement of a single SVR on a radial network. Optimality, as used in this program, can be defined as the optimal point of the user defined objective function. The objective function used for this thesis was simply to

maximise the end of line voltage. Voltages elsewhere on the network are regulated within limits by the constraints. The result is that, although mathematically the solution is optimal, it might not be practically optimal. To make the solution optimal from a practical point of view, the objective function must consider a more detailed SVR placement philosophy. Similar difficulty is experienced with the placement of multiple EVRs on low voltage networks. The following typical applications are addressed in this thesis:

- Placement of an SVR on a distribution or reticulation network. The development of a program to do SVR placement is covered in Section 4.2.
- Placement of multiple EVRs on a radial low voltage network. The development of a program to do EVR placement is covered in Section 4.3.

4.2 STEP VOLTAGE REGULATOR PLACEMENT PROGRAM

This placement program has been developed to simplify the task of placing a single Step Voltage Regulator (SVR) on a radial electrical network. This seemingly easy task is complicated by the following factors:

1. Non-linear nature of the power flow equations
2. Large number of loads
3. Mixture of load types

The program optimally places a single SVR on a balanced, three phase, radial distribution or reticulation network. For the purpose of the case studies presented in this thesis the objective function for the optimal placement is taken as to maximise the end of line voltage while satisfying all the nodal voltage limits. The program relies on the assumption that an SVR can be placed at an existing bus or somewhere on a branch between two existing busses. Since an SVR is a series device, two busses must be available to insert an SVR. In the first case where the SVR is placed at an existing bus, a single additional bus is inserted into the network. In this case the placement is fixed and only the SVR voltage ratio is taken as an optimisation variable. In the second case, where the SVR is inserted on a branch between two existing busses, two additional busses are inserted into the network. In this case, both the SVR voltage ratio as well as the SVR placement, are used as optimisation variables.

Network data is entered in the form of a Matlab M-file by entering the filename of the M-file containing the network data. The structure of network data has been adopted from MatPower and has been used throughout this program. The indexing into the network data is given in Addendum A.

The model used for the SVR is shown in Figure 4-2. This model considers the effect of variable voltage ratio on the SVR internal impedance.

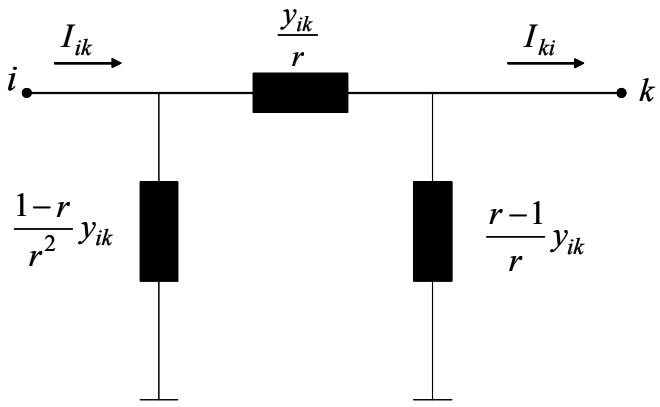


Figure 4-2: SVR model [B28]

The equations describing this model are as follows:

$$I_{ki} = y_{ik} V_k - \frac{y_{ik}}{r} V_i \tag{4.1}$$

$$I_{ik} = -\frac{y_{ik}}{r} V_k + \frac{y_{ik}}{r^2} V_i \tag{4.2}$$

The SVR placement problem has been approached as a combination of a discrete combinatorial problem and a smooth optimisation problem. The type of variables of the problem determines how the problem are divided into two sub problems. The two sub problems are solved in two separate parts of the program. The first of the two sub problems is a mixed combinatorial and a smooth problem and are implemented in part 1 of the program. The necessary parameters and variables are then transferred to part 2 of the program that is a strict smooth optimisation program. The control variables of the problem are the SVR placement and the SVR voltage ratio. The SVR placement variable is in essence a discrete variable as the network configuration changes when the SVR is moved from one branch to the next. This change in network configuration leads to a change in some of the equations describing the equality constraints enforced by the load flow equations. If the movement of a SVR is restricted to be along a certain branch, the placement variable can however be handled as a continuous one. The first approach is followed in part 1 of the program and the former in part 2 of the program. The SVR voltage ratio variable is treated as a continuous variable in both part 1 and part 2 of the program. The following notation will be used in this section to refer to these variables:

- The SVR voltage ratio variable: r
- The SVR placement variable: d

The approach followed to solve this problem is explained below:

- An SVR can either be placed at an existing bus or somewhere on a branch.
- Identify the busses available for SVR placement.
- Determine which bus results in the highest end of line voltage while satisfying all the nodal voltage limits (as well as any other limits placed on SVR and branch ratings).
- Assume that the optimal SVR placement is either at this bus or somewhere along the branch either to the source or load side of this bus (The program can handle only radial networks).
- Evaluate the SVR placement along both branches to the source and load side of the bus identified in part 1 of the SVR placement program.

The influence of the above-mentioned variables is discussed next.

Placement variable: The placement variable is handled by realizing that the SVR will divide the branch into two sections with the relationship that the total length of the two new branch elements stays constant. This relationship can be expressed as follows:

$$d_{total} = d_1 + d_2 \tag{4.3}$$

This concept of handling the placement variable is shown in Figure 4-3.

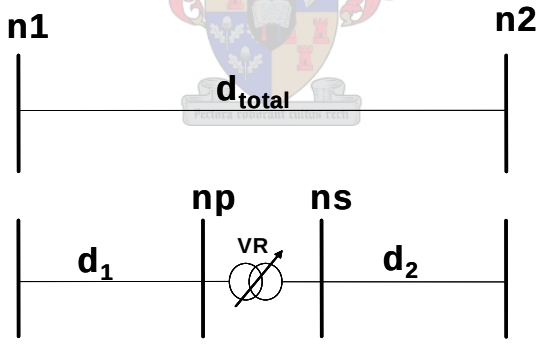


Figure 4-3: SVR placement strategy

This relationship is used to control the adjacent branch elements electrical parameters, namely resistance, reactance and capacitance (*R, X* and *B*). This variable control is done by direct manipulation of the network admittance matrix. The insertion of the SVR affects only eight entries of the network admittance matrix. This is independent of the network size or configuration. The affected entries in the network admittance matrix for a small network are shown in Figure 4-4 and Figure 4-5.

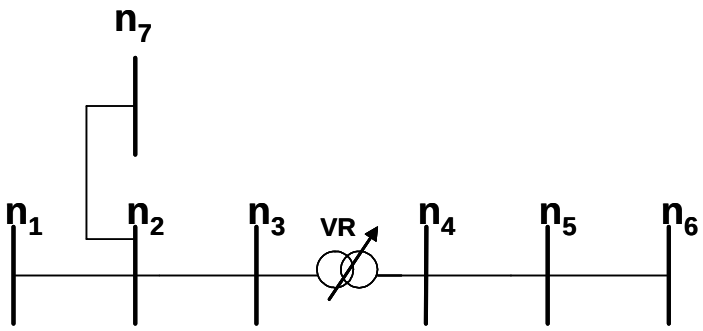


Figure 4-4: Small network with an SVR inserted

$$\begin{bmatrix} Y_{11} & Y_{12} & Y_{13} & Y_{14} & Y_{15} & Y_{16} & Y_{17} \\ Y_{21} & Y_{22} & Y_{23} & Y_{24} & Y_{25} & Y_{26} & Y_{27} \\ Y_{31} & Y_{32} & Y_{33} & Y_{34} & Y_{35} & Y_{36} & Y_{37} \\ Y_{41} & Y_{42} & Y_{43} & Y_{44} & Y_{45} & Y_{46} & Y_{47} \\ Y_{51} & Y_{52} & Y_{53} & Y_{54} & Y_{55} & Y_{56} & Y_{57} \\ Y_{61} & Y_{62} & Y_{63} & Y_{64} & Y_{65} & Y_{66} & Y_{67} \\ Y_{71} & Y_{72} & Y_{73} & Y_{74} & Y_{75} & Y_{76} & Y_{77} \end{bmatrix}$$

Figure 4-5: Admittance matrix entries affected

VR voltage ratio: In part 2 of the SVRP program the SVR voltage ratio is also used as an optimisation variable. The voltage ratio variable *r* is a totally independent smooth variable. This variable is used to control the branch electrical parameters representing the SVR.

Program structure: Figure 4-6 shows a flow diagram of the SVR placement program.

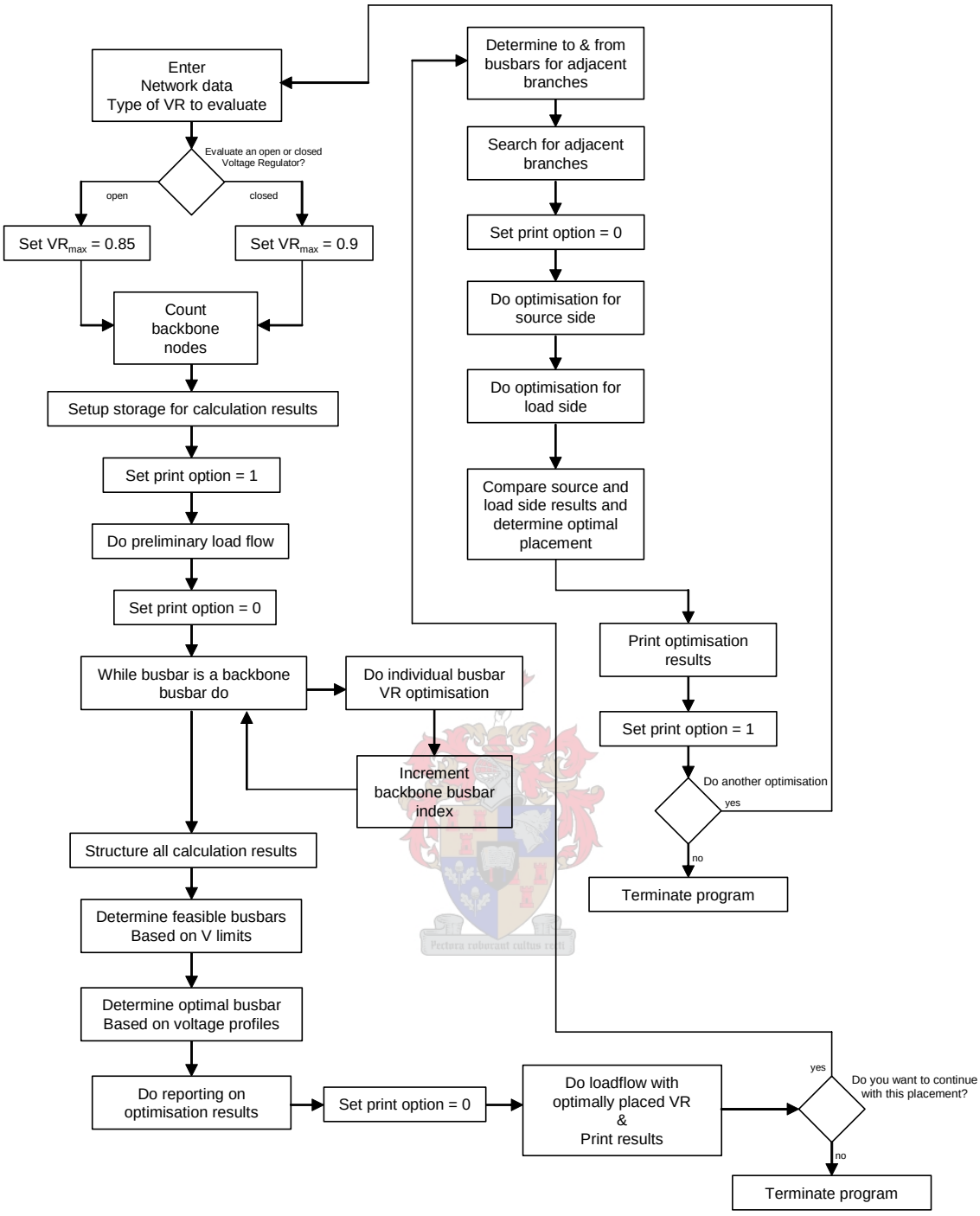


Figure 4-6: Flow diagram of the SVR placement program

The operation of the first part of the SVR placement program is now explained:

First the network data is entered. The user is asked to enter the name of the M-file that contains the network data. The case name is entered via an input box shown in Figure 4-7.

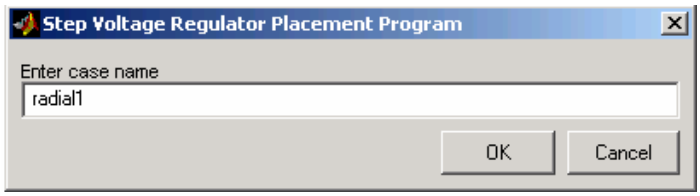


Figure 4-7: Input GUI of SVR placement program

The user must then decide if he or she wants to evaluate an open or closed delta SVR. This is done via an input box by entering a 'C' for closed delta or an 'O' for open delta configuration. This is shown in Figure 4-8.

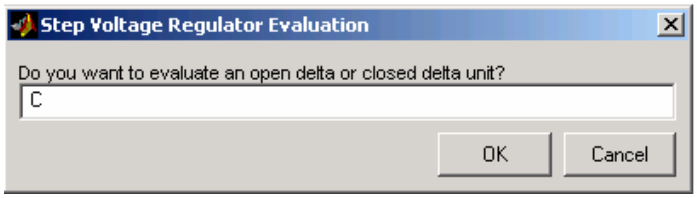


Figure 4-8: Input GUI for SVR type

Next, the program identifies the backbone of the network. The program has been developed to handle complex radial networks. That means that a radial network can have tee-offs as long as there are no parallel paths created. For the SVR placement, it is necessary to know which busses and branches belong to the backbone. The program has been developed to use the zone data of the network data structure to carry the information about the backbone. This works very well since the zone data is not used for anything else in this program. The backbone data is extracted from the bus data in the network data structure by a search algorithm that looks at the zone data column. Once the backbone busses are extracted and stored, the backbone branches are obtained by another search algorithm. This algorithm filters branches that have backbone busses on both ends. This information is needed in part 1 and part 2 of the program to place either an SVR at a bus or on a branch.

The program then does an ordinary load flow to check that the input data is valid and solvable. If the load flow does not solve, it is normally a good indication that there are errors in the data**.** The load flow is performed by the ordinary load flow function of MatPower. This function will be described briefly. The load flow is executed by the function *runpf,* which is based on the Matpower function *newtonpf*. The function initially converts the external bus numbers to a set of internal bus numbers. This is to ensure the bus numbers are sequential and start at number one. Next, the bus admittance matrix is calculated by calling a function *makeYbus*. This function uses the bus and branch data and returns the full bus

admittance matrix. The effect of positive and negative bus power injection are then structured into a complex bus power injection matrix. The necessary data is obtained from the bus and generator data of the network data structure. Now, the initial values of voltages for the Newton Raphson process are extracted from the network data and the complex starting voltages calculated. The function then runs a Newton Raphson power flow and returns the results by calling a function *Newtonpf*. This function returns the final complex voltages along with a flag which indicates whether it converged or not, as well as the number of iterations performed. The bus, generator and branch data structures are then updated to match the power flow solution. The power flow is then completed and the original bus numbers reassigned. If needed, all the power flow results are printed. This is set by the variable *printopt*.

The program continues by executing the first optimisation loop. The optimisation loop does consecutive SVR voltage ratio optimisation routines on each of the busses defined as backbone busses. The first part of the optimisation loop consists of inserting a bus SVR by calling the function *InsetbusVR*.

The *InsertbusVR* function inserts an SVR on the load side of the current backbone bus as determined by the counter variable of the optimisation loop. To enable this placement, a new bus has to be inserted first. This bus connects the secondary side of the SVR to the branch connected to the next bus of the backbone. This insertion function adds the new bus data to the bus data matrix and the SVR branch data to the branch data matrix. The SVR secondary bus number is assigned as the next available bus number, namely the highest bus number plus one. The SVR is now added as a branch between the mentioned busses. The *from* and *to* bus numbers of the new branch are assigned accordingly. The last part of the insertion function is to update the *from* bus of the old branch between the optimisation bus and the adjacent bus on the load side. This insertion strategy is illustrated in Figure 4-9.

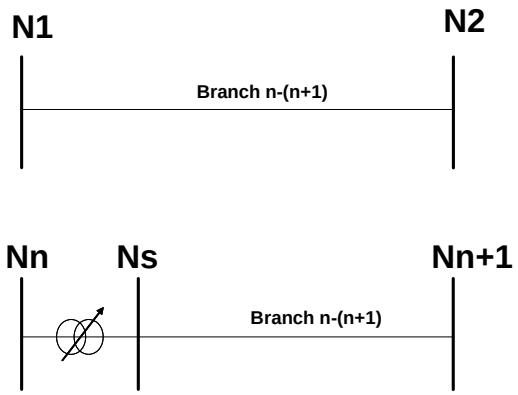


Figure 4-9: Insertion of an SVR at a bus

Optimisation process: The optimisation loop now does a non-linear optimisation on the SVR voltage ratio at the currently identified backbone bus. The optimisation process of part 1 and part 2 of the program is very similar and will be discussed in detail. The only difference in part 2 is the inclusion of the placement variable as an additional optimisation variable. This forces some changes to the structure of the problem in the structure of limits and initial values.

Store results: The voltage magnitude of all the backbone busses is stored as a column in a bus SVR voltage matrix. The SVR voltage ratio is stored in an SVR voltage ratio vector.

Increment: After each backbone bus has been optimised, the counter variable is increased by one to point to the next backbone bus and the optimisation process in started again. This is continued until all the backbone busses have been done.

Stop: When all the backbone busses have been evaluated, the optimisation loop is exited and part 1 of the SVR placement program is continued.

Determine optimal bus: From the nodal voltage magnitudes obtained from the optimisation loop, the optimal bus is now determined according to the following rules:

- All bus voltages must be within the specified limits. These limits are specified in the bus data of the network data. Each bus has an upper and a lower voltage limit associated with it. Although these limits are also included as constraints in the optimisation algorithm, the limits are tested again.
- From the allowable placements, the optimal placement is chosen as the placement that results in the highest end of line voltage.

End of part 1: The necessary reporting is done and the user is then asked if he/she wants to continue with part 2 of the SVR placement program. If the answer is yes, part 2 of the SVR

placement program is called with the necessary parameters and variables. This is done by entering 'Y' or 'YES' in the pop up box shown in Figure 4-10. If the user does not want to continue, the SVR placement program is exited by clicking the Cancel button or entering '*N*'.

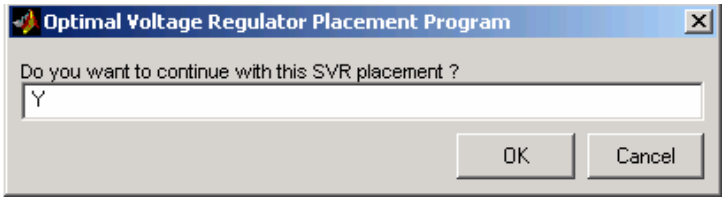


Figure 4-10: GUI to continue with part 2 of the program

Part 2: Part 2 of the program does an SVR optimisation on both sides of the bus determined in part 1 of the program. This is done by consecutive smooth optimisation processes with the SVR voltage ratio and the SVR placement as optimisation variables. The exact construction of the optimisation problem will be explained later.

Determine adjacent branches: In order to do the optimisation on either side of the optimal bus, it is necessary to determine the *from* and *to* busses of the adjacent branches. This is done by a simple search algorithm. The search algorithm will be explained for the load side branch. The search algorithm first searches for all the branches with *from* busses equal to the optimisation bus from part 1 of the program. From this it creates a list of candidate busses. From this list of candidate busses it searches for the *to* bus to be equal to one of the busses defined as backbone busses. Since the network is assumed to be radial, this search produces a unique branch. The search for the source side branch is similar to that explained for the load side branch above.

Insert branch SVR: As in part 1 of the program**,** an SVR must be inserted into the network. This time the SVR is installed somewhere along the branch identified in the previous part of the program. As a starting point for the optimisation process, the SVR is installed at the location 50% along the branch and with SVR voltage ratio equal to 1. This is significantly different from part 1 of the program in that there is no bus at this location. Because of this difference, a separate insertion function was developed to perform this task. This insertion function is described as follows. Two new busses are added to the bus matrix. The bus numbers of these two new busses are taken as the next two consecutive available bus numbers. Namely, *(last bus number+1)* and *(last bus number+2)*. The rest of the bus data for these two busses is set up as default. Next, the new branches are inserted. Three new branches are needed. The SVR is a branch inserted between the two new busses. Then a

branch on either side of the newly inserted SVR is inserted. The same type of search algorithm is used to set up the correct *from* and *to* bus indices into the branch matrix data as explained above. Since the SVR must be inserted half way along the old branch, the impedance of the two new branches is easily calculated from the old branch impedances. A branch is modeled using the equivalent PI model. The last part of the insertion process is to remove the old branch from the branch matrix.

SVR voltage ratio and placement smooth optimisation: The optimisation loop now does a non-linear optimisation on the SVR voltage ratio and placement on the currently identified backbone branch. The optimisation process of part 1 and part 2 of the program are very similar and will be discussed in detail later. The only difference is that in part 2, the placement variable is included as an additional optimisation variable. This forces some changes to the structure of the problem in the limits and initial values structure.

Store results: The voltage magnitude, placement variable and SVR voltage ratio values are stored for further calculation and reporting.

Implementing ordinary power flow: After a successful optimisation is completed, an ordinary Newton Raphson power flow is done with the standard Matpower reporting format on bus voltages, network losses and branch flows.

Reporting: Additional optimisation reporting is done to give the values of the two optimisation variables. The SVR voltage ratio in given in p.u. and the SVR placement variable in percentage.

Optimisation process: The following is a detailed description of the optimisation process of part 1 & 2 of the program.

Set up initial guess: The optimisation algorithm used by the *fmincon* function needs a feasible starting point to ensure a feasible solution to the optimisation process. The feasible starting point is set up by assuming all voltage magnitudes equal to 1 p.u. and all voltage angles as angle zero. This is not guaranteed to be a feasible starting point but is close enough to the feasible region for the algorithm to effectively solve the placement problem. Technically a better approach would be to first do an ordinary load flow and use those voltage magnitude and angle results to populate the feasible starting point. Such a load flow must obviously be done after the SVR has been inserted to account for the effect that the SVR

impedance and voltage ratio will have on the voltage magnitude and angles in the load flow. Also the limits of the new busses inserted must be accounted for. The feasible starting point for the SVR voltage ratio and the placement variable is obtained from arbitrarily setting up initial values within the specified limits. These values have been set up as follows:

- SVR voltage ratio = 1, i.e. the SVR is initially set to nominal voltage ratio
- Placement variable = 0.5, i.e. initially the SVR is placed in the centre of the branch

The above initialisation is done by setting up a row vector called *x0* containing the initial variable values as described. This vector is automatically sized and populated by this part of the program. The size of the vector is determined by assuming only one reference bus in the network (this is currently a limitation of this program) and no PV busses (Power Voltage busses not supported at this stage). With these assumptions the number of voltage magnitude and voltage angle variables is equal to *2*n-2* where *n* is the number of busses including the reference bus. Added to this is the SVR voltage ratio variable and the placement variable, which brings the total number of variables to *2*n*. With this information the *x0* vector can easily be sized from the bus data matrix, which is an input variable to this part of the program.

A useful feature of the voltage magnitude and voltage angle variables is that their values fall within a relatively narrow range when expressed in per unit and in radians. The placement variable expressed in percentage and the SVR voltage ratio has this same useful feature. At this stage of the program, the necessary additional busses and the SVR branch element have already been inserted and the necessary data changes have been made to the network data in the form of the bus data and the branch data.

Upper and lower limits: The next step is to set up the upper and lower limits of the optimisation variables. The sizing of the variable space is the same as described above for setting up the initial value. The structure of the variable space is as follows:

$$x = \left[V_{Mag_{node\,2}} \cdots V_{Mag_{noden}}, V_{Ang_{node\,2}} \cdots V_{Ang_{noden}}, VR_{vratio}, VR_{Placement} \right] \tag{4.4}$$

The upper and lower limits of the variable space must be structured in the form $lb \le x \le ub$ where lb is the lower limit vector and ub is the upper limit vector. The limits for the voltage magnitude are the upper and lower voltage limits as specified in the bus data of the network data. The limits for the voltage angles are set to ± 90º. If voltage stability is a factor and needs to be included, the voltage angle limits can be set to obtain any required stability margin. The limits for the SVR voltage ratio are set automatically according to the user

specification for maximum SVR boost and buck. The limits on the placement variable are theoretically $0 \le VR_{Placement} \le 1$. However, to prevent numerical instability (division by zero) the limits have however been implemented as $0.001 \le VR_{Placement} \le 0.999$.

Non-linear equality limits: The non-linear equality limits form the heart of the optimisation process as they put very strict boundaries on the variable space in the form of the power flow equations. This is because the power flow equations must always be satisfied exactly as they are implemented as equality constraints. The *fmincon* function requires the non-linear equality and inequality constraints to be specified in a separate function. The variables and parameters are passed to this function via the *fmincon* function and the constraint variables are returned to the *fmincon* function. The following parameters are used by the non-linear constraint function *VRCon*:

- MVABase
- Bus data
- Branch data
- *from bus* of the SVR
- *to bus* of the SVR

The non-linear constraint function *VRCon*: The *VRCon* function performs the following functions:

- Modifies the bus admittance matrix
- Sets up the bus power vector
- Sets up the complex voltage vectors from the voltage magnitude and angle variables
- Sets up the complex power equations
- Sets up real and reactive power equations from the complex power equations

These functions will now be explained.

Modify the bus admittance matrix: The bus admittance matrix is modified by calling a low level function called *ModYBus*. The purpose of the *ModYbus* function is to include the effect that the optimisation variables, SVR voltage ratio and the placement variable, have on the bus admittance matrix. The *ModYBus* function accepts the following input parameters:

- baseMVA
- bus data matrix

- branch data matrix
- *from bus* of the SVR
- *to bus* of the SVR
- SVR placement variable
- SVR voltage ratio variable

The function first finds the branches that need to be changed. It then includes the necessary variables in the branch data and calculates the new modified bus admittance matrix, which is now a function of the optimisation variables, namely the SVR placement variable and the SVR voltage ratio variable. The calculation of the bus admittance matrix is similar to that used in MatPower. The formulae for this can be found in most books that cover load flow analysis and will therefore not be discussed further.

Set up the bus power vector: The bus power vector is a row vector consisting of the constant power loads located at each bus. Each of the entries is a complex quantity of the form: $P + jQ$. Constant impedance loads are not included in the bus power vector. They are included as shunt impedances in the bus data of the network data and are absorbed into the bus admittance matrix. The bus power vector is extracted from the bus matrix data by using the correct indexing.

Set up the complex voltage vectors from the voltage magnitude and angle variables: The complex voltage vectors are set up by inserting the necessary indexing into the variable space $\bar{x}$ and then using the following formulae:

$$\overline{V_n} = V_{Mag_{nodeN}} e^{j\frac{PI}{180}V_{Ang_{nodeN}}} \tag{4.5}$$

$$x = \left\lfloor V_{Mag_{node2}} \ldots V_{Mag_{nodeN}}, V_{Ang_{node2}} \cdots V_{Ang_{nodeN}}, VR_{vratio}, VR_{Placement} \right\rfloor \tag{4.6}$$

Set up the complex power equations: Kirchoff's law states that the sum of the power entering a bus is equal to the power leaving that bus. Power can enter or leave a bus via a shunt element or a series element. A shunt element can be a load, a generator, a shunt reactor or a shunt capacitor. A series element is a branch that connects this bus to another bus and can represent a line, a transformer or a series capacitor or reactor. For the purpose of this program, the generator is connected to a reference or a slack bus. The program does not cater

for PV busses. Therefore, the complex power equations do not have to be solved at the bus connected to the reference bus. This is because the voltage magnitude and angle are already known. For the rest of the busses, the complex power entering and leaving each bus is calculated as follows:

$$S_k = V_k \sum_{n=1}^{N} Y_{kn} V_n \tag{4.7}$$

This set of non-linear equations is enough to solve the power flow of the given network.

Set up real and reactive power equations from the complex power equations: In order to have as many equations as variables, it is necessary to break up the complex power equations into real and reactive power equations. These equations can be expressed as follows:

$$P_k = V_k \sum_{n=1}^{N} Y_{kn} V_n \cos(\delta_k - \delta_n - \theta_{kn}) \tag{4.8}$$

$$Q_k = V_k \sum_{n=1}^{N} Y_{kn} V_n \sin(\delta_k - \delta_n - \theta_{kn}) \tag{4.9}$$

$$k = 1,2,...,N$$

Objective function: The objective function of the program is defined in a function called *SVRObjective*. All optimisation variables are available within this function to define an objective function. The objective function for the program and case studies presented in this thesis is simply to maximise the magnitude of the end of the line voltage.

This concludes the description of the placement program developed for optimal SVR placement. The use of the program will be illustrated with the aid of two case studies in Sections 5.3 and 5.4. The next section will discuss the program developed for optimal EVR placement.

4.3 LV ELECTRONIC VOLTAGE REGULATOR PLACEMENT

The EVR placement program has been developed to place EVRs on a given single phase, low voltage radial network. The EVRs have a 5 kVA power rating and voltage regulation capability of up to 20% voltage boost. It has been determined that these ratings are most suitable for application on rural and deep rural applications. The objective function of the optimal placement problem is to minimise the installation cost of the EVRs while maintaining acceptable voltage levels on all the load nodes. Because of the fixed cost per EVR this objective function is equivalent to minimising the total number of EVRs installed on a given network. This multiple placement of EVRs on a network leads to the reconfiguration of certain parts of the network. This reconfiguration is a discrete operation and is implemented by updating the network admittance matrix. Due to the discrete nature of the problem, it was decided to use a genetic algorithm to solve it. The genetic algorithm adopts the principle of natural selection to optimise the number of combinations of multiple EVR placements on a given network. Solutions are initially created randomly and then, by the mechanisms of genetic operators, move towards better solutions. The program developed uses a single population genetic algorithm and each individual within the population is a binary encoded chromosome as shown in Figure 4-11.

1	0	0	1	1	1	0	1

Figure 4-11: EVR placement chromosome

Each allele of the chromosome has a binary value indicating whether or not an EVR is installed at a certain node or not. The chromosome in Figure 4-11 represents a network with eight candidate nodes and with EVRs installed at nodes 1, 4, 5, 6, and 8. The feasible solution space is defined by the upper and lower node voltage limits as well as the maximum power rating of the EVRs. These constraints are as follows:

$$V_{\min} \le V_i \le V_{\max} \tag{4.10}$$

$$S_i \le S_{EVR} \tag{4.11}$$

Maintenance of a feasible population: Genetic algorithms are not inherently capable of handling constraints. Constraints in genetic algorithms can be handled by either discarding infeasible individuals or by adding a suitable penalty function corresponding to the constraints. In this program the former is chosen and found to work well. Due to the

structure of the program, feasibility is tested on the entire population. If an individual is found to be infeasible, a replacement is immediately created via the creation function *CROMGEN*. The new individual/s is inserted and the population is tested again. This process is repeated until the entire population is feasible.

Creating the initial population: New individuals are created by the function *CHROMGEN*. This function is called within the main program to generate the initial population as well as to generate new individuals to replace infeasible individuals throughout the algorithm. The syntax of this function is as follows:

$$CHROM = CHROMGEN(VRnum,NVAR,NIND) \tag{4.12}$$

Input parameters: The *CHROMGEN* function eccepts the following input parameters:

- ***VRnum*:** This is a user-defined value that limits the allowable maximum number of EVRs that will be installed on the given network. The program's performance is greatly enhanced by limiting the total number of possible EVR installations as it reduces the total number of possible combinations and limits the number of infeasible individuals created by the *CHROMGEN* function. This control of the genetic makeup of the generated chromosomes has been specifically designed for the program and is not part of the standard genetic algorithm.

- ***NVAR*:** *NVAR* is an internally calculated variable that controls the size of the chromosomes of a population. This is done by checking the number of nodes present in the input file containing the network data. The *NVAR* parameter is calculated within the main program *SGAVR*.

- ***NIND*:** *NIND* is the number of individuals within a population for a specific study. The value is user definable through the input GUI. The number of individuals should not be too high as they slow down the optimisation process. They should also not be too low as this limits the search ability of the algorithm. For the networks presented in the case study a value of four produced the best results.

Output parameters: As output the *CHROMGEN* function gives the randomly generated individuals as a matrix with each individual a row within the matrix.

Fitness: The fitness of an individual is calculated by the *OBJVR1* function. This function is a link function between the main program *SGAVR* and the load flow function *GeneticLV*. The *OBJVR1* function handles the individual calls to the load flow function and records the fitness of each individual of a specific population. The syntax of this function is as follows:

$$[ObjVal,feas] = OBJVR1(Chrom,NVAR,file) \quad (4.13)$$

Input parameters: The *OBJVR1* function accepts the following input parameters:

- ***Chrom*:** A two-dimensional matrix containing the current population. Individual chromosomes are structured as rows.
- ***NVAR*:** The size of the chromosomes. Not currently used.
- ***file*:** The name of the M file containing the network data. This variable is passed on from the main program *SGAVR* and is passed on to the load flow function *GeneticLF*.

Output parameters: The *OBJVR1* function has the following output variables that are passed on to the main program *SGAVR*:

- ***ObjVal*:** A column vector containing the fitness value of an individual. The fitness value is equal to the number of EVRs installed by the specific individual.
- ***Feas*:** A row vector containing the feasibility information of each individual within a newly evaluated population. Feasible individuals are represented by a one, and infeasible individuals are represented by a zero.

Ranking function: The ranking of individuals in a population is done by the function *RANKING*. The syntax of this function is as follows:

$$FitnV = RANKING(ObjV, RFun, SUBPOP) \quad (4.14)$$

This function ranks individuals represented by their associated fitness, and returns a column vector *FitnV* containing the corresponding individuals rank. For multiple sub populations the ranking is performed separately for each sub population. The program developed uses only one population.

Input parameters: The *RANKING* function accepts the following input parameters:

- ***ObjV*** - Column vector containing the fitness values of the individuals in the current population.

Output parameters:

- ***FitnV*** – A column vector containing the rank values of the individuals in the current population.

Selection of individuals for recombination: The selection process is performed by the *SELECT* function. This function performs universal selection and is capable of handling multiple populations. The *SELECT* function is called from the main function *SGAVR* and makes use of calls to low level selection functions for the actual selection process. The syntax of this function is as follows:

$$SelCh = SELECT(SEL\_F, Chrom, FitnV, GGAP, SUBPOP) \quad (4.15)$$

Input parameters: The *SELECT* function accepts the following input parameters:

- ***SEL_F*** - Name of the selection function. Stochastic Universal Sampling is used for this program. This type of selection is implemented by the *SUS* function.

- ***Chrom*** - Matrix containing the individuals of the current population. Each row corresponds to one individual.

- ***FitnV*** - Column vector containing the fitness values of the individuals in the population.

- ***GGAP*** - (optional) Generation gap is the rate of individuals to be selected. If omitted 1.0 is assumed. *GGAP* is set to 0.5 for this program. A selection of *GGAP* to 0.5 ensures that the number of new individuals generated will be half the size of the population. This, together with an adequate reinsertion strategy, implements elitism in the *SGAVR* program. This feature also contributes to the performance of the program.

Output parameters: The select function produces the following output:

- ***SelCh*** – A two-dimensional matrix containing the selected individuals.

Recombination: *RECOMBIN* is the function that performs recombination between pairs of individuals and returns the newly generated individual. The function handles multiple populations and calls the low-level recombination function for the actual recombination process. The syntax of this function is as follows:

$$NewChrom = RECOMB(REC\_F, OldChrom, RecOpt, SUBPOP) \quad (4.16)$$

Input parameters: The *RECOMB* function accepts the following input parameters.

- ***REC_F*:** A string containing the name of the recombination or crossover function.
- ***Chrom*:** A matrix containing the chromosomes of the old population. Each line corresponds to one individual

Output parameter: The *RECOMB* function produces the following output parameter.

- ***NewChrom***: A matrix containing the chromosomes of the population after recombination. The format of the matrix is the same as *OldChrom*.

Crossover: The *XOVSP* function performs single-point crossover between pairs of individuals and returns the current generation after mating. The syntax of the function is as follows:

$$NewChrom = XOVSP(OldChrom, XOVR) \quad (4.17)$$

Input parameters: The *XOVSP* function accepts the following input parameters:

- ***OldChrom*** – A matrix containing the chromosomes of the old population. Each line corresponds to one individual.
- ***XOVR*** – The probability of recombination occurring between pairs of individuals.

Output parameter: The *XOVSP* function produces the following output.

- ***NewChrom*** – A matrix containing the chromosomes of the population after crossover, ready to be mutated and/or evaluated, in the same format as *OldChrom*.

Stochastic Universal Sampling: The *SUS* function performs selection with Stochastic Universal Sampling. The syntax of this function is as follows:

$$NewChrIx = SUS(FitnV, Nsel) \quad (4.18)$$

Input parameters: The SUS function accepts the following input parameters:

- ***FitnV***: A column vector containing the fitness values of the individuals in the population.

- ***Nsel*:** The number of individuals to be selected.

Output parameters: The *SUS* function produces the following output:

- ***NewChrIx*** – A column vector containing the indexes of the selected individuals relative to the original population, but shuffled. The new population is obtained by calculating *OldChrom(NewChrIx)*.

Mutation: The *MUT* function takes the representation of the current population, mutates each element with given probability and returns the resulting population. The syntax of this function is as follows:

$$NewChrom = MUT(OldChrom,Pm,BaseV) \quad (4.19)$$

Input parameters: The *MUT* function accepts the following input parameters:

- ***OldChrom*** - A matrix containing the chromosomes of the current population. Each row corresponds to an individual's representation.

- ***Pm*** - Mutation probability (scalar). Default value of $Pm = 0.7/Lind$, where *Lind* is the chromosome length.

Output parameter: The *MUT* function produces the following output parameters:

- ***NewChrom*** - A Matrix containing a mutated version of *OldChrom*.

Insertion of new individuals: The *REINS* function reinserts offspring in the population. The syntax of this function is as follows:

$$[Chrom] = REINS(Chrom, SelCh, SUBPOP, InsOpt, ObjVCh, ObjVSel) \quad (4.20)$$

Input parameters: The *REINS* function accepts the following input:

- ***Chrom*** – A matrix containing the individuals of the current population. Each row corresponds to one individual.

- ***SelCh*** – A matrix containing the offspring of the current population. Each row corresponds to one individual.

- ***SUBPOP*** - (optional) The number of sub populations. If omitted, 1 sub population is assumed. The *SGAVR* program uses one population.

Output parameters: The *REINS* function produces the following output:

- ***Chrom*** – A matrix containing the individuals of the current population after reinsertion.

This concluded the detailed description of the important functions of the *SGAVR* program developed to place EVRs on low voltage networks. The following is a short overview of the program and its features.

The *SGAVR* program has been developed in Matlab and makes use of two Toolboxe, namely the Genetic Algorithm Toolbox and MatPower. The program has been based on the Matpower data structures and the genetic operators of the Genetic Algorithm Toolbox. The features of the program can be summarised as follow:

- The program solves low voltage (230 V), single phase networks.
- The program can accept networks of any practical size.
- The program can only handle radial networks.
- The program uses a power base of 100 kVA. This ensures that most LV networks result in matrices that are well scaled and ensures that no numerical difficulties are experienced.
- The program is based on a simple genetic algorithm with a slight modification to handle constraints.
- Another modification to the standard genetic algorithm has been implemented to limit the number of EVR installations to a user specified number. This accelerates the optimisation process considerably.

Figure 4-12 shows a flow diagram of the program.

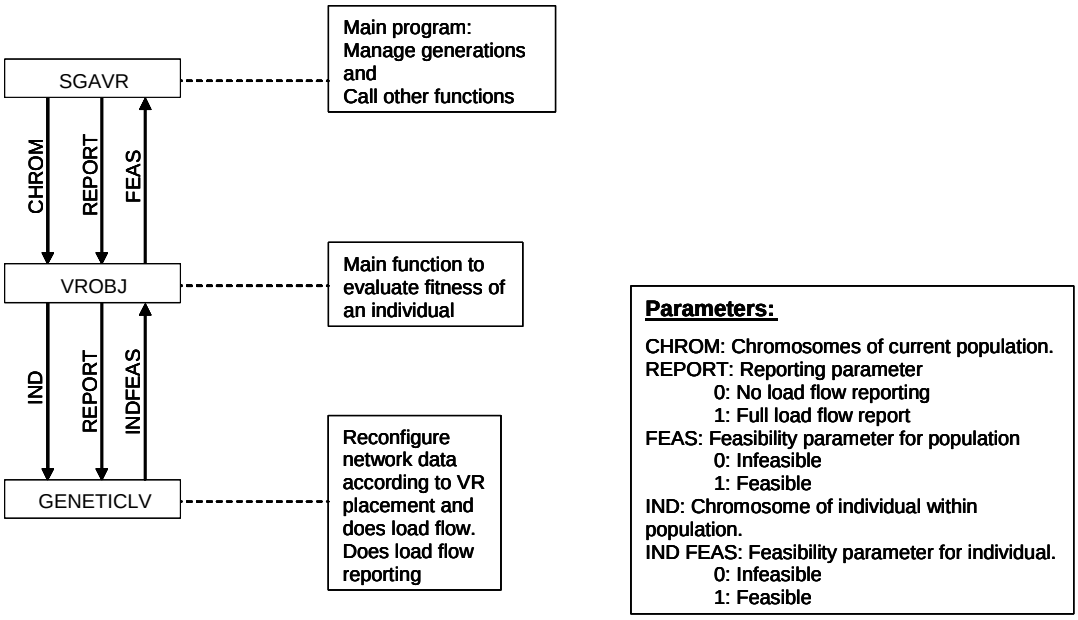


Figure 4-12: Main parts of the *SGAVR* program

A Graphical User Interface has been designed for the *SGAVR* program to make easy access to the algorithm parameters possible. See Figure 4-13.

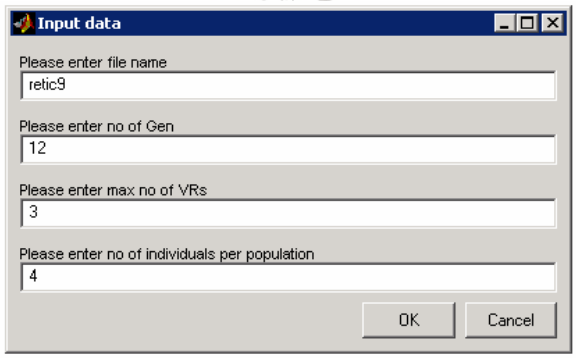


Figure 4-13: *SGAVR* input GUI

From this graphical user interface the user can enter the following parameters:

- The file name of the M-file containing the network data
- The maximum number of generations that must be executed
- The maximum number of EVRs to be installed on the network
- The size of a population

This concludes the discussion of the EVR placement program developed as part of this thesis. Chapter 5 shows how the program is used to place EVRs on a typical Eskom electrification network.

CHAPTER 5 CASE STUDIES

5.1 OVERVIEW

This chapter discusses four case studies to show the application of the optimisation programs developed in this thesis. The first case study illustrates the placement of a compensator on the 132 kV Eros-Zimbane-Pembroke network to solve a thermal overload problem currently experienced. The PSAT program was used to evaluate the application of a Phase Angle Regulator (PAR) to solve this network problem. The results show that a PAR will be a very effective solution to this problem.

The second case study details the placement of a 22 kV Step Voltage Regulator (SVR) on a simplified radial network. The purpose of this study is to show that the SQP technique implemented in the placement program is able to solve a multi objective optimisation problem. The multi objective optimisation problem was set up by optimising both the SVR primary side voltage, as well as the end of line voltage. The results are compared with PowerFactory simulations and found to be very accurate.

The next case study examines the placement of an SVR on the 11 kV Kwaaihoek – Whitney network in the Eastern Cape. The placement program was used to optimise the end of line voltage of this network by the placement of a single SVR. The result obtained is compared to the existing placement of the SVR and found to be very similar.

The last case study uses the EVR placement program to optimise the voltage distribution on a 230 V LV network. The network used is based on the Thaba Lesobo network in the Aliwal North area. Two adjacent transformer zones of this electrification project were linked to investigate the ability of the EVR to stretch LV networks. The network parameters were then scaled to simulate a typical deep rural network and the optimisation routine was repeated. In both cases the program was able to place EVRs to regulate the voltage distribution within limits.

5.2 CASE STUDY 1: 132 KV PEMBROKE – ZIMBANE – KOKSTAD NETWORK.

This case study shows the use of the PSAT optimisation software to analyse the application of a PAR on the 132 kV Eros – Zimbane – Pembroke interconnected network. The purpose of the device is to regulate the transfer of active power in order to alleviate a thermal overload condition. A PAR can be realised either as a solid state device [B21] or an electro-mechanical device [B20]. The latter is also known as a Phase Shifting Transformer (PST) and is the conventional method of achieving phase shift.

The 132 kV Eros – Zimbane – Pembroke Eskom distribution network is 365 km long, it is supplied from two substations and supplies a peak load of 115 MVA to the Butterworth, Ncora and Umtata areas. Figure 5-1 shows the geographical position of the network. The network is interconnected with 70 MW of distribution generation and has a +35 MVAR, -10 MVAR SVC situated at Zimbane substation near Umtata city. Two 16 MVAR capacitor banks have been installed at Kokstad substation to improve the 132 kV voltage level. Three 5 MVAR static capacitor banks have also been installed to maintain voltage levels in different areas. On the Southern side the network, voltage is regulated at Pembroke substation via two 220/132 kV transformers and on the northern side it is regulated at Ariadne substation via two 400/132 kV transformers. From Ariadne, the network is unregulated all the way up to Zimbane.

The most serious problem on this network is an overload condition on the Eros - Kokstad line during peak load conditions. This line is built with Wolf conductor on a wooden structure, which is templated for 110 MVA at 70 ºC. Due to the loading problem on the Eros - Kokstad network, the loading on the Kokstad - Zimbane line is limited to 70 MVA at Qumbu substation. This limit is implemented with a trip signal on the Qumbu-Mnt Frere 132 kV breaker at Qumbu substation.

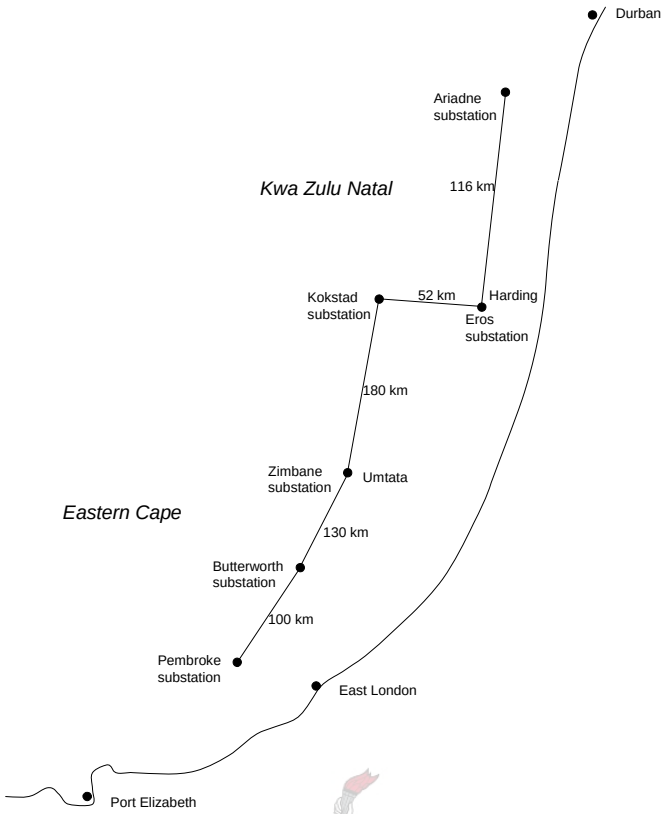


Figure 5-1: Geographical layout of the Eros-Zimbane-Pembroke line

During the winter months when this area experiences its peak load, the loading limit causes major operational problems on this network. Manual load shedding is often required to limit the loading on the Kokstad – Zimbane line and on several occasions over the last few years the whole system had been blacked out.

As can be seen from Figure 5-2, the contribution to the Umtata load from Kokstad is much more than from Pembroke. This high load contribution from Kokstad is due to a power angle of around 35 º between Pembroke and Ariadne, with the Ariadne side leading. The effect of this imbalance in load transfer is that the 132 kV Pembroke – Zimbane line is underutilised and cannot be used to overcome the overloading on the 132 kV Eros – Kokstad line.

The other major problem experienced on this network is low voltage in the Zimbane and Kokstad areas during peak load and contingency conditions. The Zimbane voltage depends heavily on the Zimbane SVC as well as the distribution generation in the area. Even with the SVC in service, the 132 kV Zimbane voltage reaches a level as low as 120 kV during periods of high system loading.

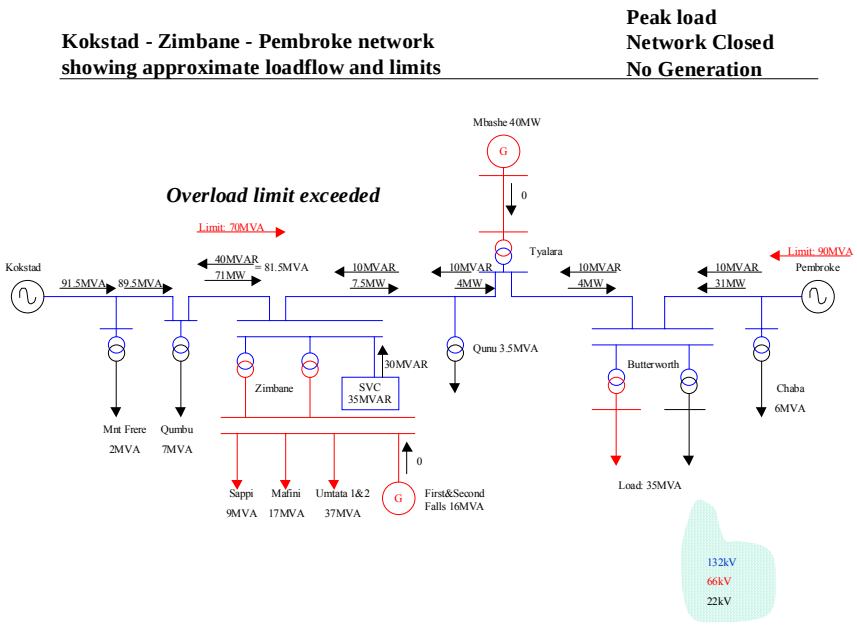


Figure 5-2: Kokstad – Pembroke network

As explained, the major problem on this network is an imbalance in the transfer of active power from the two source substations Pembroke and Ariadne. This imbalance leads to severe overload conditions on the Eros-Kokstad line. This problem is currently managed by splitting the network during periods of peak load as well as to use the local hydro generation to control the loading as far as possible. To find a sustainable solution to the mentioned problems the following two alternatives are discussed:

- Alternative 1: Operate the network radial by splitting the network at Zimbane substation.
- Alternative 2: Control the active power flow on the Eros – Pembroke network.

Alternative 1: The first alternative is to operate the network radial by splitting the network at Zimbane substation. Zimbane substation has a double 132 kV bus as well as a double 66 kV bus. It also has two 40 MVA, 132/66 kV transformers supplying the 66 kV load. The 66 kV load is made up of four 66 kV feeders that supply surrounding areas. The 132 kV and 66 kV bus couplers can be used to split the 132 kV bus and the 66 kV bus so that half the load is supplied from the Pembroke side and the other half of the load is supplied from the Eros side. Splitting the network at Zimbane substation has the following advantages and disadvantages:

Advantages:

- Cost effective. No additional equipment has to be installed.

Disadvantages:

- The SVC can only control one side of the network. This means that the other side might suffer from low voltage during high load periods.
- Both sides of the network are radial supplies, which means that any fault on either side of the network will result in a loss of supply.
- Synchronising the two networks with the local distribution generation can become complicated.

Alternative 2: Control the power flow on the 132 kV Pembroke – Zimbane and the 132 kV Kokstad - Zimbane lines by installing a suitable PAR in the form of a PST. The installation of a PST has the following advantages and disadvantages:

Advantages:

- It will fully utilise the power transfer capacity of the Pembroke – Zimbane line. If this line can be fully utilised the total available supply to the Butterworth – Zimbane area would be 200 MVA. This is much higher than the present load of about 120 MVA. Together with 70 MVA of peaking reserve, in the form of local hydro generation, this will be more than enough to supply the area far into the future.
- It will effectively eliminate the thermal overload on the Eros-Kokstad line.
- Voltage profile will be improved and losses reduced. Some of the beneficial side effects of having an improved active power transfer is an improved voltage profile along the 132 kV network as well as a reduction in technical losses.

Disadvantages:

- Cost. Preliminary design and costing of a suitable PST shows that the total project cost for purchasing and commissioning of a PST is R 10m.

The rest of this section will show the modelling of a PST on the Eros-Zimbane-Pembroke network using the PSAT software. In order to do the analysis, the parameters for the network model were first determined by doing the necessary simulations in Power Factory. The uncompensated network was then modelled in PSAT to determine its power transfer characteristics over a wide range of operating conditions in the form of an angle sweep. The model for the uncompensated network includes the Zimbane SVC. An optimisation routine was then run to do the placement of a suitable PAR. The objective function for this optimisation routine has been set to control active power transfer by specifying a phase angle set point.

The network model for the uncompensated network was determined from simulations done in Power Factory. Figure 5-3 shows the single line presentation of the Eskom network from Zimbane substation up to the point of common coupling at Tutuka substation. The result boxes show the results of an approximate load flow. For the purpose of determining the voltage angle at Zimbane, the Zimbane 132 kV bus was split so that bus 2 is supplied from Pembroke and bus 1 is supplied from Kokstad. This network model was used to simulate the power angle at Zimbane between the Pembroke and the Ariadne sources. From this approximate load flow simulation, the power angle is seen to be 35.76º. This value correlates well with what is experienced on the real network.

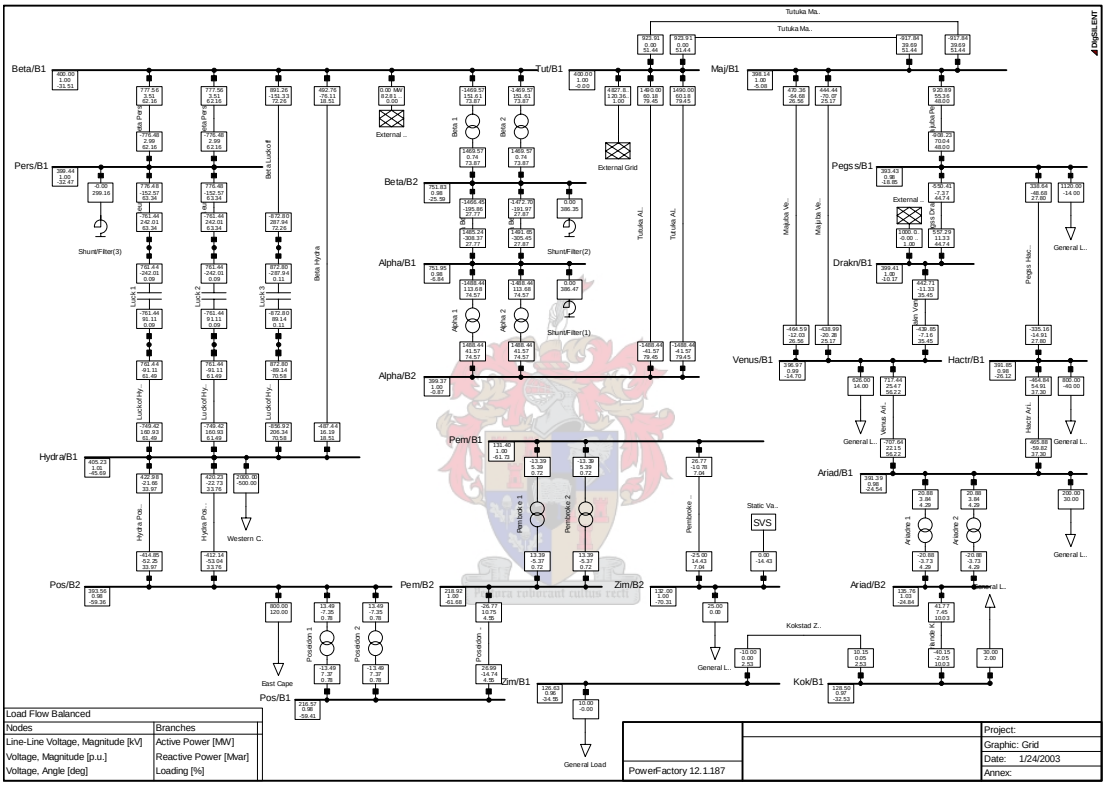


Figure 5-3: Approximate load flow of interconnected network

In order to use the analysis software to evaluate the PAR, an approximate two-source network model was developed. In this network model, Pembroke 132 kV bus is modelled as the sending end source and Eros 132 kV bus is modelled as the receiving end source. The equivalent source voltage and impedances for the sending and receiving end are calculated using equations (5.1) to (5.4).

$$Z_{Series} = \frac{V_{Z_{Series}}}{I_{SS}} = \frac{|V_{SS}|^2 - V_{PCCS}V_{SS}^*}{S_{SS}^*} \tag{5.1}$$

$$Z_{Shunt} = \frac{V_{PCCS}}{I_{Shunt}} = \frac{|V_{PCCS}|^2 V_{SS}^*}{S_{SS}^* V_{PCCS}^* - S_{PCCS}^* V_{SS}^*} \tag{5.2}$$

$$V_S = \frac{V_{SS} Z_{Shunt}}{Z_{Series} + Z_{Shunt}} \tag{5.3}$$

$$Z_S = \frac{Z_{Series} Z_{Shunt}}{Z_{Series} + Z_{Shunt}} \tag{5.4}$$

The following parameters have been used for the Pembroke equivalent source model:

- Pembroke voltage: VSS = 1.05 p.u. ∠0.251 rad
- Pembroke power: SSS = 0.826 + j0.563 p.u.
- Zimbane voltage: V_{PCCS} = 0.92 p.u. ∠0 rad
- Zimbane power: S_{PCCS} = 0.565 p.u. ∠0 rad

These parameters include the effect of the source impedance as well as the effect of the network between the sending end and the point of common coupling at Zimbane substation. The Zimbane load is modelled as a constant impedance load of 65 MW and 35 MVAR. The real part of the load represents the load supplied by the 66 kV bus at Zimbane substation. This load is modelled as a constant impedance load at unity power factor. The reactive part of the load represents the Zimbane SVC loading. This load has been modelled as a constant power load. Because the transfer of active power is the main concern in this study, the angle stability analysis is the best approach to establish the effectiveness of a compensator to control the active power transfer. Because of the direct relation between active power transfer on a network and voltage angle difference, the voltage angle can be regulated directly to achieve active power regulation. This was done because phase angle is a standard objective function in the optimisation algorithm of the program. The result of an angle stability analysis on the uncompensated network is shown in Figure 5-4.

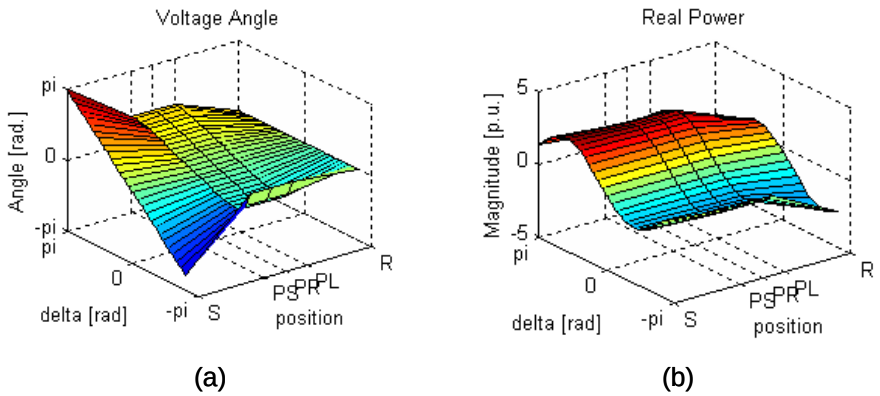


Figure 5-4: (a) voltage angle and (b) active power transfer across the Zimbane network

The variation of active power transfer has a typical sinusoidal shape across the network. It can further be seen that in the operating range around -35 degrees, there is actually an active power transfer from Eros towards Pembroke. At -35 degrees, the active power transfer is as follows (all quantities are given in p.u. on a 100 MVA base).

- Total power supplied from the Pembroke network: -0.1624 + 0.1772i p.u.
- Power supplied to Zimbane from the Pembroke network: -0.1639 + 0.1603i p.u.
- Power supplied to Zimbane from the Eros network: -0.7090 + 0.1603i p.u.
- Total power supplied by the Eros network: -0.8021 + 0.0543i p.u.

This clearly shows the undesirable imbalance in active power contribution from the Pembroke side of the network. The same network model has been used to analyse the effect of inserting a PST at Zimbane substation to regulate the active power flow. The device is installed on the Pembroke side of the Zimbane load. The results of this study are shown in Figure 5-5.

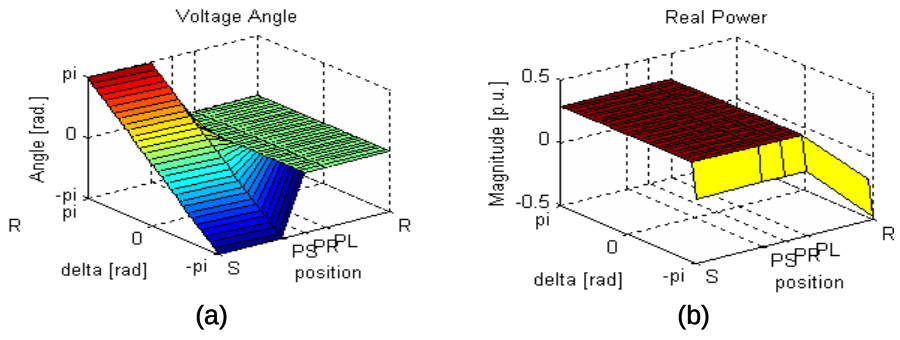


Figure 5-5: (a) voltage angle and (b) active power transfer control with a PST

As can be seen from Figure 5-5, the active power transfer from Pembroke and Eros can be controlled to a desirable operating point by applying the correct phase angle regulation in the

form of a PST. For this case study, the active power control has been achieved as follows (all quantities are given in p.u. on a 100 MVA base).

- Total power supplied from the Pembroke network: 0.3471 + 0.0855i p.u.
- Power supplied to Zimbane from the Pembroke network: 0.3437 + 0.0481i p.u.
- Power supplied to Zimbane from the Eros network: -0.2211 - 0.2519i p.u.
- Power leaving Eros network: -0.2402 - 0.2736i p.u.

The power and phase angle rating of such a device is a complex issue and will not be discussed as part of this thesis. The basic principle of active power control is, however, clearly demonstrated. Because of its effective solution to the Eros – Zimbane – Pembroke network, the application of PSTs by Eskom Distribution has been supported by Eskom Resources and Strategy research. A research report [B37] on the application of PSTs covers the application of this technology in detail.

This case study shows the PSAT optimisation program is capable of controlling active power transfer over an interconnected network by applying a PAR. The SQP optimisation routine determined the correct operating point for the compensator for the entire phase angle operating range of – pi to + pi radians. The PSAT program can further be used to determine the design rating of a suitable PST for this network. However, in order to do this however more load flow simulations will have to be done to determine worst case operating scenarios.

5.3 CASE STUDY 2: 22 KV STEP VOLTAGE REGULATOR PLACEMENT

In this case study the SVR placement program described in 4.2 is used to optimally place a 22 kV open delta SVR on a radial reticulation network. The purpose of this case study is to prove the accuracy of the optimisation algorithm as well as that of the Newton-Raphson load flow solver. This is done by back substituting load flow results from PowerFactory into the objective function of the optimisation algorithm and comparing results. The objective function for this optimisation problem has been defined as to optimise the voltage at the incoming side of the SVR as well as the voltage at the end of the line. This is a very good example of a constraint, non-linear optimisation problem and serves as a good test of the optimisation algorithm to give accurate results. Additional non linearities are introduced by the SVR and the constant impedance load model used. The constraints are introduced by the voltage limits on all the busses as well as the loading limits on all the branches including the SVR itself. Figure 5-6 shows a photo of a typical open delta voltage regulator installation. The model used to represent an SVR is shown in Figure 4-2.

Figure 5-6: Photo of an open delta SVR

The PowerFactory network model used to verify results is shown in Figure 5-7.

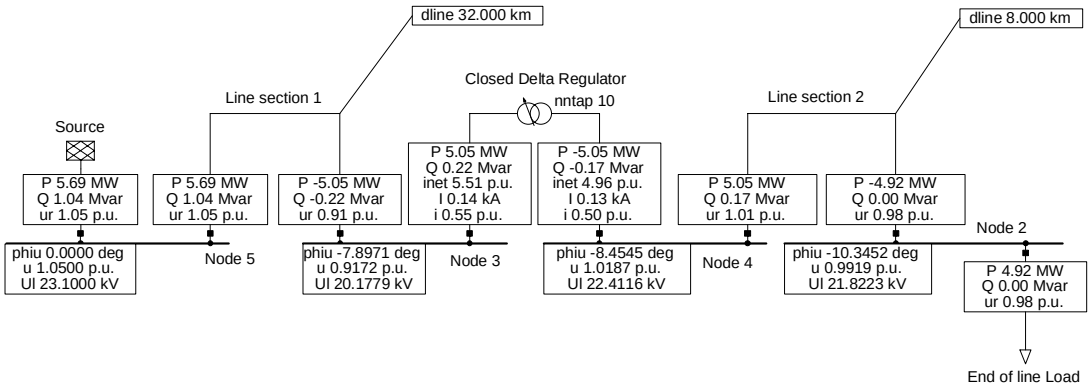


Figure 5-7: PowerFactory simulation

The SVR placement in the PowerFactory simulation is controlled from a DPL (DigSilent Programming Language) program developed for this purpose. The DPL code of the simulation is given in Addendum G. The details of the network modeled are given in Table 8.

Table 8: Network details

Network	22 kV radial
Source	1.05 p.u.
Backbone	40 km Hare conductor
Load type	Constant impedance load
Load size	4 MVA

The network data represents a long, lightly loaded reticulation line typical of the Eastern Cape. The data for the SVR used in the placement optimisation is given in Table 9.

Table 9: SVR details

Compensator	Open delta SVR
Voltage regulation	0% to 10%
Operation	Automatic
Placement	Between 0 km and 40 km

This data represents a typical open delta SVR. The optimisation program will find the optimal SVR voltage ratio and SVR placement to optimise the objective function inside the feasible variable space. The feasible variable space is defined by the active and reactive power flow equations and constrained by the voltage limits assigned to the busses. However,

for the purpose of this study the bus voltage limits were not applied, as they would not affect the results.

A special graphical user interface was developed to simplify the task of defining the objective function. The user is asked to enter the desired transformer primary voltage as well as the desired end of line voltage. Both these quantities must be entered in PU. This graphical user interface is shown in Figure 5-8.

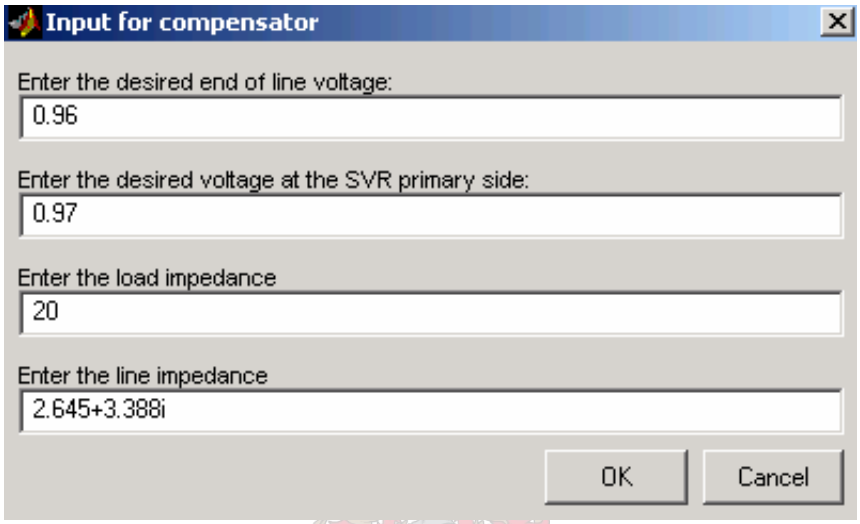


Figure 5-8: Input window for network parameters

Results obtained from the PowerFactory simulations were used to define the optimisation objective function via the program graphical user interface. The results of the optimisation are shown with the PowerFactory results in Table 10.

Table 10: Results from SVR optimisation

Power Factory				Optimisation program	
Distance (%)	SVR tap	V Prim (p.u.)	V Load (p.u.)	Distance (%)	SVR VR (p.u.)
20	0	1.0216	0.9159	19.9672	1.0000
20	3	1.0198	0.9426	20.0071	0.9700
20	6	1.0179	0.9709	19.9854	0.9400
20	10	1.0150	1.0112	20.0180	0.9000
50	0	0.9803	0.9159	50.0320	1.0000
50	3	0.9761	0.9402	50.0337	0.9701
50	6	0.9715	0.9567	51.1090	0.9502
50	10	0.9647	1.0015	50.0780	0.8999
80	0	0.9410	0.9159	80.0326	1.0001
80	3	0.9346	0.9378	79.9881	0.9698
80	6	0.9275	0.9604	80.0282	0.9400
80	10	0.9172	0.9919	79.3390	0.9006

As can be seen from Table 10 the optimisation algorithm was able to optimise the objective function by matching the voltage at the SVR primary side and at the end of the line simultaneously. This means the function value of the objective function was zero every time, which is obviously the minimum possible value. It is therefore concluded that the SQP method used in the SVR placement program is able to solve the non-linear constraint optimisation problem of placing a single SVR on a radial line.

5.4 CASE STUDY 3: 11 KV STEP VOLTAGE REGULATOR PLACEMENT

This case study is done to further show the capabilities of the optimisation program developed to place a SVR. The program is tested with a multiple bus network to show that the SQP method used can handle large constraint, non-linear problems. The non linear optimisation problem in this case study consists of 50 variables and 102 inequality constraints and 48 equality constraints. The objective function of the optimisation process in this study has been set to maximise the end of line voltage at bus 24. This optimisation is not realistic in terms of Eskom practice, but it is sufficient for this illustrative purpose. A more realistic objective function is to optimise both the SVR primary side voltage as well as the end of line voltage [B45].

A closed delta SVR is placed on the 11 kV Kwaaihoek – Whitney network near Grahamstown in the Eastern Cape. The geographical layout of the network is shown in Figure 5-9. This figure also shows the location of the existing SVR at structure KW-WN-104. To simplify the study, only the backbone network is considered. All spurs are represented as lumped loads on the backbone network. As will be seen from the results, this is a valid simplification.

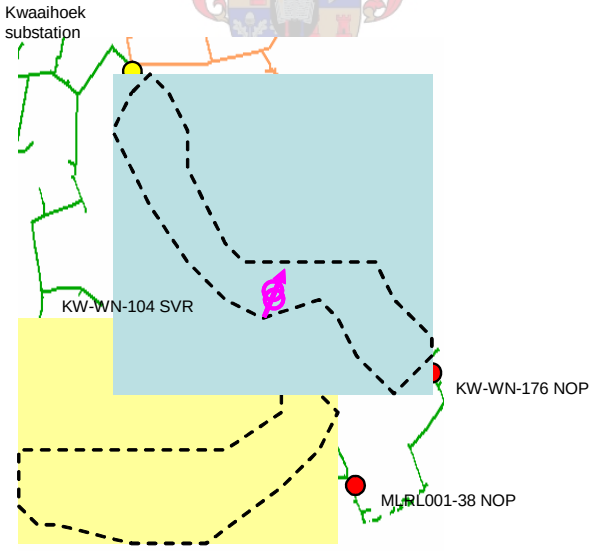


Figure 5-9: Kwaaihoek-Whitney network

The simplified radial network (without the SVR) as used in this case study is shown in Figure 5-10.

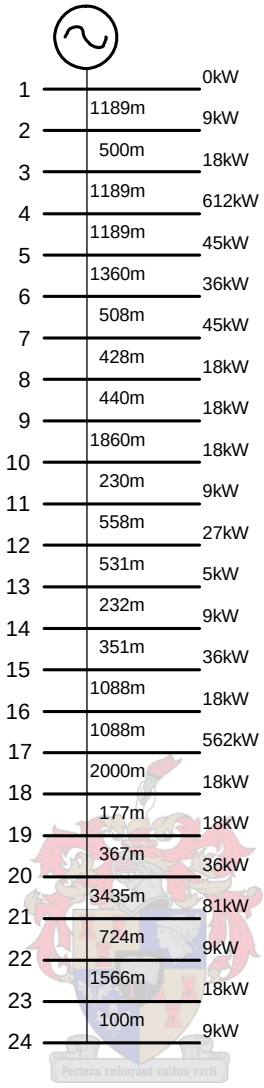


Figure 5-10: Kwaaihoek-Whitney single line diagram

The network consists of 24 busses with a total load of 1674 kVA at 0.98 pf. This loading has been determined from the 2006 load forecast for this network. The total line length of the backbone network is 21.11 km. For the simulation, the source voltage is regulated at 1.03 p.u., and the loads are modelled as constant power loads. This is a realistic model that compares well with the official Eskom ReticMaster model used to design this network.

The voltage profile along this simplified radial network is given by Figure 5-11. As can be seen the end of the line voltage is 91%, which is slightly lower than the specified minimum voltage used when placing an SVR. The branch loading of the network is shown in Figure 5-12. As can be seen, the total load supplied by the source is 1770 kVA.

Bus #	Voltage Mag(pu)	Voltage Ang(deg)	Generation P (MW)	Generation Q (MVAR)	Load P (MW)	Load Q (MVAR)
1	1.030	0.000	1.77	0.40	-	-
2	1.014	-0.243	-	-	0.01	0.00
3	1.007	-0.347	-	-	0.02	0.00
4	0.991	-0.597	-	-	0.60	0.12
5	0.981	-0.758	-	-	0.04	0.01
6	0.969	-0.938	-	-	0.04	0.01
7	0.965	-1.004	-	-	0.04	0.01
8	0.962	-1.058	-	-	0.02	0.00
9	0.959	-1.112	-	-	0.02	0.00
10	0.945	-1.340	-	-	0.02	0.00
11	0.944	-1.368	-	-	0.01	0.00
12	0.940	-1.436	-	-	0.03	0.01
13	0.936	-1.499	-	-	0.01	0.00
14	0.934	-1.526	-	-	0.01	0.00
15	0.932	-1.568	-	-	0.04	0.01
16	0.925	-1.691	-	-	0.02	0.00
17	0.918	-1.814	-	-	0.55	0.11
18	0.915	-1.871	-	-	0.02	0.00
19	0.915	-1.876	-	-	0.02	0.00
20	0.914	-1.884	-	-	0.04	0.01
21	0.911	-1.946	-	-	0.08	0.02
22	0.910	-1.950	-	-	0.01	0.00
23	0.910	-1.956	-	-	0.02	0.00
24	0.910	-1.956	-	-	0.01	0.00
		Total:	1.77	0.40	1.64	0.33

Figure 5-11: Kwaaihoek-Whitney voltage profile

From Bus	To Bus	From Bus Injection P (MW)	From Bus Injection Q (MVAR)	To Bus Injection P (MW)	To Bus Injection Q (MVAR)	Loss (I^2 * Z) P (MW)	Loss (I^2 * Z) Q (MVAR)
1	2	1.77	0.40	-1.74	-0.39	0.026	0.01
2	3	1.73	0.39	-1.72	-0.38	0.011	0.01
3	4	1.71	0.38	-1.68	-0.36	0.025	0.01
4	5	1.08	0.24	-1.07	-0.24	0.011	0.01
5	6	1.03	0.23	-1.01	-0.22	0.011	0.01
6	7	0.98	0.21	-0.98	-0.21	0.004	0.00
7	8	0.93	0.20	-0.93	-0.20	0.003	0.00
8	9	0.91	0.20	-0.91	-0.20	0.003	0.00
9	10	0.89	0.19	-0.88	-0.19	0.012	0.01
10	11	0.86	0.18	-0.86	-0.18	0.001	0.00
11	12	0.85	0.18	-0.85	-0.18	0.003	0.00
12	13	0.82	0.17	-0.82	-0.17	0.003	0.00
13	14	0.81	0.17	-0.81	-0.17	0.001	0.00
14	15	0.80	0.17	-0.80	-0.17	0.002	0.00
15	16	0.77	0.16	-0.76	-0.16	0.005	0.00
16	17	0.74	0.15	-0.74	-0.15	0.005	0.00
17	18	0.19	0.04	-0.19	-0.04	0.001	0.00
18	19	0.17	0.03	-0.17	-0.03	0.000	0.00
19	20	0.15	0.03	-0.15	-0.03	0.000	0.00
20	21	0.12	0.02	-0.11	-0.02	0.000	0.00
21	22	0.04	0.01	-0.04	-0.01	0.000	0.00
22	23	0.03	0.01	-0.03	-0.01	0.000	0.00
23	24	0.01	0.00	-0.01	-0.00	0.000	0.00
					Total:	0.129	0.07

Figure 5-12: Kwaaihoek-Whitney loading

The network model, as described above, has been used to place an SVR somewhere along this network by using the optimisation program. The first part of the program does a voltage ratio optimisation at each bus on the network. The result of the first part of the program is shown in Figure 5-13 and Figure 5-14. These results have been obtained by setting the bus voltage limits to ± 7%. Figure 5-13 shows that the first fifteen busses are suitable candidates for SVR placement. This agrees with the voltage profile of Figure 5-11, which shows a voltage of 93.2% at bus 15. The voltage at bus 16 is 92.5%, which is below the allowable minimum voltage of 93%. The minimum allowable voltage for all busses has been set to 93%. This is 3% higher than the absolute minimum allowable voltage of 90%. The extra 3% was added to cater for lower voltage levels in the future due to load growth. Using a minimum bus voltage of 93% makes the final SVR placement more robust. With the objective function set to maximise the end of the line voltage the optimal bus for SVR placement has been calculated as bus fifteen as shown by Figure 5-14. The voltage ratio, as determined by the optimisation process, is 0.8649. This shows that at this location the SVR is operating close to its maximum voltage boost capability of 15%.

```
================================================================================
|     Based on +/-7% Voltage limits                                            |
|     The allowable nodes for VR placement are:                                |
|     Node                 1                                                   |
|     Node                 2                                                   |
|     Node                 3                                                   |
|     Node                 4                                                   |
|     Node                 5                                                   |
|     Node                 6                                                   |
|     Node                 7                                                   |
|     Node                 8                                                   |
|     Node                 9                                                   |
|     Node                 10                                                  |
|     Node                 11                                                  |
|     Node                 12                                                  |
|     Node                 13                                                  |
|     Node                 14                                                  |
|     Node                 15                                                  |
================================================================================
```

Figure 5-13: Candidate busses for SVR placement

```
================================================================================
|     The optimal location for VR placement is around:                         |
|     Node                 15                                                  |
================================================================================
```

Figure 5-14: Result from part 1

The voltage profile and branch loading along the network with the SVR inserted at bus 15 is shown in Figure 5-15 and Figure 5-16. It is seen that the end of line voltage at bus 24 is

104.1%. The SVR has thus improved the end of line voltage by 10.041% while maintaining all bus voltages within the ± 7% limit. The voltage at the secondary side bus of the inserted SVR is not considered as a constraint in the optimisation process because there is no load connected at that bus and is thus not important from a quality of supply point of view.

Bus Data						
Bus #	Voltage Mag(pu)	Voltage Ang(deg)	Generation P (MW)	Generation Q (MVAR)	Load P (MW)	Load Q (MVAR)
1	1.030	0.000	1.78	0.42	-	-
2	1.014	-0.235	-	-	0.01	0.00
3	1.007	-0.335	-	-	0.02	0.00
4	0.991	-0.577	-	-	0.60	0.12
5	0.980	-0.729	-	-	0.04	0.01
6	0.969	-0.900	-	-	0.04	0.01
7	0.965	-0.962	-	-	0.04	0.01
8	0.961	-1.012	-	-	0.02	0.00
9	0.958	-1.063	-	-	0.02	0.00
10	0.944	-1.277	-	-	0.02	0.00
11	0.942	-1.303	-	-	0.01	0.00
12	0.938	-1.367	-	-	0.03	0.01
13	0.935	-1.425	-	-	0.01	0.00
14	0.933	-1.451	-	-	0.01	0.00
15	0.931	-1.490	-	-	0.04	0.01
16	1.054	-2.989	-	-	0.02	0.00
17	1.048	-3.083	-	-	0.55	0.11
18	1.045	-3.127	-	-	0.02	0.00
19	1.045	-3.131	-	-	0.02	0.00
20	1.045	-3.137	-	-	0.04	0.01
21	1.042	-3.184	-	-	0.08	0.02
22	1.042	-3.187	-	-	0.01	0.00
23	1.041	-3.192	-	-	0.02	0.00
24	1.041	-3.193	-	-	0.01	0.00
25	1.060	-2.894	-	-	-	-
		Total:	1.78	0.42	1.64	0.33

Figure 5-15: Voltage profile with SVR inserted at bus 15

Branch Data							
From Bus	To Bus	From Bus Injection P (MW)	From Bus Injection Q (MVAR)	To Bus Injection P (MW)	To Bus Injection Q (MVAR)	Loss (I^2 * Z) P (MW)	Loss (I^2 * Z) Q (MVAR)
1	2	1.78	0.42	-1.75	-0.41	0.027	0.01
2	3	1.74	0.41	-1.73	-0.40	0.011	0.01
3	4	1.71	0.40	-1.69	-0.38	0.026	0.01
4	5	1.09	0.26	-1.08	-0.26	0.011	0.01
5	6	1.03	0.25	-1.02	-0.24	0.011	0.01
6	7	0.98	0.23	-0.98	-0.23	0.004	0.00
7	8	0.94	0.22	-0.93	-0.22	0.003	0.00
8	9	0.92	0.22	-0.91	-0.22	0.003	0.00
9	10	0.90	0.21	-0.88	-0.21	0.012	0.01
10	11	0.87	0.20	-0.86	-0.20	0.001	0.00
11	12	0.86	0.20	-0.85	-0.20	0.003	0.00
12	13	0.83	0.19	-0.82	-0.19	0.003	0.00
13	14	0.82	0.19	-0.82	-0.19	0.001	0.00
14	15	0.81	0.19	-0.80	-0.19	0.002	0.00
16	17	0.74	0.15	-0.74	-0.15	0.004	0.00
17	18	0.19	0.04	-0.19	-0.04	0.000	0.00
18	19	0.17	0.03	-0.17	-0.03	0.000	0.00
19	20	0.15	0.03	-0.15	-0.03	0.000	0.00
20	21	0.11	0.02	-0.11	-0.02	0.000	0.00
21	22	0.04	0.01	-0.04	-0.01	0.000	0.00
22	23	0.03	0.01	-0.03	-0.01	0.000	0.00
23	24	0.01	0.00	-0.01	-0.00	0.000	0.00
25	16	0.76	0.16	-0.76	-0.16	0.004	0.00
15	25	0.77	0.18	-0.76	-0.16	0.007	0.02
				Total:		0.135	0.09

Figure 5-16: Load flow with SVR at bus 15

The optimisation process is then continued by considering the placement of an SVR somewhere on the branch to the source and load side of the bus that has been identified as optimal. This happens during the first part of the program, namely between bus 14 and bus 15 or between bus 15 and bus 16. The results of the source side placement are shown in Figure 5-17. As can be seen, the constraint in this placement is the voltage at bus 16. This is expected. The voltage limits on the primary and secondary side of the SVR are not taken as constraints in this optimisation process. This is evident from the voltage magnitude at bus 26 (which is the secondary side bus of the SVR) being 107.1%, which is slightly higher than the allowable 107%. The results of this optimisation process are given in Figure 5-18. In this part of the program both the SVR placement as well as voltage ratio are taken as optimisation variables. The objective function is again taken as to maximise the end of line voltage at bus 24. As can be seen the SVR has been placed at the end of the line segment between bus 15 and 1, namely at 99% of the total line length and with a voltage ratio of 0.86. The end of line voltage achieved by this placement is 105.76%, which is higher than the 104.1% obtained in the discrete optimisation of part one.

```
================================================================================
|   Nodal Voltages:                                                            |
|   Node  2       1.0140                                                       |
|   Node  3       1.0073                                                       |
|   Node  4       0.9916                                                       |
|   Node  5       0.9816                                                       |
|   Node  6       0.9706                                                       |
|   Node  7       0.9666                                                       |
|   Node  8       0.9634                                                       |
|   Node  9       0.9602                                                       |
|   Node 10       0.9469                                                       |
|   Node 11       0.9452                                                       |
|   Node 12       0.9414                                                       |
|   Node 13       0.9378                                                       |
|   Node 14       0.9363                                                       |
|   Node 15       0.9339                                                       |
|   Node 16       1.0700                                                       |
|   Node 17       1.0642                                                       |
|   Node 18       1.0616                                                       |
|   Node 19       1.0613                                                       |
|   Node 20       1.0609                                                       |
|   Node 21       1.0581                                                       |
|   Node 22       1.0579                                                       |
|   Node 23       1.0576                                                       |
|   Node 24       1.0576                                                       |
|   Node 25       0.9234                                                       |
|   Node 26       1.0701                                                       |
================================================================================
```

Figure 5-17: SVR on the load side of optimal bus

```
================================================================================
|   Optimisation variables:                                                    |
|   Placement in %:   0.99                                                     |
|   VR Voltage Ratio: 0.86                                                     |
================================================================================
```

Figure 5-18: SVR placement on the load side of optimal bus

The result of the source side placement is given in Figure 5-19. Again the placement is at the end of the line segment and the SVR voltage ratio is close to optimal at 14%.

```
================================================================================
|   Optimisation variables:                                                    |
|   Placement in %:   0.99                                                     |
|   VR Voltage Ratio: 0.86                                                     |
================================================================================
```

Figure 5-19: SVR placement on the source side of optimal bus

From these three optimisation routines the optimal placement of the SVR is that calculated in the load side placement routine as given by Figure 5-18.

The program developed is able to place a SVR on a radial network to optimise the end of line voltage while maintaining acceptable voltage levels at all the busses on the network. The results obtained from the above study compare very well with the actual placement of the SVR on the Kwaaihoek-Whitney network at structure KW-WN-104. This structure is 10.66

km down the line compared to the 11.65 km obtained in the optimisation study (99% along the branch between busses 15 and 16). The case study shows that the optimisation program is able to solve relatively large, non-linear constraint problems and it is also shown that it does give realistic solutions.

5.5 CASE STUDY 4: LOW VOLTAGE ELECTRONIC VOLTAGE REGULATOR PLACEMENT

The purpose of this case study is to show how the genetic optimisation program can be used to place EVRs on low voltage networks. The main purpose of applying EVRs is to maximise the length of LV feeders in an effort to make electrification more cost effective. It has already been proven that the use of 5 kVA EVRs is a feasible means of compensation on deep rural electrification applications with population densities of less than 200 stands per square kilometre and using an ADMD of 0.4 kW [B38].

The network that will be analysed is the Thaba Lesobo network, which is an electrification project in the Aliwal North area in the Eastern Cape. Two transformer zones of this electrification project are used in this case study as a basis to place EVRs. The Thaba Lesobo electrification project has an estimated population density of 400 stands per square kilometre. It is further assumed that all loads are constant impedance loads and at unity power factor. The conductor used is a 70 mm^2 bundle conductor. The second part of the case study analyses a deep rural application by stretching the same network and scaling down the loading. The loading is scaled down by decreasing the population density as well as decreasing the ADMD. The above load modelling is based on a simplified deterministic method and can only be used for illustrative purposes. To ensure accurate LV designs a more complete stochastic method should be used.

Figure 5-20 shows how these two transformer zones are joint to form one extended transformer zone. The network configuration in Figure 5-20 will be used throughout this case study.

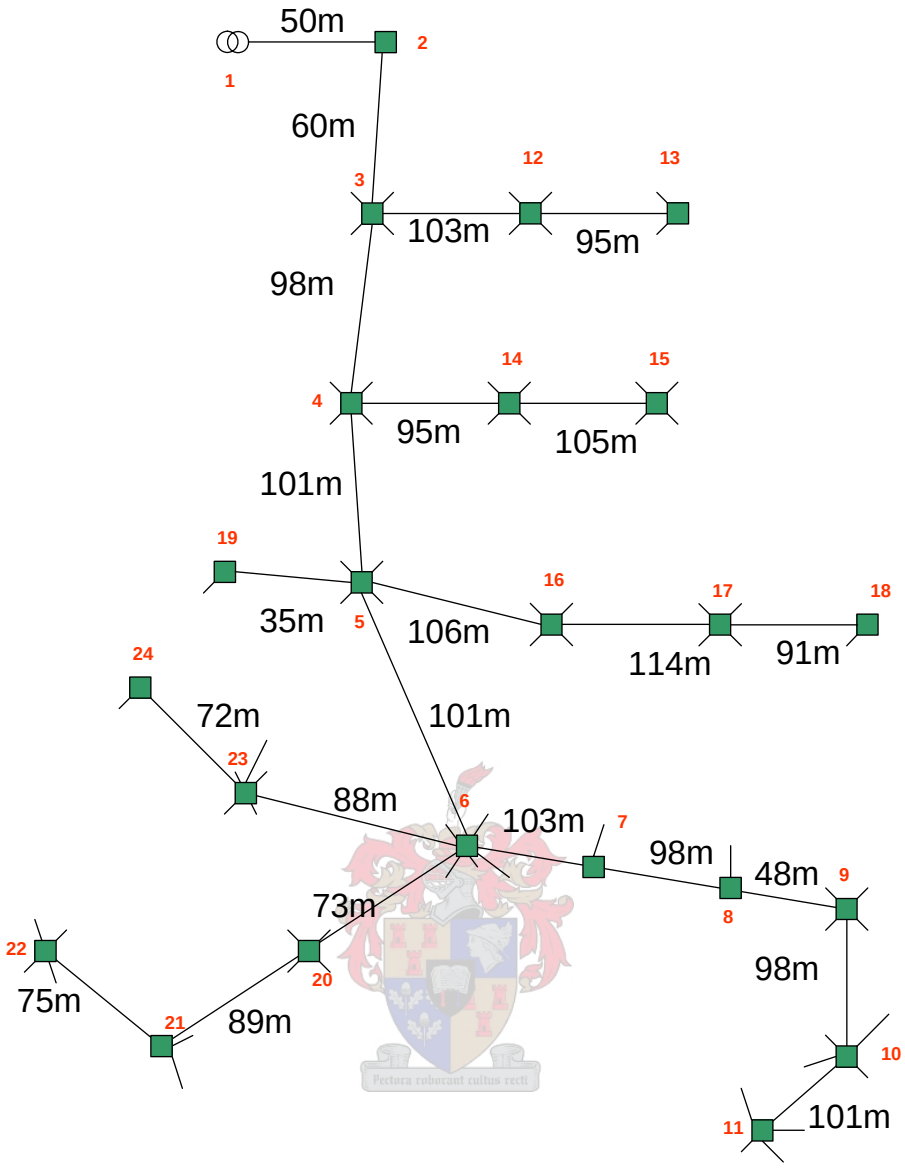


Figure 5-20: Transformer zones 28 and 29 combined

The load flow results of the Thaba Lesobo combined transformer zones 28 and 29 with 50% load density, are shown in Figure 5-21 and Figure 5-22. As can be seen, the voltage reaches 90% at node 8 on the backbone of the network. The loading on the backbone branch from node 8 to node 9 is 1.93 kVA, which is far below the rating of the EVR. It should therefore be possible to apply a single EVR at node 8 to regulate the voltage at nodes 9, 10 and 11. However, when the optimisation routine is run, two EVRs are installed. This is because the voltage at node 7 drops below 90% due to the increase in current drawn by the constant impedance loads. The voltages at nodes 21 and 22 also drop below 90% for the same reason. The two EVRs are installed on the branch between node 6 and 7 and the other one on the

branch between node 20 and 21 as a result. The voltage profile and branch loading results of this EVR placement are shown in Figure 5-23 and Figure 5-24.

Bus Data						
Bus #	Voltage Mag(pu)	Ang(deg)	Generation P (kW)	Q (kVAR)	Load P (kW)	Q (kVAR)
1	1.000	0.000	13.09	0.14	-	-
2	0.986	-0.106	-	-	-	-
3	0.968	-0.238	-	-	-	-
4	0.944	-0.428	-	-	-	-
5	0.925	-0.590	-	-	-	-
6	0.911	-0.708	-	-	-	-
7	0.905	-0.757	-	-	-	-
8	0.900	-0.801	-	-	-	-
9	0.898	-0.820	-	-	-	-
10	0.895	-0.847	-	-	-	-
11	0.893	-0.861	-	-	-	-
12	0.966	-0.259	-	-	-	-
13	0.965	-0.266	-	-	-	-
14	0.941	-0.454	-	-	-	-
15	0.940	-0.468	-	-	-	-
16	0.921	-0.623	-	-	-	-
17	0.919	-0.642	-	-	-	-
18	0.918	-0.645	-	-	-	-
19	0.925	-0.591	-	-	-	-
20	0.908	-0.733	-	-	-	-
21	0.906	-0.752	-	-	-	-
22	0.904	-0.762	-	-	-	-
23	0.909	-0.727	-	-	-	-
24	0.908	-0.729	-	-	-	-
		Total:	13.09	0.14	0.00	0.00

Figure 5-21: Thaba Lesobo zone 28 and 29 half density voltage profile

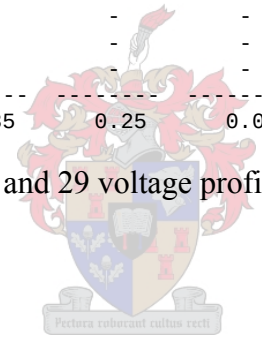
Branch Data							
From Bus	To Bus	From Bus Injection P (kW)	From Bus Injection Q (kVAR)	To Bus Injection P (kW)	To Bus Injection Q (kVAR)	Loss (I^2 * Z) P (kW)	Loss (I^2 * Z) Q (kVAR)
1	2	13.09	0.14	-12.91	-0.11	0.188	0.03
2	3	12.91	0.11	-12.68	-0.08	0.226	0.03
3	4	10.81	0.08	-10.54	-0.04	0.268	0.04
4	5	8.41	0.04	-8.23	-0.02	0.176	0.02
5	6	5.85	0.02	-5.76	-0.01	0.089	0.01
6	7	2.29	0.01	-2.27	-0.00	0.014	0.00
7	8	2.11	0.00	-2.10	-0.00	0.012	0.00
8	9	1.93	0.00	-1.93	-0.00	0.005	0.00
9	10	1.28	0.00	-1.28	-0.00	0.004	0.00
10	11	0.64	0.00	-0.64	-0.00	0.001	0.00
3	12	1.12	0.00	-1.12	-0.00	0.003	0.00
12	13	0.37	0.00	-0.37	-0.00	0.000	0.00
4	14	1.42	0.00	-1.42	-0.00	0.005	0.00
14	15	0.71	0.00	-0.71	-0.00	0.001	0.00
5	19	0.17	0.00	-0.17	-0.00	0.000	0.00
5	16	1.53	0.00	-1.52	-0.00	0.006	0.00
16	17	0.85	0.00	-0.84	-0.00	0.002	0.00
17	18	0.17	0.00	-0.17	-0.00	0.000	0.00
6	20	1.65	0.00	-1.64	-0.00	0.005	0.00
20	21	0.99	0.00	-0.98	-0.00	0.002	0.00
21	22	0.66	0.00	-0.65	-0.00	0.001	0.00
6	23	0.99	0.00	-0.99	-0.00	0.002	0.00
23	24	0.17	0.00	-0.16	-0.00	0.000	0.00
				Total:		1.010	0.14

Figure 5-22: Thaba Lesobo zone 28 and 29 half density branch loading

As can be seen from Figure 5-23, the optimisation algorithm was able to keep all nodal voltages within the nodal voltage limits that are set as the constraints of the genetic optimisation algorithm. Any individual that produces a nodal voltage outside the voltage limits, causes the entire population to be marked as infeasible. The genetic algorithm will replace the infeasible individual until all the individuals of the population are feasible. At that stage the population becomes feasible as a whole and the algorithm continues. The lowest voltage occurs at node 20, which is 90%. This is still allowable since the lower voltage limit has been implemented as greater than or equal to 90%.

Bus Data						
Bus #	Voltage Mag(pu)	Ang(deg)	Generation P (kW)	Q (kVAR)	Load P (kW)	Q (kVAR)
1	1.000	0.000	13.85	0.25	-	-
2	0.985	-0.106	-	-	-	-
3	0.967	-0.237	-	-	-	-
4	0.941	-0.427	-	-	-	-
5	0.919	-0.590	-	-	-	-
6	0.904	-0.710	-	-	-	-
7	0.997	-2.337	-	-	-	-
8	0.992	-2.381	-	-	-	-
9	0.989	-2.401	-	-	-	-
10	0.986	-2.428	-	-	-	-
11	0.984	-2.442	-	-	-	-
12	0.964	-0.259	-	-	-	-
13	0.963	-0.265	-	-	-	-
14	0.938	-0.453	-	-	-	-
15	0.936	-0.468	-	-	-	-
16	0.916	-0.623	-	-	-	-
17	0.913	-0.643	-	-	-	-
18	0.913	-0.646	-	-	-	-
19	0.919	-0.591	-	-	-	-
20	0.900	-0.737	-	-	-	-
21	0.998	-1.441	-	-	-	-
22	0.997	-1.452	-	-	-	-
23	0.902	-0.729	-	-	-	-
24	0.901	-0.731	-	-	-	-
25	1.004	-2.289	-	-	-	-
26	1.000	-1.423	-	-	-	-
		Total:	13.85	0.25	0.00	0.00

Figure 5-23: Zones 28 and 29 voltage profile after EVR placement



Branch Data							
From Bus	To Bus	From Bus Injection P (kW)	From Bus Injection Q (kVAR)	To Bus Injection P (kW)	To Bus Injection Q (kVAR)	Loss (I^2 * Z) P (kW)	Loss (I^2 * Z) Q (kVAR)
1	2	13.85	0.25	-13.64	-0.22	0.210	0.03
2	3	13.64	0.22	-13.39	-0.19	0.252	0.03
3	4	11.52	0.19	-11.22	-0.14	0.306	0.04
4	5	9.10	0.14	-8.89	-0.12	0.207	0.03
5	6	6.53	0.11	-6.42	-0.10	0.112	0.02
7	8	2.56	0.00	-2.54	-0.00	0.014	0.00
8	9	2.35	0.00	-2.34	-0.00	0.006	0.00
9	10	1.56	0.00	-1.55	-0.00	0.005	0.00
10	11	0.78	0.00	-0.77	-0.00	0.001	0.00
3	12	1.12	0.00	-1.11	-0.00	0.003	0.00
12	13	0.37	0.00	-0.37	0.00	0.000	0.00
4	14	1.41	0.00	-1.41	-0.00	0.005	0.00
14	15	0.70	0.00	-0.70	-0.00	0.001	0.00
5	19	0.17	0.00	-0.17	-0.00	0.000	0.00
5	16	1.51	0.00	-1.51	-0.00	0.006	0.00
16	17	0.84	0.00	-0.83	-0.00	0.002	0.00
17	18	0.17	0.00	-0.17	-0.00	0.000	0.00
6	20	1.85	0.02	-1.85	-0.01	0.007	0.00
21	22	0.80	0.00	-0.79	0.00	0.001	0.00
6	23	0.98	0.00	-0.98	-0.00	0.002	0.00
23	24	0.16	0.00	-0.16	-0.00	0.000	0.00
25	7	2.78	0.01	-2.76	-0.00	0.017	0.00
6	25	2.78	0.08	-2.78	-0.01	0.000	0.08
26	21	1.20	0.00	-1.19	-0.00	0.003	0.00
20	26	1.20	0.01	-1.20	-0.00	0.000	0.01
					Total:	1.162	0.25

Figure 5-24: Zones 28 and 29 branch flows after EVR placement

To simulate a more realistic deep rural network the Thaba Lesobo network is stretched and the loading decreased. This is done by doubling all conductor lengths and by halving the number of stands and using an ADMD of 0.2 kVA. The network configuration of this network is shown in Figure 5-25. The voltage profile and branch loadings of this deep rural network are shown in Figure 5-26 and Figure 5-27. As can be seen the backbone voltage reaches 90% at node 10. The loading on the branch between nodes 10 and 11 is 0.32 kVA. From this information it appears that a single EVR on the branch between node 10 and 11 would result in all nodal voltages to be within limit and the EVR rating to be less than 5 kVA.

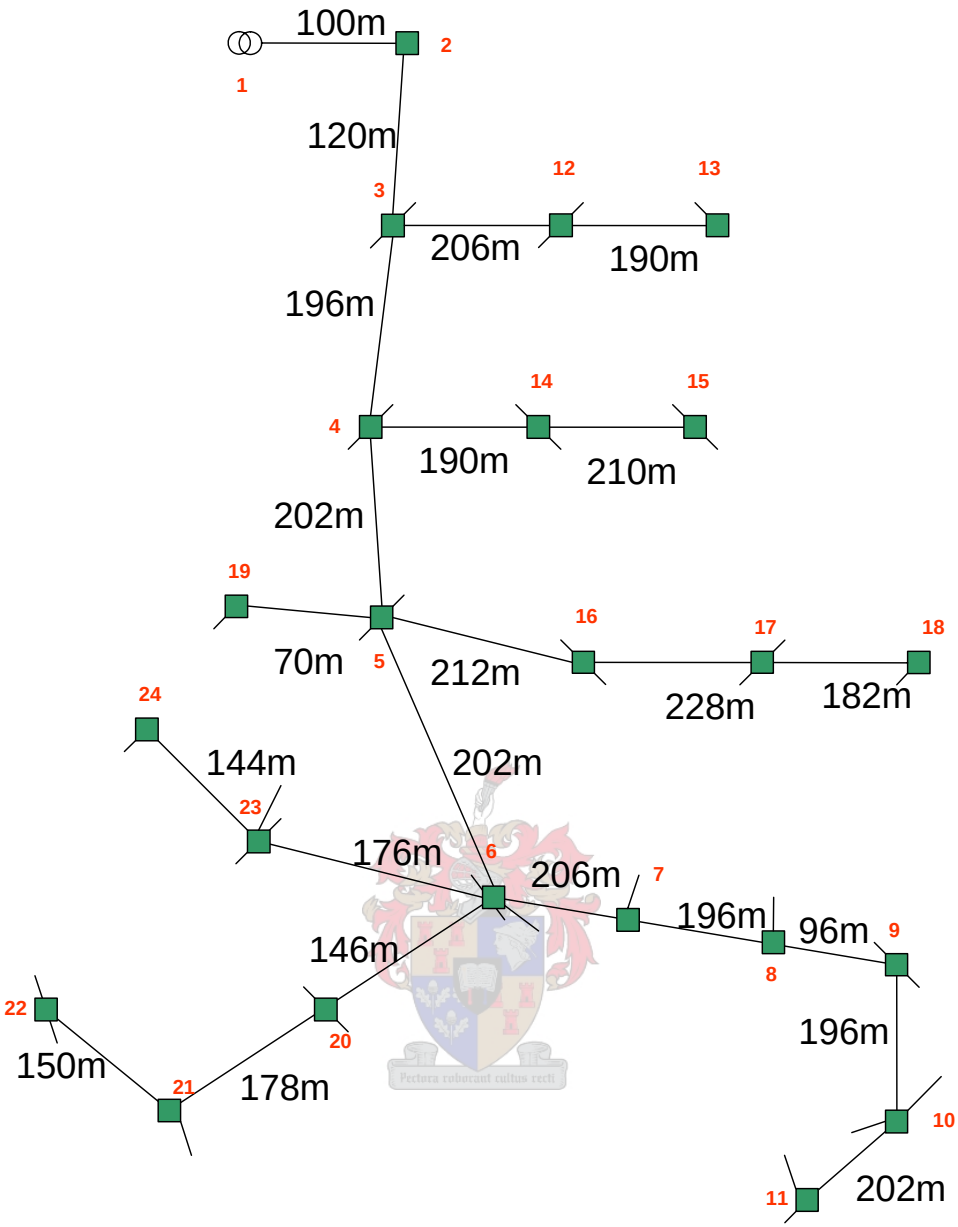


Figure 5-25: Trfr zones 28 and 29 stretched and deloaded

Bus Data						
Bus #	Voltage Mag(pu)	Voltage Ang(deg)	Generation P (kW)	Generation Q (kVAR)	Load P (kW)	Load Q (kVAR)
1	1.000	0.000	5.54	0.06	-	-
2	0.988	-0.090	-	-	-	-
3	0.973	-0.200	-	-	-	-
4	0.952	-0.367	-	-	-	-
5	0.933	-0.525	-	-	-	-
6	0.917	-0.653	-	-	-	-
7	0.911	-0.709	-	-	-	-
8	0.905	-0.756	-	-	-	-
9	0.903	-0.775	-	-	-	-
10	0.900	-0.803	-	-	-	-
11	0.898	-0.817	-	-	-	-
12	0.972	-0.214	-	-	-	-
13	0.971	-0.221	-	-	-	-
14	0.950	-0.380	-	-	-	-
15	0.949	-0.387	-	-	-	-
16	0.930	-0.547	-	-	-	-
17	0.928	-0.562	-	-	-	-
18	0.927	-0.569	-	-	-	-
19	0.932	-0.527	-	-	-	-
20	0.914	-0.683	-	-	-	-
21	0.911	-0.708	-	-	-	-
22	0.910	-0.718	-	-	-	-
23	0.915	-0.671	-	-	-	-
24	0.915	-0.676	-	-	-	-
		Total:	5.54	0.06	0.00	0.00

Figure 5-26: Scaled zones 28 and 29 voltage profile

Branch Data							
From Bus	To Bus	From Bus Injection P (kW)	From Bus Injection Q (kVAR)	To Bus Injection P (kW)	To Bus Injection Q (kVAR)	Loss (I^2 * Z) P (kW)	Loss (I^2 * Z) Q (kVAR)
1	2	5.54	0.06	-5.47	-0.05	0.067	0.01
2	3	5.47	0.05	-5.39	-0.04	0.081	0.01
3	4	4.82	0.04	-4.72	-0.02	0.105	0.01
4	5	4.17	0.02	-4.09	-0.01	0.085	0.01
5	6	3.22	0.01	-3.17	-0.00	0.053	0.01
6	7	1.32	0.00	-1.31	-0.00	0.009	0.00
7	8	1.15	0.00	-1.14	-0.00	0.007	0.00
8	9	0.98	0.00	-0.98	-0.00	0.002	0.00
9	10	0.65	0.00	-0.65	-0.00	0.002	0.00
10	11	0.32	0.00	-0.32	-0.00	0.001	0.00
3	12	0.38	0.00	-0.38	-0.00	0.001	0.00
12	13	0.19	0.00	-0.19	-0.00	0.000	0.00
4	14	0.36	0.00	-0.36	-0.00	0.001	0.00
14	15	0.18	0.00	-0.18	-0.00	0.000	0.00
5	19	0.17	0.00	-0.17	-0.00	0.000	0.00
5	16	0.52	0.00	-0.52	-0.00	0.001	0.00
16	17	0.35	0.00	-0.34	-0.00	0.001	0.00
17	18	0.17	0.00	-0.17	-0.00	0.000	0.00
6	20	1.00	0.00	-1.00	-0.00	0.004	0.00
20	21	0.67	0.00	-0.66	-0.00	0.002	0.00
21	22	0.33	0.00	-0.33	-0.00	0.000	0.00
6	23	0.50	0.00	-0.50	-0.00	0.001	0.00
23	24	0.17	0.00	-0.17	-0.00	0.000	0.00
					Total:	0.424	0.06

Figure 5-27: Scaled zones 28 and 29 branch flows

However, when the optimisation routine is however executed, the EVR is inserted on the branch between node 5 and 6. This shows the placement capability of the program very well as it placed the single EVR at an optimal location. The voltage profile after EVR placement is shown in Figure 5-28 and the branch loading in Figure 5-29. The placement reduced the voltage at node 5 from 93.3% to 92.2%, which is still above the allowable 90%. From Figure 5-29 it can be seen that the loading on the branch between nodes 5 and 6, where the EVR is installed, is 3.88 kVA, which is well below the maximum EVR rating of 5 kVA.

Bus Data						
Bus #	Voltage Mag(pu)	Voltage Ang(deg)	Generation P (kW)	Generation Q (kVAR)	Load P (kW)	Load Q (kVAR)
1	1.000	0.000	6.27	0.22	-	-
2	0.986	-0.082	-	-	-	-
3	0.970	-0.184	-	-	-	-
4	0.945	-0.337	-	-	-	-
5	0.922	-0.481	-	-	-	-
6	1.007	-2.730	-	-	-	-
7	1.000	-2.786	-	-	-	-
8	0.994	-2.832	-	-	-	-
9	0.991	-2.852	-	-	-	-
10	0.988	-2.879	-	-	-	-
11	0.986	-2.893	-	-	-	-
12	0.968	-0.198	-	-	-	-
13	0.967	-0.205	-	-	-	-
14	0.943	-0.350	-	-	-	-
15	0.943	-0.357	-	-	-	-
16	0.920	-0.503	-	-	-	-
17	0.918	-0.519	-	-	-	-
18	0.917	-0.525	-	-	-	-
19	0.922	-0.483	-	-	-	-
20	1.003	-2.760	-	-	-	-
21	1.000	-2.785	-	-	-	-
22	0.999	-2.795	-	-	-	-
23	1.005	-2.748	-	-	-	-
24	1.004	-2.753	-	-	-	-
25	1.024	-2.601	-	-	-	-
	Total:		6.27	0.22	0.00	0.00

Figure 5-28: Scaled zones 28 and 29 voltage profile after EVR placement

Branch Data							
From Bus	To Bus	From Bus Injection P (kW)	From Bus Injection Q (kVAR)	To Bus Injection P (kW)	To Bus Injection Q (kVAR)	Loss (I^2 * Z) P (kW)	Loss (I^2 * Z) Q (kVAR)
1	2	6.27	0.22	-6.19	-0.21	0.086	0.01
2	3	6.19	0.21	-6.08	-0.19	0.104	0.01
3	4	5.52	0.19	-5.38	-0.17	0.140	0.02
4	5	4.85	0.17	-4.73	-0.16	0.117	0.02
6	7	1.60	0.00	-1.58	-0.00	0.011	0.00
7	8	1.38	0.00	-1.38	-0.00	0.008	0.00
8	9	1.18	0.00	-1.18	-0.00	0.003	0.00
9	10	0.78	0.00	-0.78	-0.00	0.003	0.00
10	11	0.39	0.00	-0.39	0.00	0.001	0.00
3	12	0.38	0.00	-0.37	-0.00	0.001	0.00
12	13	0.19	0.00	-0.19	-0.00	0.000	0.00
4	14	0.36	0.00	-0.36	-0.00	0.001	0.00
14	15	0.18	0.00	-0.18	0.00	0.000	0.00
5	19	0.17	0.00	-0.17	0.00	0.000	0.00
5	16	0.51	0.00	-0.51	-0.00	0.001	0.00
16	17	0.34	0.00	-0.34	-0.00	0.001	0.00
17	18	0.17	0.00	-0.17	-0.00	0.000	0.00
6	20	1.21	0.00	-1.20	-0.00	0.005	0.00
20	21	0.80	0.00	-0.80	-0.00	0.002	0.00
21	22	0.40	0.00	-0.40	0.00	0.001	0.00
6	23	0.61	0.00	-0.61	-0.00	0.001	0.00
23	24	0.20	0.00	-0.20	0.00	0.000	0.00
25	6	3.88	0.01	-3.82	-0.00	0.064	0.01
5	25	3.88	0.16	-3.88	-0.01	0.000	0.14
					Total:	0.549	0.22

Figure 5-29 Scaled zones 28 and 29 branch flows after EVR placement

This case study shows that the genetic algorithm used in the placement program is able to place a number of EVRs on a radial network so that the network voltages are within specified limits as well as keeping the EVR within its 5 kVA rating. The main limitation of the program is that it only places EVRs with voltage regulation set to 10% boost. To overcome this limitation, the program can be further developed to place EVRs of different voltage regulation settings. This enhancement can be implemented in the genetic algorithm in two ways. The first approach is to include a smooth optimisation routine as a sub routine to the genetic algorithm that optimises the voltage ratio of the EVR. The objective of this optimisation routine will depend on the specific application, but can be set to maximise the voltage profile of the entire network. The control variables of this optimisation problem are the voltage ratios of all the EVRs placed by the genetic algorithm outer loop. The constraints of the inner loop smooth optimisation would be the voltage limits of all the nodes as well as the rating of the EVRs. Such a smooth optimisation routine can be implemented by an SQP technique similar to the one used in the program developed for SVR placement.

Alternatively the genetic algorithm can be adapted to handle EVRs with different voltage regulation settings. This will necessitate significant changes to the existing algorithm. The

existing algorithm uses binary encoding with a *one* representing the presence of an EVR and a *zero* indicating that no EVR is installed at a specific node. With different types of EVRs this encoding cannot handle all the information. A more advanced encoding will have to be used such as value encoding. Using this encoding a *zero* can represent no EVR present, a *one* can represent an EVR with voltage regulation set to 10% and a *two* can represent an EVR with voltage regulation set to 20%. The rest of the algorithm will have to be adapted to handle this type of encoding.

It is very difficult to predict which of these two approaches will be more effective. The first approach using the smooth optimisation sub routine will probably be more reliable as it will consider the continuous range of voltage regulation of the EVR, namely 0 to 20% voltage boost. The disadvantage of this approach is that the smooth optimisation will slow down the algorithm.

The second approach that enhances the genetic algorithm to consider different EVRs with voltage regulation settings by using value encoding will probably be faster and more robust. The disadvantage of using this approach is that it might not always give a global optimal solution. This is, however, a known drawback of any genetic algorithm and can be controlled by customising the algorithm.

This concludes the discussion of the three case studies. It has been shown that PSAT is able to implement active power control on the 132 kV Eros-Zimbane-Pembroke network by installing a PST at Zimbane substation. The second case study uses the placement program for SVRs to first place an SVR on a simple radial network and then on a complex radial network. These two case studies show the ability of the program to handle multi objective problems as well as relatively large problems. The last case study shows how the EVR placement program is used to place a number of EVRs on a radial network by making use of a genetic algorithm. Chapter 6 and Chapter 7 conclude this thesis by proposing further development work and making final remarks.

CHAPTER 6 FUTURE WORK

This thesis has shown three applications of optimisation theory in power flow studies. Together with PSAT, two new programs have been developed to place SVRs and EVRs on different types of networks. Although the programs are functional they can be further enhanced. It is therefore suggested that the work on the application of optimisation techniques to place voltage regulators, be continued. The continuation of the work can be set out as follows.

The SVR placement program should be further enhanced to include user definable objective functions. This will make the program more flexible and enable the user to customise the placement optimisation according to specific network characteristics. A suitable graphical user interface should be developed that will allow easy access to all the optimisation variables.

The EVR placement program should be further enhanced to include different types of EVRs. EVRs of different through power and current capacity, as well as different voltage ratios, should be included. With the different types of EVRs, complex cost functions should also be included in the objective function as well as in the set of constraint functions.

The genetic algorithm of the EVR placement program should be further enhanced to improve the speed of the program. At the moment a large number of infeasible replacement individuals are created. The replacement function creates individuals with completely random genetic information. This can be improved upon by making the individual replacement function more intelligent. For instance, the replacement individual's genetic makeup can be based on the genetic material of other feasible individuals. Genetic variety can then be introduced by a mutation process. The improvement of the replacement process will vastly improve the efficiency of the program.

Another major improvement to the EVR placement program would be to have some intelligence in the formulation of the initial population. This can be achieved by allowing the user to construct the individuals of the initial population. The user might have a good idea of what would be the optimal placement of EVRs based on experience or prior knowledge of the network.

The application software developed in this thesis should be implemented in commercial software. The idea of optimal placement of voltage regulators as well as the optimal placement of a combination of voltage regulators and shunt capacitors, has already been suggested to DigSilent and will hopefully be included in a future version of Power Factory. This will be an addition to the already existing optimal capacitor placement functionality of Power Factory.

An effort must be made to keep up to date with the development of new optimisation methods and the impact they can have on applications in power system studies.

Further development work on the optimisation tools must consider integration with Eskom standards. Network planning and operation and quality of supply environments are structured to comply with certain standards. The standards with the most direct impact on voltage regulation are the NRS 048 [B34] and the Voltage Limits and Apportionment Standard [B35]. Another very important methodology that must be incorporated is that of Herman – Beta load modelling. The SVR and EVR placement programs use deterministic load modelling and do not cater for the stochastic nature of loads. The inclusion of Herman – Beta load modelling in the optimisation routines of the voltage regulator placement programs will result in more realistic results.

Further development of the EVR placement program that will greatly enhance its scope of application will be to include spatial layout to the design of new LV networks. The objective of such a program will be to minimise the total design cost of an electrification project. The total cost is made up of the transformer cost, the network cost and the EVR cost. The control variables will be the placement of EVRs, the placement of the transformers as well as technology selection and conductor type selection. Other variables that should be included in the objective functions of such optimisation programs should include reliability, performance, Quality of Supply parameters like Total Harmonic Distortion, as well as detailed cost functions.

The ultimate goal of developing optimisation software should be the automation of the network planning, strengthening and operational planning processes. The long-term goal that should be worked towards is the complete automation of all processes involving the optimal maintenance of a power system. This must include the planning, normal and abnormal operation, reliability, loss minimisation and security of a network.

CHAPTER 7 CONCLUSION

Mathematical optimisation is a very powerful tool for solving complex practical problems. A wide variety of techniques are available for solving different kinds of problems depending on the structure of the problem and the requirements for the solution for example, speed of solution and accuracy of solution. In the field of power system studies, only a handful of applications have been found and successfully solved using optimisation methods. These applications include generation scheduling and placement of shunt capacitor banks. The main objective of these optimisation routines is to minimise cost.

The compensator placement problem is especially difficult to solve with classical methods from Calculus because of the size of the problem, the non-linearity of the equations and the mixed continuous and discrete variables describing the problem. With the further development of PSAT as part of this thesis, it was realised that the compensator placement problem is suitable for being solved as an optimisation problem.

Three specific applications in Eskom Distribution were identified and focused on in this thesis. The placement of a PST, the placement of an SVR and the placement of multiple EVRs. Development of PSAT and two new programs, the SVR placement program and the EVR placement program, were used to address these three applications.

Because of the closed structure of commercially available power system simulation tools, the development work has been done in Matlab. An advantage of using Matab is the availability of general purpose optimisation functionality in the form of Toolboxes. The availability of these Toolboxes dramatically reduces time spent on developing the algorithms and functions for the programs presented in this thesis.

In order to make the functionality and techniques used in this thesis available to users in the industry (not only for academic use), it is necessary to communicate with the developers of commercial PSSTs (Power System Simulation Tools) with respect to the application and development of optimisation techniques. This will also prevent unnecessary duplication of research. To this extent, the idea of optimal voltage regulator placement has already been discussed with DigSilent GmbH. This will, however, have to be followed up with DigSilent GmbH to ensure implementation.

CHAPTER 8 REFERENCES

8.1 JOURNALS AND CONFERENCE PAPERS

[B1] F.J. Rossouw, 'Analysis Of Voltage Regulation And Network Support Technologies', University of Stellenbosch, Stellenbosch, 2000.

[B2] Mark S. Gockenbach, 'An active-set algorithm for convex QPs'.

[B3] H. Glavitsch, 'Optimal Power Flow Algorithms', Swiss Federal Institute of Technology, Switzerland.

[B4] H.W. Dommel, W.F. Tinney; Optimal Power Flow Solutions; IEEE Transactions on Power Apparatus and Systems, Vol. PAS-87, pp. 1866-1867, Oct. 1968.

[B5] R.B.Squires; Economic Dispatch of Generation Directly from Power System Voltages and Admittances', AIEE Trans. PAS Vol. 52. Part III (1961),pp. 1235-1244.

[B6] J.Carpentier; 'Application of Newton's Method to Load Flow Computations', Proc. PSCC 1, London (1963).

[B7] W.F. Tinney, J.W. Walker; 'Direct Solutions of Sparse Network Equations by Optimally Ordered Triangular Factorization', Proceedings of the IEEE, Vol. 55, No. 11, November 1967.

[B8] O.Alsac, B.Stott; 'Optimal Power Flow with Steady-State Security', IEEE Trans. PAS, Vol. 93. (1974), pp. 745-751.

[B9] J.D Weber, 'Implementation of a Newton-based optimal power flow into a power system simulation environment', B.S., University of Wisconsin - Platteville, 1995.

[B10] Deqiang Gan, 'A Transient Stability Constrained Optimal Power Flow', School of Electrical Engineering Cornell University, Ithaca, NY.

[B11] Zbigniew Michalcwics, 'Genetic Algorithms, Numerical Optimization, and Constraints', Department of Computer Science, University of North Carolina.

[B12] Benjamin Baran, 'Reactive Power Compensation using a Multi-objective Evolutionary Algorithm', 2001 IEEE Porto Power Tech Conference.

[B13] R. Hooshmand, 'Optimal Choice of Fixed and Switched Capacitors for Distribution Systems by Genetic Algorithm', Shahid Chamran University, Ahwaz, Iran.

[B14] A. Mendes, 'An evolutionary approach for capacitor placement in distribution networks', Campinas SP-Brazil.

[B15] Dulce F. Pires, 'A Tabu search Multi objective Approach to Capacitor Allocation in Radial Distribution Systems', July 2001 Campus do IPS, Setubal, Portugal.

[B16] Young-Jae Jeon, 'An efficient Simulated Annealing Algorithm for Network Regonfiguration in Large-Scale Distribution Systems', October 2002 IEEE Transactions on Power Delivery, Vol 17, No4.

[B17] Andrew Chipperfield, 'Genetic Algorithm Toolbox – User's Guide', University of Sheffield.

[B18] DigSilent document, 'Optimal Capacitor Placement'.

[B19] Navid Azizi, 'Automated Analog Circuit Design Using Genetic Algorithms', Department of Electrical and Computer Engineering University of Toronto.

[B20] IEEE Power Engineering Society, 'IEEE Guide for the Application, Specification, and Testing of Phase-Shifting Transformers'.

8.2 BOOKS AND MANUALS

[B21] N. G. Hingorani and L. Gyugi, '*Understanding FACTS'*, IEEE Press, New York, 2000.

[B22] J.D. Glover and M. Sarma, '*Power System Analysis and Design'*, PWS Publishing Company, Boston, 1994.

[B23] B. S. Gottfried, '*Introduction to optimization theory'*, Prentice Hall, New Jersey, 1973

[B24] D. G. Luenberger, '*Introduction to linear and non linear programming'*, Addison-Wesley, 1973.

[B25] R. Flecher, '*Practical Methods of Optimization*', Second Edition, John Wiley & Sons, 2003.

[B26] M.J.D. Powell, 'Non linear Optimization 1981', Academic Press, 1982.

[B27] J.J. Grainger, '*Power System Analysis'*, McGraw-Hill, Inc., 1994.

[B28] J. Arrillaga, '*Computer Analysis of Power Systems'*, John Wiley & Sons, 1990.

[B29] A. Grace, '*Optimization Toolbox User's Guide'*, The MathWorks, Inc., 1990.

[B30] Ray D. Zimmerman, 'MATPOWER A MATLAB™ Power System Simulation Package Version 2.0 User's Manual', December 1997.

[B31] A. J. Chipperfield and P. J. Fleming, ' Genetic algorithm Toolbox'.

[B32] Charles Darwin, 'The Origin of Species', 1979.

[B33] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison Wesley Publishing Company, January 1989.

8.3 OTHER REFERENCES

[B34] NRS 048, 'Electricity Supply - Quality of Supply', South African Bureau of Standards, Pretoria, 1996.

[B35] C Carter-Brown, 'Reticulation Planning Guideline Step-voltage regulators', Eskom Distribution, February 2002.

[B36] C Carter-Brown, 'Distribution voltage regulation and apportionment limits, Eskom Distribution, July 2003.

[B37] N. Burger & I vd Merwe, 'Phase Angle Regulator Technology', August 2004.

[B38] H.J. Beukes, B. Meyer, 'Converter-Based Devices for Network Support', October 2000.

[B39] Conversation with Markus Poller, Megawatt Park, March 2004.

[B40] H.J. Beukes, 'Network Options and Integration: Volume 1', Eskom SAPSSI Research Report SAPSSI/98/026, Stellenbosch, 1998.

[B41] E-mail to Ray Zimmerman, School of Electrical Engineering, Cornell University, Ithaca, NY 14853, 2 February 2004.

[B42] DigSilent News Letter 1/03.

[B43] Marek Obitko, Hochschule für Technik und Wirtschaft Dresden (FH) (University of Applied Sciences), student of the Czech Technical University in Prague, On-line document 'Genetic Algorithms'.

[B44] B. Meyer, Electronic voltage regulation of MV and LV power distribution networks, accepted for Energize, December 2000.

[B45] Conversation with Riaan Smit, Eskom Distribution, Western Region, 29 September 2004.

ADDENDUM A SVR PLACEMENT PROGRAM LISTINGS

The SVR placement program consists of 26 Matlab M-files, each performing a specific task. The following is a complete list of the M-files with a short description of its functionality.

bustypes	Builds lists of each type of bus
dSbus_dV	Computes partial derivatives of power injection w.r.t. voltage
ext2int	Converts external to internal bus numbering
idx_area	Defines variables for column indices to area
idx_brch	Defines variables for column indices to branch
idx_bus	Defines variables for column indices to bus
idx_gen	Defines variables for column indices to gen
InsertBranchVR	Inserts a new node and a transformer branch to insert the VR
insertbus	Inserts a new bus to bus data
insertBusVR	Inserts a new node and a transformer branch to insert the VR
int2ext	Converts internal to external bus numbering
makeSbus	Builds the vector of complex bus power injections
makeYbus	Builds the bus admittance matrix and branch admittance matrices
modYbus	Modifies the bus admittance matrix
mpoption	Used to set and retrieve a MATPOWER options vector
myrunpf	Runs a ordinary power flow
newtonpf	Solves the power flow using a full Newton's method
pfsoln	Updates bus, gen and branch data structures to match power flow
printpf	Prints optimal power flow results

radial	Defines the power flow data for 10 bus radial network
runtrfr	Launches the 'smooth' optimisation to obtain the optimal VR placement
trfrcon1	Set up non-linear constraints
trfrobj1	Set up objective function
VR1	Launching the Voltage Regulator placement Program
VRPart2	Launches the second part of the optimal SVR placement Program

ADDENDUM B LV EVR PLACEMENT PROGRAM LISTING

The LV EVR placement program consists of 50 Matlab M-files, each performing a specific task. The following is a complete list of the M-files with a short description of their functionality.

BS2RV	Decodes binary chromosomes into vectors of real values
BUSTYPES	Builds lists of each type of bus (ref, pv, pq)
CRTBASE	create a base vector
CRTBP	create a binary population
CRTRP	create a real-valued population
dSbus_dV	Computes partial derivatives of power injection w.r.t. voltage
ext2int	Converts external to internal bus numbering
GeneticLF	Reconfigures and solves a network for VR placement
idx_area	Defines variables for column indices to area
idx_brch	Defines variables for column indices to branch
idx_bus	Defines variables for column indices to bus
idx_gen	Defines variables for column indices to gen
InsetBusVR	Inserts a new VR at a bus
int2ext	Converts internal to external bus numbering
makeSbus	Builds the vector of complex bus power injections
makeYbus	Builds the bus admittance matrix and branch admittance matrices
MIGRATE	This function performs migration of individuals
mpoption	Used to set and retrieve a MATPOWER options vector

MUT	This function takes the representation of the current population, mutates each element with given probability and returns the resulting population
MUTATE	This function takes a matrix OldChrom containing the representation of the individuals in the current population, mutates the individuals and returns the resulting population
MUTBGA	This function takes a matrix OldChrom containing the real representation of the individuals in the current population, mutates the individuals with probability MutR and returns the resulting population
myrunpf	Runs a power flow
newtonpf	Solves the power flow using a full Newton's method
OBJVR1	Calculates the VR placement according to index into the chromosome data
pfsoln	Updates bus, gen, branch data structures to match power flow soln
printpf	Prints optimal power flow results
radial	Network data for 10 bus radial network
RANKING	This function performs ranking of individuals
RECDIS	This function performs discreet recombination between pairs of individuals and returns the new individuals after mating
RECINT	This function performs extended intermediate recombination between pairs of individuals and returns the new individuals after mating
RECLIN	This function performs extended line recombination between pairs of individuals and returns the new individuals after mating
RECMUT	This function performs line recombination with mutation features between pairs of individuals and returns the new individuals after mating
RECOMBIN	This function performs recombination between pairs of individuals and returns the new individuals after mating. The function handles multiple populations and calls the low-level recombination function for the actual recombination process
REINS	This function reinserts offspring into the population
REP	This function replicates a matrix in both dimensions
RESPLOT	This function plots some results during computation
retic1	Network data for 9 bus reticulation networks
RWS	Implements Roulette Wheel Selection

SCALING	This function implements a linear fitness scaling algorithm
SELECT	This function performs universal selection
SGAVR	This script implements the Simple Genetic Algorithm
SUS	This function performs selection with *STOCHASTIC UNIVERSAL SAMPLING*
XOVDP	This function performs double point crossover between pairs of individuals
XOVDPRS	This function performs double-point 'reduced surrogate' crossover between pairs of individuals
XOVMP	Multi-point crossover
XOVSH	Crossover Shuffle
XOVSHRS	This function performs shuffle 'reduced surrogate' crossover between pairs or individuals
XOVSPRS	This function performs single-point 'reduced surrogate' crossover between pairs or individuals

ADDENDUM C DATA STRUCTURE

Branch data:

```
F_BUS                       = 1;       f, from bus number
T_BUS                       = 2;       t, to bus number
BR_R                        = 3;       r, resistance (p.u.)
BR_X                        = 4;       x, reactance (p.u.)
BR_B                        = 5;       b, total line charging susceptance (p.u.)
RATE_A                      = 6;       rateA, MVA rating A (long term rating)
RATE_B                      = 7;       rateB, MVA rating B (short term rating)
RATE_C                      = 8;       rateC, MVA rating C (emergency rating)
TAP                         = 9;       ratio, transformer off nominal turns ratio
SHIFT                       = 10;      angle, transformer phase shift angle
BR_STATUS                   = 11;      initial branch status, 1 - in service, 0 - out of service
```

Bus data structure:

```
BUS_I     = 1;       bus number (1 to 29997)
BUS_TYPE  = 2;       bus type (1 - PQ bus, 2 - PV bus, 3 - reference bus, 4 - isolated bus)
PD        = 3;       Pd, real power demand (MW)
QD        = 4;       Qd, reactive power demand (MVAR)
GS        = 5;       Gs, shunt conductance (MW at V = 1.0 p.u.)
BS        = 6;       Bs, shunt susceptance (MVAR at V = 1.0 p.u.)
BUS_AREA  = 7;       %% area number, 1-100
VM        = 8;       Vm, voltage magnitude (p.u.)
VA        = 9;       Va, voltage angle (degrees)
BASE_KV   = 10;      baseKV, base voltage (kV)
ZONE      = 11;      zone, loss zone (1-999)
VMAX      = 12;      maxVm, maximum voltage magnitude (p.u.) (not in PTI format)
VMIN      = 13;      minVm, minimum voltage magnitude (p.u.) (not in PTI format)
```

ADDENDUM D QUADRATIC EXAMPLE

This example shows how the active set strategy is used to solve a constraint non-linear problem.

Problem is given by (D.1).

$$\begin{aligned} \min \quad & (x_1 - 3)^2 + 2(x_2 - 2)^2 \\ & \text{subject to} \\ & x_1 \geq 0 \\ & x_2 \geq 0 \\ & x_1 - x_2 \geq -1 \\ & -2x_1 - x_2 \geq -3 \end{aligned} \qquad \text{(D.1)}$$

Because this is a problem in two variables, the objective function and constraint functions can be visualised as three-dimensional surfaces. See Figure D-1.

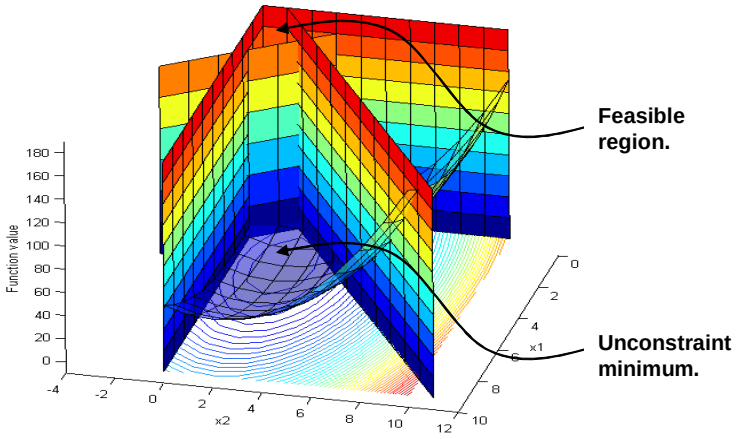


Figure D-1: Quadratic objective function with linear constraints

As can be seen from the surface plot in Figure 4.3, the constraint minimum is definitely different from the unconstraint minimum.

Solution:

The first step is to put the objective function in the standard quadratic form:

The objective function $(x_1 - 3)^2 + 2(x_2 - 2)^2$ in standard quadratic form is (D.2).

$$\frac{1}{2}\begin{bmatrix} x_1 & x_2 \end{bmatrix}\begin{bmatrix} 2 & 0 \\ 0 & 4 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} x_1 & x_2 \end{bmatrix}\begin{bmatrix} -6 \\ -8 \end{bmatrix} \tag{D.2}$$

and the constraints in standard form are as given by (D.3).

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & -1 \\ -2 & -1 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \geq \begin{bmatrix} 0 \\ 0 \\ -1 \\ -3 \end{bmatrix} \tag{D.3}$$

A feasible starting point is chosen as: $x_0 = (0,0)$. For this chosen starting point the active set is: $A_0 = (1,2)$. That means that at point $x_0 = (0,0)$, constraints 1 and 2 are satisfied. The problem given in (D.4) must now be solved:

$$\min \frac{1}{2}\begin{bmatrix} x_1 & x_2 \end{bmatrix}\begin{bmatrix} 2 & 0 \\ 0 & 4 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} x_1 & x_2 \end{bmatrix}\begin{bmatrix} -6 \\ -8 \end{bmatrix}$$

$$\text{subject to} \tag{D.4}$$

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} \delta_1 \\ \delta_2 \end{bmatrix} = 0$$

The solution is given by (D.5).

$$\begin{bmatrix} \delta_1 \\ \delta_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{D.5}$$

The Lagrange multipliers are now calculated to satisfy the derivative of the Lagrangian function by equation (D.6).

$$\begin{bmatrix} 2 & 0 \\ 0 & 4 \end{bmatrix}\begin{bmatrix} 0 \\ 0 \end{bmatrix} + \begin{bmatrix} -6 \\ -8 \end{bmatrix} = \begin{bmatrix} \lambda_1 & \lambda_2 \end{bmatrix}\begin{bmatrix} 1 \\ 1 \end{bmatrix} \tag{D.6}$$

which gives (D.7).

$$\begin{bmatrix}\lambda_1\\ \lambda_2\end{bmatrix}=\begin{bmatrix}-6\\ -8\end{bmatrix} \tag{D.7}$$

The negative Lagrange multiplier can be interpreted as follow: A negative Lagrange multiplier indicates that the objective function can be decreased by movement in a feasible direction away from that constraint. This can obviously be achieved by making that constraint inactive. If more than one active constraint has a negative Lagrange multiplier, the most negative one is chosen to leave the active set. This is the case in the sample problem and therefore the constraint with the most negative Lagrange multiplier is selected to leave the active set of constraint, namely a_2 will leave the active set. The active set now becomes $A_1=(1)$. The next step is to solve this new problem given by (D.8)

$$\min \frac{1}{2}\begin{bmatrix}\delta_1 & \delta_2\end{bmatrix}\begin{bmatrix}2 & 0\\ 0 & 4\end{bmatrix}\begin{bmatrix}\delta_1\\ \delta_2\end{bmatrix}+\begin{bmatrix}\delta_1 & \delta_2\end{bmatrix}\begin{bmatrix}-6\\ -8\end{bmatrix}$$

$$\text{subject to} \tag{D.8}$$

$$\begin{bmatrix}1 & 0\end{bmatrix}\begin{bmatrix}\delta_1\\ \delta_2\end{bmatrix}=0$$

The Lagrangian for this problem is given by (D.9).

$$\frac{1}{2}\begin{bmatrix}\delta_1 & \delta_2\end{bmatrix}\begin{bmatrix}2 & 0\\ 0 & 4\end{bmatrix}\begin{bmatrix}\delta_1\\ \delta_2\end{bmatrix}+\begin{bmatrix}\delta_1 & \delta_2\end{bmatrix}\begin{bmatrix}-6\\ -8\end{bmatrix}-\lambda_1\begin{bmatrix}1 & 0\end{bmatrix}\begin{bmatrix}\delta_1\\ \delta_2\end{bmatrix} \tag{D.9}$$

The associated optimality conditions are given by (D.10).

$$\begin{aligned}2\delta_1-6&=\lambda_1\\ 4\delta_2-8&=0\\ \delta_1&=0\end{aligned} \tag{D.10}$$

Which gives (D.11).

$$\begin{bmatrix}\delta_1\\ \delta_2\end{bmatrix}=\begin{bmatrix}0\\ 2\end{bmatrix} \tag{D.11}$$

and

$$[\lambda_1] = [-6]$$

This point is not feasible thus one of the inactive constraints must become active. It is now necessary to determine which inactive constraint must enter the active set. This is done using the following argument. Assume the previous point was feasible and the current point is not feasible. This can only happen if one of the inactive constraints was violated during the move from the previous point x_{k-1} to the current point x_k. By definition, an inactive constraint $a_i x_k \geq b_i$, for every $i \notin A$, is violated when $a_i x_k < b_i$. This can only happen if $a_i \delta_{k-1} < 0$. Therefore all the inactive constraints must be tested for $a_i \delta_{k-1} < 0$. This is the method described in [2] and it does seem to work. However it does not seem to be very efficient. Further work will have to be done to confirm this. This method will be used in this report. The inactive constraints 2, 3, and 4 are tested as in (D.12).

$$\begin{aligned} a_2 \begin{bmatrix} 0 \\ 2 \end{bmatrix} &= 2 > 0 \\ a_3 \begin{bmatrix} 0 \\ 2 \end{bmatrix} &= -2 < 0 \\ a_4 \begin{bmatrix} 0 \\ 2 \end{bmatrix} &= -2 < 0 \end{aligned} \qquad \text{(D.12)}$$

Therefore, constraints 3 & 4 are both candidates to enter the active set. Since $\begin{bmatrix} \delta_1 \\ \delta_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 2 \end{bmatrix}$ was not feasible, a_1 does not leave the active set. Furthermore, constraint 3 will be chosen to enter the active set. The active set becomes: $A_2 = (1,3)$

The shortest step length α to this new constraint is now calculated using the formula (D.13).

$$\alpha_1 = \frac{b_i - a_i^T x_k}{a_i^T s_k} \qquad \text{(D.13)}$$

Therefore, (D.14).

$$\alpha_1 = \frac{-1 - \begin{bmatrix} 1 & -1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \end{bmatrix}}{\begin{bmatrix} 1 & -1 \end{bmatrix} \begin{bmatrix} 0 \\ 2 \end{bmatrix}} = \frac{1}{2} \tag{D.14}$$

Therefore, the next feasible point using $x_2 = x_1 + \alpha_1 \delta_1$ is (D.15).

$$x_2 = \begin{bmatrix} 0 \\ 0 \end{bmatrix} + \frac{1}{2} \begin{bmatrix} 0 \\ 2 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{D.15}$$

Now g_2 is calculated as (D.16).

$$g_2 = Hx_2 + g = \begin{bmatrix} 2 & 0 \\ 0 & 4 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} + \begin{bmatrix} -6 \\ -8 \end{bmatrix} = \begin{bmatrix} -6 \\ -4 \end{bmatrix} \tag{D.16}$$

and a new equality constraint problem is constructed and must be solved as in (D.17).

$$\begin{bmatrix} \delta_1 & \delta_2 \end{bmatrix} \begin{bmatrix} 2 & 0 \\ 0 & 4 \end{bmatrix} \begin{bmatrix} \delta_1 \\ \delta_2 \end{bmatrix} + \begin{bmatrix} \delta_1 & \delta_2 \end{bmatrix} \begin{bmatrix} -6 \\ -4 \end{bmatrix} - \lambda_1 \delta_1 - \lambda_2 (\delta_1 - \delta_2) \tag{D.17}$$

subject to (D.18).

$$\begin{aligned} \delta_1 &= 0 \\ \delta_1 - \delta_2 &= 0 \end{aligned} \tag{D.18}$$

The solution to this problem is given by (D.19).

$$\begin{bmatrix} \delta_1 \\ \delta_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, therefore \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$
$$and \tag{D.19}$$
$$\begin{bmatrix} \lambda_1 \\ \lambda_2 \end{bmatrix} = \begin{bmatrix} -10 \\ 4 \end{bmatrix}$$

Since constraint 1 has a negative Lagrange multiplier, it must leave the active set and a new equality constraint problem must be solved with only constraint 2 active. The derivative of this new Lagrangian function is given by (D.20).

$$\begin{bmatrix} 2 & 0 \\ 0 & 4 \end{bmatrix}\begin{bmatrix} \delta_1 \\ \delta_2 \end{bmatrix}+\begin{bmatrix} -6 \\ -4 \end{bmatrix}-\lambda_2(\delta_1-\delta_2) \tag{D.20}$$

This gives (D.21).

$$\begin{bmatrix} \delta_1 \\ \delta_2 \end{bmatrix}=\begin{bmatrix} \frac{5}{3} \\ \frac{5}{3} \end{bmatrix} \tag{D.21}$$

and

$$\lambda_3=-\frac{8}{3}$$

Again, the inactive constraints must be checked for blocking since this is not a feasible point. Obviously constraint 1 & 2 will not be blocking and constraint 3 is already active. Therefore, only constraint 4 needs to be checked. Again, using the formula $\alpha_i=\frac{b_i-a_i^T x_k}{a_i^T s_k}$ to compute the step length α_3 gives (D.22).

$$\alpha_3=\frac{2}{5} \tag{D.22}$$

.Therefore, (D.23).

$$x_4=x_3+\alpha_3\delta_3=\begin{bmatrix} \frac{2}{3} \\ \frac{5}{3} \end{bmatrix} \tag{D.23}$$

This new constraint equality problem must now be solved with point $x_4=\begin{bmatrix} \frac{2}{3} \\ \frac{5}{3} \end{bmatrix}$ as the origin.

First the new gradient is calculated as $g_5=Hx_4+g$, which gives (D.24).

$$\begin{bmatrix} 2 & 0 \\ 0 & 4 \end{bmatrix} \begin{bmatrix} \frac{2}{3} \\ \frac{5}{3} \end{bmatrix} + \begin{bmatrix} -6 \\ -8 \end{bmatrix} = \begin{bmatrix} -\frac{16}{3} \\ -\frac{4}{3} \end{bmatrix} \tag{D.24}$$

and the derivative of the new Lagrangian function becomes (D.25).

$$\begin{bmatrix} 2 & 0 \\ 0 & 4 \end{bmatrix} \begin{bmatrix} \delta_1 \\ \delta_2 \end{bmatrix} + \begin{bmatrix} -\frac{16}{3} \\ -\frac{4}{3} \end{bmatrix} - \lambda_3(\delta_1 - \delta_2) - \lambda_4(-2\delta_1 - \delta_2) \tag{D.25}$$

together with the original constraints 3 & 4 gives solution (D.26)

$$\begin{bmatrix} \delta_1 \\ \delta_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

and

$$\lambda_3 = -\frac{2}{3}$$

$$\lambda_4 = 2 \tag{D.26}$$

$$x_5 = x_4 + \alpha\delta_4 = \begin{bmatrix} \frac{2}{3} \\ \frac{5}{3} \end{bmatrix}$$

Because constraint 3 has a negative Lagrange multiplier, it must leave the active set in the next iteration. The derivative of the new Lagrangian function with only constraint 4 active, becomes (D.27).

$$\begin{bmatrix} 2 & 0 \\ 0 & 4 \end{bmatrix} \begin{bmatrix} \delta_1 \\ \delta_2 \end{bmatrix} + \begin{bmatrix} -\frac{16}{3} \\ -\frac{4}{3} \end{bmatrix} - \lambda_4(-2\delta_1 - \delta_2) \tag{D.27}$$

and together with the original constraint 4 , $(-2\delta_1 - \delta_2) = 0$, gives the optimality equations. These equations are solved to give (D.28).

$$\begin{bmatrix}\delta_1\\ \delta_2\end{bmatrix}=\begin{bmatrix}\frac{1}{9}\\ -\frac{2}{9}\end{bmatrix} \qquad \text{(D.28)}$$

and

$$\lambda_4=\frac{20}{9}$$

Now, this solution must be checked for possible blocking constraints. Only constraint 2 is a possible blocking constraint. Using the formula to calculate the step length to this constraint gives (D.29).

$$\alpha_5=\min\frac{b_2-a_2^T x_5}{a_2^T\delta_5},1$$

with

$$\frac{b_2-a_2^T x_5}{a_2^T\delta_5}=\frac{15}{2},\ \alpha_5=1. \qquad \text{(D.29)}$$

and

$$x_6=x_5+\alpha_5\delta_5=\left(\frac{7}{9},\frac{13}{9}\right)$$

and

$$\lambda_4=\frac{20}{9}$$

Since the Lagrange multiplier of the active constraint is positive, this is the final solution of the constraint problem.

Verification of results:

The following is the Matlab code necessary to solve the above problem using the Optimization Toolbox. Matlab code is given in green and Matlab results are given in blue.

Write the following M-file:

```
function f = Gochen(x)
f = (x(1)-3).^2 + 2*(x(2)-2).^2;
```

Define the linear constraints:

```
a = [-1 0;0 -1;-1 1;2 1];
b = [0;0;-1;-3]
```

Define the starting point:

```
x0 = [0; 0];
```

Set the output options for the optimisation routine so that details at each iteration are reported on.

```
options = optimset('display','iter')
```

Call the fmincon function with the above parameters.

```
[x,fval,exitflag,output,lambda] = fmincon(@Gochen,x0,A,b,[],[],[],[],[],options)
```

The results are:

```
                                        max         Directional
 Iter    F-count        f(x)    constraint    Step-size    derivative      Procedure
    1          3          17             0             1          -17.3
    2          7     5.66667             0             1          -0.27
    3         11     5.56078             0             1        -0.0104
    4         15     5.55556             0             1      -1.01e-014
```

Optimisation terminated successfully:

```
x =
    0.7778
    1.4444
```

To see the value of the Lagrange multiplier at the solution point enter:

```
lambda.ineqlin
```

which gives:

```
ans =
         0
         0
         0
    2.2222
```

As can be seen, the solution from Matlab is exactly the same as that obtained from the manual calculation.

ADDENDUM E TAYLOR EXPANSION

The following is an example to show how the Taylor expansion is used to approximate a function around an operating point. The function used is $f(x) = x^3 - 2x - 5$. The starting point is chosen as $f(6) = 199$. In this example $q(x)$ is the Taylor approximation of $f(x)$.

$$\begin{aligned} f(x) &= x^3 - 2x - 5 \\ f'(x) &= 3x^2 - 2 \\ f''(x) &= 6x \\ f(6) &= 199 \\ f'(6) &= 106 \\ f''(6) &= 36 \end{aligned} \tag{E.1}$$

$$q(x) = f(x_0) + f'(x_0)(x - x_0) + f''(x_0)(x - x_0)^2$$

$$q(6) = 18x^2 - 110x + 211$$

$$q'(6) = 0 \Rightarrow x = 3.056$$

This point is now used to generate the next quadratic model of the original objective function. As can be seen, movement is already made in the direction of decent. The above procedure is repeated for $x = 3.056$

$$\begin{aligned} f(3.056) &= 17.43 \\ f'(3.056) &= 26.02 \\ f''(3.056) &= 18.34 \end{aligned} \tag{E.2}$$

$$q(3.056) = 9.17x^2 - 30x + 23.56$$

$$q'(3.056) = 0 \Rightarrow x = 1.637$$

The result after two iterations is shown in Figure E-1.

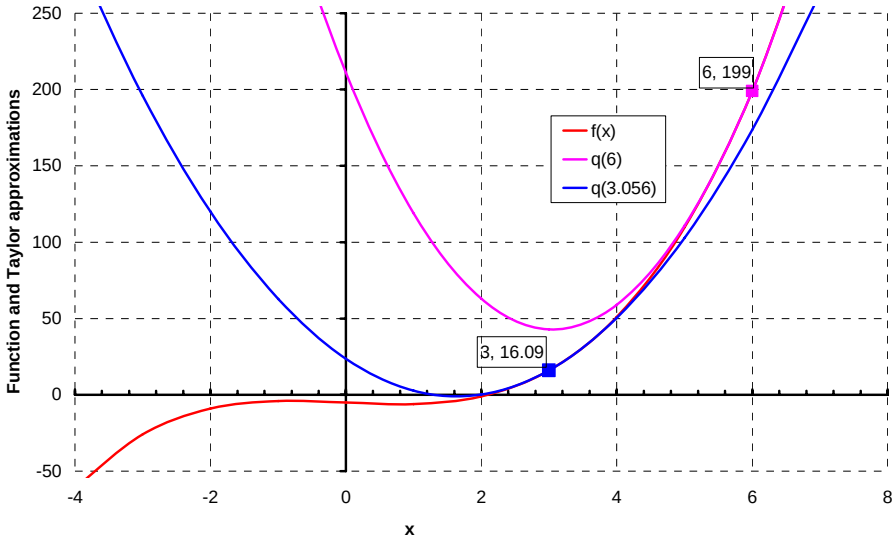


Figure E-1: Illustration of the use of the Taylor series expansion

ADDENDUM F OPTIMISATION PARAMETERS

Table 11: Mathworks Optimization Toolbox optimisation parameters

DerivativeCheck	Compare user-supplied analytic derivatives (gradients or Jacobian) to finite differencing derivatives.
DiffMaxChange	Maximum change in variables for finite difference derivatives.
DiffMinChange	Minimum change in variables for finite difference derivatives.
Display	Level of display. 'off' displays no output; 'iter' displays output at each iteration; 'final' displays just the final output; 'notify' displays output only if function does not converge.
GradConstr	Gradients for the non-linear constraints defined by the user.
GradObj	Gradient(s) for the objective function(s) defined by the user.
Hessian	If 'on', function uses user-defined Hessian, or Hessian information (when using HessMult), for the objective function. If 'off', function approximates the Hessian using finite differences.
HessMult	Hessian multiply function defined by the user.
HessPattern	Sparsity pattern of the Hessian for finite differencing. The size of the matrix is n-by-n, where n is the number of elements in x0, the starting point.
LargeScale	Use large-scale algorithm if possible.
MaxFunEvals	Maximum number of function evaluations allowed.
MaxPCGIter	Maximum number of PCG iterations allowed.
PrecondBandWidth	Upper bandwidth of preconditioner for PCG.
TolCon	Termination tolerance on the constraint violation.
TolFun	Termination tolerance on the function value.

TolPCG	Termination tolerance on the PCG iteration.
TypicalX	Typical x values. The length of the vector is equal to the number of elements in x0, the starting point.

ADDENDUM G DPL CODE

Table 12: Variable declaration

Myloadflow	Load Flow Calculation
Line 1	Line section 1
Line 2	Line section 2
MyRes	MyResults

```
int x;
x=1;
while (x <40)
{
Line1:dline = x;
Line2:dline = 40-x;
myloadflow.Execute();
MyRes.WriteDraw();
x = x+1;
}
```

Figure G-1: DPL code