A comparison of exhaustive and genetic algorithm evaluation for power system control design

Simon Douglas Wookey

A thesis submitted for the degree of Bachelor of Engineering (Electrical)

School of Engineering & Information Technology
University College
The University of New South Wales
Australian Defence Force Academy

01 November 2010
Contents

List of Figures iii

List of Tables iv

Abstract v

Declaration vi

Acknowledgments vii

1 Introduction 1

1.1 Contextual information and framework 1

1.2 Aims and purpose of research 5

1.3 Methodology and outline of thesis process 5

2 Literature Review 8

2.1 Previous research 8

2.2 Position of thesis 9

3 Experimental Methodology 11

3.1 Function evaluation 11

3.2 Reasons for choices 15

4 Experimental Results and Discussion 16

4.1 First function 16

4.2 Second function 18

4.3 Discussion 21

5 Conclusion and Recommendations for future Work 26

5.1 Conclusion 26

5.2 Future research and work 27

5.3 Recommendations 27
## List of Figures

1.1 A flow chart describing the exhaustive evaluation algorithm. ........................................ 2

1.2 An example of one method of solving a function for every combination of variable values in MATLAB. ............................................................. 3

3.1 Comparing solutions within exhaustive evaluation. ..................................................... 12

3.2 Initialising population of dynamic parameters in MATLAB. ..................................... 13

4.1 The five variables and solution varying with each successive generations. ................. 18

4.2 The twelve variables and solution varying with each successive generations. ............. 20

4.3 Time taken to exhaustively evaluate five and 12 dynamic parameters for varying levels of coarseness. ............................................................... 21

4.4 Number of chromosomes and iterations evaluated over time. .................................. 25

4.5 The solution given for exhaustive and genetic algorithm evaluation over time. ............. 25

D.1 A comparison of the time taken to evaluate functions to increasing accuracy and/or with increasing complexity. Not to scale. ................................. 45

D.2 Solving a simple, single-variable function in MATLAB. ............................................. 45

D.3 Example $y(x)$ data displayed graphically. .............................................................. 46

D.4 Initialising a population for a simple, single-variable function. ................................. 47

D.5 Determining the fitness of a population. ................................................................. 47

D.6 The fitness of each chromosome determines the size of its assigned chunk on a roulette wheel. ......................................................................................... 48

D.7 Selecting parents from a population a population. ................................................... 49

D.8 Producing children from parents using two-point crossover. ..................................... 52

D.9 Examples of how the population tends towards the optimal solution. ......................... 53
List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>Data for three levels of coarseness for the first function.</td>
<td>16</td>
</tr>
<tr>
<td>4.2</td>
<td>Data for four levels of coarseness for the first function.</td>
<td>16</td>
</tr>
<tr>
<td>4.3</td>
<td>Data for five levels of coarseness for the first function.</td>
<td>17</td>
</tr>
<tr>
<td>4.4</td>
<td>Data for 25 levels of coarseness for the first function.</td>
<td>17</td>
</tr>
<tr>
<td>4.5</td>
<td>Data for genetic algorithm evaluation for the first function.</td>
<td>18</td>
</tr>
<tr>
<td>4.6</td>
<td>Data for three levels of coarseness for the second function.</td>
<td>19</td>
</tr>
<tr>
<td>4.7</td>
<td>Data for four levels of coarseness for the second function.</td>
<td>19</td>
</tr>
<tr>
<td>4.8</td>
<td>Genetic algorithm evaluation for the second function.</td>
<td>21</td>
</tr>
<tr>
<td>A.1</td>
<td>Planned thesis timeline.</td>
<td>29</td>
</tr>
<tr>
<td>A.2</td>
<td>Actual thesis timeline.</td>
<td>30</td>
</tr>
<tr>
<td>D.1</td>
<td>Detailed $x$ and $y$ values for a simple, single-variable function.</td>
<td>46</td>
</tr>
<tr>
<td>D.2</td>
<td>Initial population data.</td>
<td>47</td>
</tr>
<tr>
<td>D.3</td>
<td>Initial population and fitness data.</td>
<td>48</td>
</tr>
<tr>
<td>D.4</td>
<td>Two-point crossover of two parents.</td>
<td>50</td>
</tr>
</tbody>
</table>
Abstract

This Thesis described a technique for using genetic algorithms to evaluate norm-bounded matrices, whose elements are multivariable functions. Two complex functions were solved using both exhaustive evaluation and genetic algorithms. The accuracy of each solution, as well as the time and computational resources required for evaluation, were compared. A conclusion was given on the advantages and disadvantages of exhaustive and genetic algorithm evaluation. This thesis also examined the present use of genetic algorithms in the field of power system control design, and the applicability of wider use of such a technique in future research. Finally, a recommendation of the use of genetic algorithms in power system control design was given.
Declaration

I hereby declare that this submission is my own work and to the best of my knowledge it contains no material previously published or written by another person, nor material which to a substantial extent has been accepted for the award of any other degree or diploma at UNSW or any other educational institution, except where due acknowledgement is made in the thesis. Any contribution made to the research by colleagues, with whom I have worked at UNSW or elsewhere, during my candidature, is fully acknowledged. I also declare that the intellectual content of this thesis is the product of my own work, except to the extent that assistance from others in the project’s design and conception or in style, presentation and linguistic expression is acknowledged.

Signature     Date
Acknowledgments

I would sincerely like to thank my supervisor, A/Prof. H.R. Pota, who has guided me through the course of the project. A/Prof. Pota agreed to become my supervisor at short notice after I encountered significant problems with my thesis in First Semester, 2010. He also provided me with this topic to pursue. Since taking me on as a thesis student, he has always been willing to see me at any time of the day, and to answer my questions as I needed them answered. This thesis would not have been possible without his guidance and support.

I would also like to thank M.J. Hossain for taking the time from his own PhD research to assist me in the understanding of the various aspects of power system control design that I required for this project. Additionally, I would like to thank S.M. Elsayed, who also took time from his research to assist me with proper evaluation of genetic algorithms when I struggled to achieve the correct results. His assistance in helping me to solve functions using genetic algorithms in MATLAB was invaluable. I am deeply indebted to both these men, because without their assistance, I would not have been able to accomplish the research and work necessary to complete this thesis.

I wish to express my gratitude to my family, who have provided support and love for me not only during the course of this thesis, but over my four years of university study. In particular, I wish to thank my uncle Greg Watkins, himself a graduate of UNSW with a Bachelor of Engineering (Electrical). He has assisted me during this thesis by providing me with the benefit of his experience completing his own thesis. He has also been available to informally discuss various ideas and problems that I have encountered, and he regularly contacted me to remain apprised of my progress. When I began writing, he provided detailed and considerable suggestions and criticism, from which I benefited enormously. I would like to thank the rest of my family for their understanding of and assistance to me, especially in my final year.

Finally, I wish to thank my friends, both students at UNSW@ADFA and elsewhere, for the good times that we have had over the years. I am grateful to them for providing me with a way to escape my studies occasionally, and a way to keep myself motivated to succeed at university.

Simon Wookey
Chapter 1

Introduction

1.1 Contextual information and framework

Genetic algorithms (GAs) are a method of evaluating complex functions in an efficient manner. They facilitate intelligent variable choices that provide the greatest chance of determining the best solution in the shortest time. The techniques that can be used within GAs are discussed elsewhere this thesis (in Appendix D).

The use of genetic algorithms previously and currently in the field of power system control design, as well as in numerous other fields, has been successful but limited. This research is discussed further in Chapter 2 of this thesis. Within the field of power system control design, there are two main areas in which GAs can be employed: complex, one time evaluation in design of the system, or continuous, rapid evaluation for direct control of the system. The potential benefits of successfully integrating GAs into power system control design are impressive, and include increased accuracy for both design and control applications, as well as faster evaluation. In real terms, increased efficiency and less calculation time translates to greater performance and lower cost, depending on the application.

1.1.1 Exhaustive evaluation

Exhaustive evaluation is a method of solving functions, where every possible combination of inputs is evaluated and a solution saved. Exhaustive evaluation does not employ any form of intelligence to increase the efficiency or accuracy of solving a function. Rather, the function is evaluated for every permutation of the dynamic parameters that govern the behaviour of the function. The accuracy of a solution is defined with regards to a particular function, but generally it means how close the given solution is relative to the optimal solution. Defining accuracy quantitatively becomes difficult when the optimal solution is not known. It is also useful to consider functions in terms of simplicity: this is defined in terms of the time taken to solve the function. If a function takes a minute to solve, then it is simpler than one which takes an hour to solve. This could be because of the number of iterations of the function required for evaluation, or the time taken for a single iteration.

In a multivariable function, for example \( y(a, b, c) \), the argument can be written as:

\[
\begin{bmatrix}
a \\
b \\
c
\end{bmatrix}
\begin{bmatrix}
1 & a_2 & \ldots & a_{n-1} & a_n \\
1 & b_2 & \ldots & b_{n-1} & b_n \\
1 & c_2 & \ldots & c_{n-1} & c_n
\end{bmatrix}
\]
and every combination of \((a, b, c)\) is required to be evaluated for the solution of \(y\) to be fully achieved. It is apparent that with a single variable, this process is quite simple. When many variables are used, such as with \(y(a, b, c)\), selecting every combination becomes more difficult, yet no less necessary. This is the basis of exhaustive evaluation. The algorithm used in exhaustive evaluation is shown in flowchart from in Figure 1.1.

![Flowchart of Exhaustive Evaluation Algorithm](image)

**Figure 1.1:** A flow chart describing the exhaustive evaluation algorithm.

Exhaustive evaluation is achieved using the steps outlined in the flow chart. It is the standard method for solving simple functions, and it is widely taught and used through all levels of education. It is one of the most intuitive method of evaluating functions, and amongst the simplest methods to implement. The time required to solve a simple function in MATLAB is minimal; solving \(y = x^2\) for a single variable \(x\) is written as follows:

\[
x = [x_1, x_2, \ldots, x_n-1, x_n]; \\
y = x^2;
\]

This example is trivial, but it illustrates the point in question because it requires \(x\) to be predefined. Now in some circumstances this can be a desirable quality: it allows for a prescribed level of accuracy to be achieved. Users can define how many increments there are between \(x_{\text{min}}\) and \(x_{\text{max}}\), and this means Exhaustive evaluation also allows for an estimate to be made of the time required to evaluate every possible combination of variables. Because of this, exhaustive evaluation is a flexible method, and it is able to be tailored to meet a level of accuracy or a time limit, given the function is sufficiently simple.

When \(n\) dynamic parameters are defined, some method of choosing each combination must be used. This requires that the minimum and maximum bounds of the dynamic parameter(s) be defined, as well as the size of the increment \(\Delta \text{var}_n\). The simplest method to cycle through every combination is to use nested for loops, as shown in Figure 1.2.

Using for loops means that every possible combination will be tested. This method was used in the exhaustive evaluation of the two functions: see Appendix [2] on page 54 for the complete code that was used.

**Advantages**

The main advantage of using exhaustive evaluation is the inherent ability to customise the solution to meet the level of accuracy that the user requires. Another advantage is the simplicity of implementation.
§1.1 Contextual information and framework

Disadvantages

Exhaustive evaluation must evaluate every possible combination of variables, and while this does allow for customisation of accuracy, it also leads to substantial time requirements.

Using exhaustive evaluation, the number of iterations $i$ is given by Equation (1.1)

$$i = \Delta_{inc}^n$$

where $\Delta_{inc}$ is the number of increments between the upper and lower bounds of each variable, and $n$ is the number of dynamic variables for the function. The number of increments for each variable is called coarseness: as the coarseness increases, then the accuracy increases. However, this is where the trade-off occurs: as the number of variables or the coarseness grows, the time taken to evaluate the function increases polynomially. For single variable functions, or functions with a very low coarseness, the number of iterations is not great.

The number of iterations required to evaluate a function where $\Delta_{inc} = 5$ and $n = 4$ is 625, but to evaluate this to $\Delta_{inc} = 15$ means that $i = 50,625$. Similarly, to evaluate a function where $\Delta_{inc} = 5$ and $n = 10$ means that $i = 9,765,625$. The time per iteration becomes very important when one must deal with large values of $i$ - if $t_i \approx 1\mu s$, then the function will be solved in around 10 s. However, if $t_i \approx 1\ s$, then the function will take around 113 days to be evaluated. For the purposes of this project, 15 levels of coarseness is considered to be the minimum level that provides sufficient accuracy to be confident that the results are reliable. This is the main disadvantage of exhaustive evaluation.

Another disadvantage is the nature in which the variable arrays are defined. Because they are defined discretely, there is a very good chance that the best solution will not be given because the best combination of variables to give it will not be available. Therefore, for a multivariable function, even at higher levels of coarseness, the exhaustive evaluation method will likely not produce the optimal result at all. This is not the case when the optimal variable values lie on the maximum or minimum of the variable limits, as seen in the evaluation of Function 1, but this is not the general case. The discrete definition of variables is another serious disadvantage.

Figure 1.2: An example of one method of solving a function for every combination of variable values in MATLAB.
1.1.2 Genetic algorithms

**Note:** Genetic algorithms are a reasonably complex concept to understand, and this brief summary may not be sufficient for some readers. For a more detailed introduction to genetic algorithms, please refer to Appendix D on page 42.

Genetic algorithms are a method of solving complex functions using the idea of evolution. They may be understood as “an ‘intelligent’ probabilistic search algorithm which can be applied to a variety of combinatorial optimisation problems.” Beasley et al. state that GAs simulate those processes in the natural world that cause evolution and progress amongst populations.

Potential solutions are encoded into chromosomes, which are then assessed for their fitness in a given environment. A chromosome, taken from the biological term, is simply a collection of the elements, or ‘genetic material’, that comprise a set of variables that can be used in order to evaluate the function. Each chromosome can be thought of as containing the a particular set of building blocks of the solution, and some sets are better than others. The function used to assess chromosomes is called the objective function. This fitness determines how well that particular chromosome solves the function.

Users must specify how fitness is assessed, and how solutions are encoded. There are often many chromosomes being tested, and these are referred to as the population. After each round, or generation, parents are selected from within the current population and children are produced to replace them in the population of the next generation. There are numerous techniques for parent selection and reproduction: see [3, 5, 6, 11] for further information regarding the multitude of techniques available, and the applications to which each is suited.

After a defined number of generations have passed, the optimal determined solution will be available for the user. The number of generations that are allowed to occur, or time allowed to pass, will determine the accuracy of the solution.

Because understanding the concepts of genetic algorithms is not immediately achievable, a worked example is included in Section D.4. An understanding of that simple example will help greatly with understanding the application of the various techniques and concepts discussed in this thesis. It explains the concepts and techniques that have been used in this thesis, but by using a far simpler example, readers who are unfamiliar with GAs should be able to pick up the techniques and concepts better than with a more complex example.

**Advantages**

One of the reasons that GAs are known to be prefereable to other methods is “...because mathematical properties such as differentiability, convexity, and linearity do not add to the complexity of the GA” [2]. This is the biggest advantage of this search method over traditional optimisation techniques. While using exhaustive evaluation, the required time for evaluation increases exponentially as complexity of the function increases (see Figure D.1) while GA evaluation time increases only linearly.

**Disadvantages**

The main disadvantage of genetic algorithms is the training that is required to properly use them. Achieving even a basic understanding of GAs is a task in itself, and not something that should be assumed to be easy. Another disadvantage is the time taken to set up a GA from scratch. Like any form of programming, writing the initial structure is only the first step, and troubleshooting and
debugging must be completed. The time required to write the algorithm to perform correctly can be days or weeks [5,6].
It is far quicker and more simple to use exhaustive evaluation, however the time that this takes to evaluate can be very long (days, weeks, years [3,10]). It is this reason that prompted researchers to explore alternative methods, one of which is the genetic algorithm [11].

1.2 Aims and purpose of research

1.2.1 Aims

The primary aim of this project was to assess the benefits and advantages of using genetic algorithms to optimise complex functions, compared to using exhaustive evaluation. This thesis seeks to determine the viability of such a practice, taking into account the trade-offs required to allow for it.

It should be noted that while this thesis deals with the optimisation of functions sourced from power system control design, the potential benefits of using genetic algorithms are not specific to power systems alone. The benefits of increased efficiency, and hence eventual lower cost, are potentially able to be reaped in any technical field.

1.2.2 Purpose

One purpose of this thesis was to be able to gain the requisite knowledge to solve a function using genetic algorithms, and to then take that knowledge and solve the functions. Another purpose was to be able to make a recommendation at the end of the process as to how easily this technique could be more widely implemented. The benefit of the time and effort required to learn about and set up the genetic algorithm must be weighed against the computation time and lessened accuracy of using simpler methods. This thesis also investigates the degree of difficulty that one might experience when faced with learning how to use genetic algorithms.

1.3 Methodology and outline of thesis process

This thesis was conducted over four distinct stages: function analysis and exhaustive evaluation, genetic algorithm research, genetic algorithm evaluation, and analysis of results. The following is a brief overview of each stage of the project.

1.3.1 Function analysis and exhaustive evaluation

This section formed the basis of the literature review (see Chapter 2) and also served to identify the requirements of exhaustively evaluating the two functions. During this stage, research was conducted into methods of solving functions similar to those two in question, as well as the purpose of the functions.
1.3.2 Genetic algorithm research

The next stage of the thesis was to conduct sufficient research into genetic algorithms to allow me to be able to properly use GAs in my research. First of all, it was necessary to begin at the very beginning, as prior to commencing this project, I had no knowledge of GAs. Davis’s *Handbook of Genetic Algorithms* [6] was an immensely useful reference during this stage of research. Using this and other references, I learned about GAs and the various techniques that one can employ when using this method to solve functions.

Following this, I researched the current state of GA use in power system control design, as well as more generally in technical fields. For a more complete literature review, see Chapter 2 on page 8.

1.3.3 Genetic algorithm evaluation

The next stage of the thesis was to put this knowledge into practice and use GAs to evaluate the two functions. This involved briefly concentrating on the simplest parent selection and reproduction techniques that would achieve an accurate solution. This section of the thesis involved all the MATLAB programming that was required to be written in order to be able to equation the functions using GAs.

1.3.4 Analysis of results

Once I had evaluated the functions using both exhaustive evaluation and genetic algorithms, I had collected enough data to compare the two method in a number of ways. I was able to compare the methods in a manner that allowed me to comment on the relative benefits and disadvantages associated with each method. This comparison is shown in Chapter 4 on page 16.

1.3.5 Structure of thesis

My thesis takes a standard structure, consisting of the introduction, literature review, experimental methodology, experimental results and discussion, and conclusions and recommendations. Following this, I have included appendices for further information, and finally a bibliography. The following is a very brief description of each chapter of the thesis.

**Chapter 2 – Literature review:** This chapter of the thesis includes the research that was conducted as part of my initial learning of genetic algorithms, and discusses past and present usage of genetic algorithms both in power system control design and in other fields.

**Chapter 3 – Experimental methodology:** This chapter of the thesis describes the method that I used to achieve the aims of the thesis. It covers the process that I took to complete the thesis. This has already been briefly covered in the introductory chapter, but Chapter 3 contains a far more detailed explanation of what I did, how I did it, and why I chose to use a particular method or technique. It can be understood as the narrative of my thesis.

**Chapter 4 – Experimental results and discussion:** This chapter of the thesis is a summary and discussion of the results that were obtained throughout the course of this thesis. It contains
the comparison between the raw results of exhaustive and genetic algorithm evaluation, as well as a summary of the performance parameters of each method, such as time taken and computational resources required. It provides data for comparing the two methods quantitatively.

Chapter 5 – Conclusions and recommendations for further work: This chapter of the thesis contains a brief summary of the results, and a recommendation as to the suitability of employing GAs more widely in power system control design. It talks about the high-level advantages and disadvantages of using and adopting genetic algorithms.
2.1 Previous research

2.1.1 Papers, and understanding GAs

The concept of using genetic algorithms in power system control design is not a new idea, however it is by no means widely implemented. A number of examples of implementation do exist, and these instances are discussed in this chapter. Papers in this field take a very similar structure: a definition of the problem or research area, followed by a definition of genetic algorithms. Despite the fact that GAs have been recognised as a very capable method of evaluating functions, their existence remains little known. The author of such a paper must therefore generally assume that readers will not be well acquainted with the concept of genetic algorithms, and therefore a plethora of definitions of the genetic algorithm are available. This definition is then followed by a necessarily protracted explanation of the algorithm, using the problem in question as the example. This is usually as a result of space restrictions - there is no room to include a proper, simple GA example in a great many journal papers.

It is critical that readers of these papers have a working knowledge of GAs, and perhaps on certain topics, the author might reasonably make the assumption that that is the case. However, if the reader is unfamiliar with GAs, beginning with a complex example will not help the reader to understand the various concepts and techniques involved. This makes the examples seen in many papers pointless - any person who is in a position to understand them is probably sufficiently acquainted with genetic algorithms to be able to understand the content of the paper without an example. By the same token, the significance of using GAs rather than other methods is lost on those readers who have difficulty understanding the example given. If space permits, providing a lower-level example is far more worthwhile than one outlined on the problem at hand. For a suitable introductory genetic algorithm example, see Appendix D.4.

Fonseca and Fleming describe a rank-based fitness assessment method for multiple objective genetic algorithms [9]. This method is not dissimilar to that used in this thesis (see Chapter 3 to view the methodology) - it describes how genetic algorithms can be used to optimise many variables over many objective functions simultaneously (which this thesis does not explore).

2.1.2 Research employing genetic algorithms in power system control

Genetic algorithm research began with John Holland and his students in Michigan in the late 60s [11], and has flourished in the years since, thanks to Schaffer’s seminal conference article [16]. GAs were traditionally used in general problem solving and optimisation, however over the last 15 or 20 years,
GAs have been adapted for use in a number of specialised fields. One area that the method has been used in is power system control design. This section will summarise a number of examples of this, including the solution, and the problem that prompted it.

Dimeo and Lee discuss the control system of a boiler-turbine plant [7]. They sought to use a genetic algorithm to implement various control methods for a nonlinear MIMO (multiple input, multiple output) plant model. These methods included a proportional-integrator (PI) controller and a state feedback controller. This is an example of a problem where the inputs are hard bounded. The authors state that the GA they have used is very similar to that found in other literature [10, 11, 14]. The authors used a similar setup to what is used in this thesis. They have also used the roulette wheel parent selection technique.

Taranto and Falcão describe in their paper how they have used genetic algorithms in the design of power system damping control [17]. They implemented classic controller design structures, but included within each controller three parameters, representing gain and phase characteristics, whose values were used to optimise the controller. These values were themselves optimised using genetic algorithms. This paper presents a similar problem to the one being studied in this thesis: genetic algorithms are used to optimise a multivariable output where the parameters have a well-defined set space. While the application is similar, Taranto and Falcão then went on to implement a truly decentralised control solution by simultaneously closing the control loop channels.

Developments in electronics capabilities in the early to mid 1990s led to an increase in the performance of power systems. However, their fast responses to system change placed greater emphasis on the control of these power systems. Reformat et al identified this shortfall, and discussed a method of combining advanced simulation and genetic algorithms in power system control design [15]. This method, the GeneticEMT, was a genetic algorithm-based method that used an electromagnetic transients (EMT) program as the evaluation environment. This method was successfully implemented on live power control systems.

Baaleh and Sakr also identified genetic algorithms as potentially being a useful tool in power system control design [2]. In that paper, the authors discuss a new method to design a supplementary power system stabilizer (PSS) that can be used in a synchronous machine infinite bus system. Their approach is based on the use of genetic algorithms to compute the optimisation settings of fixed stabiliser parameters off-line. The paper concludes that simulation of the GA-based PSS shows the technique is viable and effective, and allows great freedom in selection of parameters like frequency and voltage dampeners. This technique also allows for nonlinear control schemes to be implemented.

Abdel-Magid and Abido present a discussion on the design of robust power system stabilisers using genetic algorithms [1]. They use a traditional speed-based lead-lag PSS in their research. The authors formulated a multivariable optimisation problem to solve for a set of objective functions to characterise the PSS. The authors successfully implemented the scheme, and confirmed this result using nonlinear simulation results and eigenvalue analysis.

2.2 Position of thesis

2.2.1 Importance of topic

The use of genetic algorithms in power system control design has been successful, but quite limited. Genetic algorithms are a technique that has been heard of, but not very well understood by a great
many researchers in the field, and this means that these researchers are less inclined to use the method. GAs have been proven to be an efficient, successful method of multivariable function optimisation, however they have not been widely adopted. The reasons for this are many, but the main one is the difficulty in getting started. As has already been said, learning how to use GAs is not a quick process. This topic is important because it seeks to determine

Presenters and cameramen from a popular motoring show on television recently drove utility vehicles to the North Pole - one presenter said that while explorers had already reached the North Pole, the process was long and arduous. It was time, then, to see just how easily it could be done. This is an appropriate analogue to this thesis - genetic algorithms have already been used in power system control design, but the total process required to achieve the results was not a simple one, particularly if this included specialist learning before getting to solving the actual problem. This effort is substantial, but what other ways are there to achieve this outcome? How easily can it be done? These are the questions that this thesis seeks to answer.

### 2.2.2 Choice of topic

This topic was chosen primarily to establish how difficult it is to use genetic algorithms for optimisation, and to compare this to optimisation using exhaustive evaluation. It was also to provide a comparison between exhaustive and genetic algorithm evaluation of power systems control design, although this particular field is only one of many which could potentially benefit from wider use of genetic algorithms for optimisation. The comparison could be used to determine whether genetic algorithms might be used more widely in power system control design and other fields.
The experimental part of this thesis can be divided into two distinct components: exhaustive and genetic algorithm evaluation. This chapter details the methodology that was developed and followed throughout the course of the project. At the top level, the two processes are very similar, yet more a more detailed investigation will highlight the differences. The main steps are setting up for evaluation, evaluating the function, getting the best result, and comparing the best results of each method. This chapter describes how the functions were optimised, resulting in the delivery of the maximum weighted norm of $\tilde{\phi}$, which is considered to be the optimal solution.

3.1 Function evaluation

3.1.1 Exhaustive evaluation

Setting up for evaluation

Transfer function into MATLAB. The function (see Appendix C.1) was presented as part of a report [13] and was transferred into MATLAB. The function was recreated as a skeleton function. At this point, none of the variables, constant or dynamic, had values. The code merely existed as a framework for these values to be entered into and evaluated at a later stage.

Determine variable values. The various constant values were unknown at the beginning of the thesis, and were not given in the report from which the function was taken. However, these values were given by M.J. Hossain, co-author of [13], and these were input into MATLAB to join the skeleton function that had already been written. As well as the constants, all dynamic parameters were given at this point.

To test the suitability of the initial values of the dynamic parameters, they were varied between $\pm 20\%$ of their initial value. The number of values that each parameter could take was called the coarseness, and as coarseness increased, the accuracy of the solution increased as well.

Implement scheme for choosing parameters. With all constant and dynamic parameters now known, the next step is to develop a scheme that allows for each and every dynamic parameter combination to be selected and evaluated. This was done using nested for loops, one per parameter, using the algorithm shown in Figure 1.2.
3.1 Function evaluation

Evaluating the function

*Compare the solution of each combination.* After each combination of parameters has been evaluated, the solution for the weighted norm of $\tilde{\phi}$ is stored in a temporary variable. Additionally, the increment of each parameter is stored, as this determines the weighting (see Equations 4.1 and 4.2). This temporary variable is compared to the current best solution of the weighted norm of $\tilde{\phi}$, and whichever solution is greater is considered to be the better solution. The better solution, as well as the increments of the parameters that gave that solution, are then saved to be compared to subsequent solutions from different parameter combinations.

Getting the best solution

*Output the best solution.* Once every combination of parameters has been evaluated, the maximum weighted norm of $\tilde{\phi}$ for the given parameters has been found. At this point, the solution and other data is printed to the MATLAB console for viewing. The maximum weighted (and unweighted) norm of $\tilde{\phi}$ is returned, as is the increment of each parameter. A number between 1 and the level of coarseness is returned, where the minimum (1) indicates that that particular parameter was optimised at -20% and the maximum (the coarseness) indicates that the parameter was optimised at +20%. Values in between the min and the max mean that the optimal value lies somewhere within ±20% of the initial value.

Figure 3.1 shows the method that was used to compare the current solution with the previously identified maximum solution. After all combinations have been evaluated, the maximum solution is considered to be the best solution that the variable set can achieve.

```matlab
% After the function has been evaluated for a particular variable combination
% Assign current weighting
temp_var = [ds dsm Esmd Esqd Vts];
% Assign current weighted solution
temp_sol = norm(phi)/norm(tempvar);
% Compare the current solution with the best solution
if temp_sol > max_sol
    % If the current solution is better than the best solution, it
    % becomes the best solution
    max_sol = temp_sol;
    max_var = temp_var;
end
```

Figure 3.1: Comparing solutions within exhaustive evaluation.
3.1.2 Genetic algorithm evaluation

Setting up for evaluation

*Initialise a population.* The first step was to create an initial population, and this meant developing a method of encoding the variables into chromosomes. The population was initialised by generating a random number within the bounds each parameter. This was done as shown in Figure 3.2.

\[
\text{var} = \text{var}_{\text{min}} + \text{rand} \times (\text{var}_{\text{max}} - \text{var}_{\text{min}});
\]

Figure 3.2: Initialising population of dynamic parameters in MATLAB.

This was done for each of the five parameters, and this now included all the data required for the chromosome. However, the data had to be encoded in a such a manner that it was usable by the genetic algorithm. This was done by encoding a single chromosome as an array with five elements:

\[
X_n = [\delta^* \delta_m^* E_m^* V_t^*]
\]

where these five parameters were those five that were just generated.

When the entire initial population of \(n\) chromosomes had been generated, the individuals were stored in a matrix \(X_{\text{pop}}\), where:

\[
X_{\text{pop}} = \begin{bmatrix}
X_1 \\
X_2 \\
\vdots \\
X_{n-1} \\
X_n
\end{bmatrix} = \begin{bmatrix}
\delta_1^* & \delta_{m,1}^* & E_{m,1}^* & E_{q,1}^* & V_{l,1}^* \\
\delta_2^* & \delta_{m,2}^* & E_{m,2}^* & E_{q,2}^* & V_{l,2}^* \\
\vdots & \vdots & \vdots & \vdots & \vdots \\
\delta_{n-1}^* & \delta_{m,n-1}^* & E_{m,n-1}^* & E_{q,n-1}^* & V_{l,n-1}^* \\
\delta_n^* & \delta_{m,n}^* & E_{m,n}^* & E_{q,n}^* & V_{l,n}^*
\end{bmatrix}
\]

Now that the population \(X_{\text{pop}}\) is encoded and stored in such a manner that each chromosome can be evaluated.

Evaluating the function

*Evaluate the population.* The population must now be evaluated. This involves taking each chromosome and evaluating it using the objective function: in this case, the functions were \(\text{Eq1func}\) and \(\text{Eq2func}\), shown in Appendices E.5 and E.6 respectively. Each chromosome was evaluated individually, and the returned solution was added to the array \(\tilde{\phi}_{\text{soln}}\), such that \(\tilde{\phi}_{\text{soln}} = [\tilde{\phi}_1 \tilde{\phi}_2 \ldots \tilde{\phi}_{n-1} \tilde{\phi}_n]\). This array is then assessed for fitness: the solutions are scaled such that each \(\tilde{\phi}_{\text{soln}} = \tilde{\phi}_{\text{soln}} - \tilde{\phi}_{\text{min}}\), which means that the minimum (worst) solution becomes 0. The other solutions are now scaled relative to this minimum solution. This relative solution is called the fitness of each chromosome. This fitness \(X_{\text{fit}}\) is then normalised such that \(\text{sum}(X_{\text{fit}}) = 1\).

This array \(X_{\text{fit}}\) is used to rank each chromosome, in order of fitness. This is important for parent selection later on. Using the MATLAB command \texttt{sortrows}, the chromosomes are sorted based on the column \(X_{\text{fit}}\), meaning that now the chromosomes are ranked in the appropriate order.

*Elitism.* The first three chromosomes are taken and put to one side, and are labelled as \(X_{E1}, X_{E2}\) and \(X_{E3}\). These are the three most desirable (the elite) chromosomes, and we want them to perpetuate within the population. After the new population has been generated, these three individuals replace three random chromosomes.
Parent selection. This was done using the roulette wheel selection technique. For a more detailed description, please refer to Appendix E.4.1. Additionally, see [3, 5, 6]. Parents were selected based on the value of a random number, and the chances of selection were proportional to the fitness of each chromosome. Two parents are selected, and these two parents produce two children. Therefore, the parent selection loop must be run for half as many times as the total population size. By biasing the selection of parents towards the fitter individuals, the children are more likely to be fitter as well, and so the average fitness will be raised. This accelerates the evolution of the population towards the optimal solution.

Reproduction. After the parents had been selected, they are converted into binary strings and are randomly split into three parts, such that Parent 1, given by \( P_1 \), is written as:

\[
P_1 = \begin{bmatrix}
\delta_n^* \\
\delta_{m,n}^* \\
E_{m,n}^* \\
E_{q,n}^* \\
V_{l,n}^*
\end{bmatrix} = \begin{bmatrix}
P_{1A} & P_{1B} & P_{1C} \\
\vdots & \vdots & \vdots \\
\vdots & \vdots & \vdots \\
\vdots & \vdots & \vdots \\
\vdots & \vdots & \vdots
\end{bmatrix}, \quad P_2 = \begin{bmatrix}
P_{2A} & P_{2B} & P_{2C} \\
\vdots & \vdots & \vdots \\
\vdots & \vdots & \vdots \\
\vdots & \vdots & \vdots \\
\vdots & \vdots & \vdots
\end{bmatrix}
\]

Parent \( P_2 \) is split using two random numbers to determine the split points, and can be represented as an array of five strings like \( P_1 \), as shown above. The two children, \( C_1 \) and \( C_2 \), are made from a combination of \( P_1 \) and \( P_2 \), such that:

\[
C_1 = \begin{bmatrix}
P_{1A} & P_{2B} & P_{1C} \\
\vdots & \vdots & \vdots \\
\vdots & \vdots & \vdots \\
\vdots & \vdots & \vdots
\end{bmatrix} \quad \text{and} \quad C_2 = \begin{bmatrix}
P_{2A} & P_{1B} & P_{2C} \\
\vdots & \vdots & \vdots \\
\vdots & \vdots & \vdots \\
\vdots & \vdots & \vdots
\end{bmatrix}
\]

Mutation. Mutation is where a single bit of a child is inverted. This is programmed to occur 0.5%, and can introduce unexpected elements into the new generation. A random number between 0 and 1 is generated, and if it is less than 0.005, then mutation occurs. Some mutations prove to be extremely useful, and are widely adopted by the population over a number of generations. The majority of mutations, however, result in undesirable side effects, and cause that particular child to have a low fitness, and subsequently a low chance of reproduction.

Creating the new population. Once the new child population \( C_n \) has been made, the elite individuals from the previous generation are added to the population. Three children are randomly replaced by the elite individuals. Once this is done, the child population is the same size as the parent population. \( C_n \) must now be evaluated, and ranked. This is done by using the objective function, as was done initially, and then determining the fitness \( C_{fit} \) of the child population. Once the fitness is determined, the population is sorted accordingly, and at this stage a direct replacement of the adult \( X_{pop} \) and child \( C_n \) population occurs. At this point, \( C_n \) becomes \( X_{pop} \).

Repeat the process. Once the population has been ranked, the three elite chromosomes are copied to one side, parents are selected, children are created and then evaluated. Their fitness is assessed and they are ranked accordingly. This process is a single generation, and genetic algorithms run for a prescribed number of generations.
Getting the best solution

*The best solution.* After the algorithm has repeated the defined number of times, the program enters its final phase. The chromosomes are now evaluated and ranked for the last time, and the fittest chromosome at that point is taken to be the optimal solution.

3.1.3 Comparing the best solutions for each method

After both methods have delivered their optimal solutions, the two are compared. The largest solution is considered to be the best. However, the comparison extends further than simply magnitude of the solution. The time taken to evaluate the solution is also considered, as is the computational resources required.

3.2 Reasons for choices

**Roulette wheel method.** The roulette wheel method was selected because of its bias towards fitter parents, and the simplicity of implementation. It is a good method for optimisation problems, whereas complex modelling may benefit from using other, more ‘natural’ methods [6]. There are other examples in literature of this method being used [7].

There are numerous other methods that exist for intelligent evaluation of precisely the sort of functions that this thesis is dealing with. The reason for choosing genetic algorithms over other methods was because of their relative anonymity – other methods of optimisation are better known, and GAs only feature in a small number of examples.

The goal of evaluating these functions is to determine the best solution, which is taken to be the maximum weighted norm of \( \tilde{\phi} \). The reason for choosing the maximum weighted norm of \( \tilde{\phi} \) as the optimal solution is because the literature supports this [12, 13]. These reports suggest that as the weighted norm of \( \tilde{\phi} \) increases, the performance of the system will also increase. The details of the performance increase are another story, but the reports indicate that researchers should be interested in trying to optimise these functions.
Chapter 4

Experimental Results and Discussion

4.1 First function

4.1.1 Exhaustive evaluation

This function aims to find the quantity

\[ ||\tilde{\phi}|| \]
\[ ||[\delta^* \delta_m^* E_m^* E_q^* V_t^*]|| \]

which is hereafter referred to as the weighted norm of \( \tilde{\phi} \).

**Three levels of coarseness:** The weighted norm of \( \tilde{\phi} \) is 0.018941, and evaluation took 0.92 seconds. 243 combinations of parameters were evaluated.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>( \delta^* )</th>
<th>( \delta_m^* )</th>
<th>( E_m^* )</th>
<th>( E_q^* )</th>
<th>( V_t^* )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial Value</td>
<td>30</td>
<td>40</td>
<td>1.25</td>
<td>1.25</td>
<td>1.25</td>
</tr>
<tr>
<td>Suggested Value</td>
<td>24</td>
<td>30</td>
<td>1.5</td>
<td>1.5</td>
<td>1</td>
</tr>
<tr>
<td>Variation</td>
<td>-20%</td>
<td>-20%</td>
<td>+20%</td>
<td>+20%</td>
<td>-20%</td>
</tr>
</tbody>
</table>

Table 4.1: Data for three levels of coarseness for the first function.

**Four levels of coarseness:** The weighted norm of \( \tilde{\phi} \) is 0.018941, and evaluation took 3.47 seconds. 1024 combinations of parameters were evaluated.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>( \delta^* )</th>
<th>( \delta_m^* )</th>
<th>( E_m^* )</th>
<th>( E_q^* )</th>
<th>( V_t^* )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial Value</td>
<td>30</td>
<td>40</td>
<td>1.25</td>
<td>1.25</td>
<td>1.25</td>
</tr>
<tr>
<td>Suggested Value</td>
<td>24</td>
<td>30</td>
<td>1.5</td>
<td>1.5</td>
<td>1</td>
</tr>
<tr>
<td>Variation</td>
<td>-20%</td>
<td>-20%</td>
<td>+20%</td>
<td>+20%</td>
<td>-20%</td>
</tr>
</tbody>
</table>

Table 4.2: Data for four levels of coarseness for the first function.

**Five levels of coarseness:** The weighted norm of \( \tilde{\phi} \) is 0.018941, and evaluation took 9.93 seconds. 3125 combinations of parameters were evaluated.
\section*{4.1 First function}

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|c|c|}
\hline
Parameter & $\delta^*$ & $\delta_m^*$ & $E_m^*$ & $E_q^*$ & $V_t^*$ \\
\hline
Initial Value & 30 & 40 & 1.25 & 1.25 & 1.25 \\
\hline
Suggested Value & 24 & 30 & 1.5 & 1.5 & 1 \\
\hline
Variation & -20\% & -20\% & +20\% & +20\% & -20\% \\
\hline
\end{tabular}
\caption{Data for five levels of coarseness for the first function.}
\end{table}

It can be seen that Tables 4.1, 4.2, and 4.3 are identical. The only difference between the three runs is the time taken to produce the same result. There is no discernible change in the calculated optimal values of the five variables, because the possible values of these five variables are so limited. There are only three, four and five discreet values for each variable respectively, which means that the likelihood of any of these being the absolute best value for optimising the function is very low.

The next result is from evaluating the weighted norm of $\phi$ over a much larger coarseness, such as 25. This is a far more lengthy process, and even then, only solves for 25 discreet values of each parameter. The number of parameter combinations to be evaluated is $i = f^n = 25^5 = 9765625$. The computer can evaluate approximately 250 combinations per second, meaning that evaluation should take around 10 or 11 hours.

\subsection*{25 levels of coarseness:}

The weighted norm of $\phi$ is 0.018941, and evaluation took 45,196.71 seconds. 9,765,625 combinations of parameters were evaluated.

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|c|}
\hline
Parameter & $\delta^*$ & $\delta_m^*$ & $E_m^*$ & $E_q^*$ \\
\hline
Initial Value & 30 & 40 & 1.25 & 1.25 \\
\hline
Suggested Value & 24 & 30 & 1.5 & 1.5 \\
\hline
Variation & -20\% & -20\% & +20\% & +20\% \\
\hline
\end{tabular}
\caption{Data for 25 levels of coarseness for the first function.}
\end{table}

The results for 25 levels of coarseness are shown in Table 4.4, and they are identical to the previous three results table. By evaluating the function for more than 12 hours, the practical limit of exhaustive evaluation has been reached. The results do not change between a second and 12 hours of evaluation, The exhaustive evaluation benchmark is set at 0.018941. We shall aim to find a greater weighted norm of $\phi$ using a genetic algorithm.

\subsection*{4.1.2 Genetic algorithm evaluation}

Using the code shown in Appendix E.3, the function was evaluated. The five variables were initialised randomly between the minimum and maximum bounds, and over generations tended towards the value that gave the optimal weighted norm of $\phi$. This data was recorded and plotted on six separate axes – one for each parameter, and one for the weighted norm of $\phi$ (called $\phi_{\text{norm}}$ in the figure). This is shown in Figure 4.1.
4.2 Second function

4.2.1 Exhaustive evaluation

Much like the first function, this function aims to find the quantity

\[
\frac{||\tilde{\phi}||}{|| [\delta_1^* \delta_2^* \alpha^* E_{dr1}^* E_{dr2}^* E_{qr1}^* E_{qr2}^* V_c^* V_t^* s_1^* s_2^* K^* ] ||}
\]

which is hereafter referred to as the weighted norm of \(\tilde{\phi}\). In this function, there are twelve, rather than five, dynamic parameters. This quickly introduces the problem of calculation time, and for that...
reason, the function was only evaluated over three and four levels of coarseness, as the predicted time of evaluation of five was about around 339 hours, or just over 14 days. Because the evaluation of the weighted norm of $\tilde{\phi}$ over four levels of coarseness was predicted to take around 25 hours, the load was split over four computers running simultaneously. Unlike previously, on each of the four computers, the outermost for loop was replaced with a constant value, one for each value that would be tested with four levels of coarseness. This meant that even though the load was shared, every combination of parameters was still evaluated. The results for three and four levels of coarseness are shown in Tables 4.6 and 4.7.

Three levels of coarseness: The weighted norm of $\tilde{\phi}$ is 0.38744, and evaluation took 3410.12 seconds. 531,441 combinations of parameters were evaluated.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>$\delta_1^*$</th>
<th>$\delta_2^*$</th>
<th>$\alpha^*$</th>
<th>$E_{dr1}^w$</th>
<th>$E_{dr2}^w$</th>
<th>$E_{qr1}^w$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial Value</td>
<td>30</td>
<td>20</td>
<td>45</td>
<td>0.4</td>
<td>0.45</td>
<td>1.1</td>
</tr>
<tr>
<td>Suggested Value</td>
<td>24</td>
<td>16</td>
<td>36</td>
<td>0.32</td>
<td>0.36</td>
<td>0.88</td>
</tr>
<tr>
<td>Variation</td>
<td>-20%</td>
<td>-20%</td>
<td>-20%</td>
<td>-20%</td>
<td>-20%</td>
<td>-20%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter</th>
<th>$E_{qr2}^w$</th>
<th>$V_c^w$</th>
<th>$V_t^*$</th>
<th>$s_1^*$</th>
<th>$s_2^*$</th>
<th>$K^*$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial Value</td>
<td>1.1</td>
<td>1.36</td>
<td>1.2</td>
<td>0.25</td>
<td>0.25</td>
<td>0.65</td>
</tr>
<tr>
<td>Suggested Value</td>
<td>0.88</td>
<td>1.088</td>
<td>0.96</td>
<td>0.2</td>
<td>0.2</td>
<td>0.52</td>
</tr>
<tr>
<td>Variation</td>
<td>-20%</td>
<td>-20%</td>
<td>-20%</td>
<td>-20%</td>
<td>-20%</td>
<td>-20%</td>
</tr>
</tbody>
</table>

Table 4.6: Data for three levels of coarseness for the second function.

Three levels of coarseness: The weighted norm of $\tilde{\phi}$ is 0.38744, and evaluation took 94187.77 seconds. 16,777,216 combinations of parameters were evaluated.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>$\delta_1^*$</th>
<th>$\delta_2^*$</th>
<th>$\alpha^*$</th>
<th>$E_{dr1}^w$</th>
<th>$E_{dr2}^w$</th>
<th>$E_{qr1}^w$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial Value</td>
<td>30</td>
<td>20</td>
<td>45</td>
<td>0.4</td>
<td>0.45</td>
<td>1.1</td>
</tr>
<tr>
<td>Suggested Value</td>
<td>24</td>
<td>16</td>
<td>36</td>
<td>0.32</td>
<td>0.36</td>
<td>0.88</td>
</tr>
<tr>
<td>Variation</td>
<td>-20%</td>
<td>-20%</td>
<td>-20%</td>
<td>-20%</td>
<td>-20%</td>
<td>-20%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter</th>
<th>$E_{qr2}^w$</th>
<th>$V_c^w$</th>
<th>$V_t^*$</th>
<th>$s_1^*$</th>
<th>$s_2^*$</th>
<th>$K^*$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial Value</td>
<td>1.1</td>
<td>1.36</td>
<td>1.2</td>
<td>0.25</td>
<td>0.25</td>
<td>0.65</td>
</tr>
<tr>
<td>Suggested Value</td>
<td>0.88</td>
<td>1.088</td>
<td>0.96</td>
<td>0.2</td>
<td>0.2</td>
<td>0.52</td>
</tr>
<tr>
<td>Variation</td>
<td>-20%</td>
<td>-20%</td>
<td>-20%</td>
<td>-20%</td>
<td>-20%</td>
<td>-20%</td>
</tr>
</tbody>
</table>

Table 4.7: Data for four levels of coarseness for the second function.

4.2.2 Genetic algorithm evaluation

Using the code shown in Appendix E.4, the function was evaluated. Much like shown with the first function, the twelve variables were initialised randomly between the minimum and maximum bounds, and over generations tended towards the value that gave the optimal weighted norm of $\tilde{\phi}$. This data was recorded and plotted on thirteen separate axes – one for each parameter, and one for the weighted norm of $\tilde{\phi}$ (called $\phi_{norm}$ in the figure). This is shown in Figure 4.2.
Figure 4.2: The twelve variables and solution varying with each successive generations.
The weighted norm of $\hat{\phi}$ is 0.49521, and evaluation took 338.2 seconds. 36,000 individual chromosomes were evaluated.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>$\delta_1^*$</th>
<th>$\delta_2^*$</th>
<th>$\alpha^*$</th>
<th>$E_{dr1}'$</th>
<th>$E_{dr2}'$</th>
<th>$E_{qr1}'$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial Value</td>
<td>30</td>
<td>20</td>
<td>45</td>
<td>0.4</td>
<td>0.45</td>
<td>1.1</td>
</tr>
<tr>
<td>Suggested Value</td>
<td>24</td>
<td>16</td>
<td>36</td>
<td>0.507</td>
<td>0.454</td>
<td>0.88</td>
</tr>
<tr>
<td>Variation</td>
<td>-20%</td>
<td>-20%</td>
<td>-20%</td>
<td>+13.5%</td>
<td>+12.7%</td>
<td>-20%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter</th>
<th>$E_{qr2}'$</th>
<th>$V_{c}'$</th>
<th>$V_{t}'$</th>
<th>$s_1^*$</th>
<th>$s_2^*$</th>
<th>$K^*$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial Value</td>
<td>1.1</td>
<td>1.36</td>
<td>1.2</td>
<td>0.25</td>
<td>0.25</td>
<td>0.65</td>
</tr>
<tr>
<td>Suggested Value</td>
<td>1.1305</td>
<td>1.088</td>
<td>0.969</td>
<td>0.2</td>
<td>0.28</td>
<td>0.52</td>
</tr>
<tr>
<td>Variation</td>
<td>+2.72%</td>
<td>-20%</td>
<td>-19.2%</td>
<td>-20%</td>
<td>-12%</td>
<td>-20%</td>
</tr>
</tbody>
</table>

Table 4.8: Genetic algorithm evaluation for the second function.

### 4.3 Discussion

#### 4.3.1 Time

It has been established in Chapter 1 that the time taken to evaluate the function is critical. Figure 4.3 shows the time taken to evaluate a function five and 12 dynamic parameters if the rate of evaluation is approximately 300 per second for the first function, and 200 per second for the second function.

Figure 4.3: Time taken to exhaustively evaluate five and 12 dynamic parameters for varying levels of coarseness.
4.3 Discussion

For the first function, the mean evaluation time per chromosome is approximately 5.5 ms. This times
varies from run to run between about 4.5 ms and 7.5 ms. Because the two functions are very similar
(see \(C.1\) and \(C.2\) for the functions, or \(E.5\) and \(E.6\) for the MATLAB code), other than the number
of dynamic parameters in each, the evaluation time per chromosome is very similar. It ranged from
about 4.5 ms and 7 ms, with a mean of approximately 5.5 ms. The time difference between runs
can be accounted for by the varying load on the test computer’s CPU, caused by different programs
running simultaneously to MATLAB.

This is one advantage of genetic algorithms - while the time difference to compute the same level
of coarseness over five and 12 variables is tremendous, using genetic algorithms it is not. It makes
good sense to state that the time required for evaluation is based on complexity of the function,
rather than the number of variables. The evaluation time difference per chromosome between the two
function is negligible because the objective function receives the dynamic parameters within the script
as constants, and evaluates them for only a single iteration.

For the first function, given the current boundary conditions, the increase in time required to evaluate
the function to anywhere greater than 3 levels of coarseness is unnecessary. The solution is the same
for all levels of coarseness at least until 25, and for genetic algorithm evaluation. In this case, the
exhaustive evaluation method provides the solution that it faster both to set up and to evaluate.

If the population size is kept constant, then the number of generations that a genetic algorithm is run
for can be considered to be the same as the time it is run for, when the time taken per chromosome
and per generation is known. Therefore, the time required for each method can be compared directly.

4.3.2 Optimal value

The first function

The results of the exhaustive evaluation of the first function are shown in Tables 4.1, 4.2, 4.3 and 4.4.

As seen in Figure 4.1, the mean value of each of the five variables begin very close to the initial value
of that variable, represented by the red broken line at the centre of each graph. This is expected,
because the parameters are initialised over a range, the mean of which is the initial value given by
M.J. Hossain.

The algorithm determines that for a population of 120 chromosomes, the parameters \(\delta^*, \delta_m^*\) and \(V_{t}^*\) all
tend toward their minimum value (-20% of the initial value), as predicted by the exhaustive evaluation
that was carried out earlier. The broken black lines represent the upper and lower bounds of each
parameter. This occurs for each parameter after 70, 110 and 80 generations respectively.

It is worth noting that the two values \(E_{x_m}^*\) and \(E_{y_q}^*\) do not converge to a steady value, and further testing
for different population sizes and generation lengths (see ?? for supplementary results) indicates that
these values will not converge. According to Elsayed [8], 150 generations should be sufficient time for
variables in a problem such as this to reach their optimal value. After much effort and consultation
with Elsayed, the reason for the continued deviation could not be found. In the interests of prudent
allocation of project time, further investigation into the cause of the non convergence was abandoned.

The weighted norm of \(\tilde{\phi}\) is shown in the bottom-right graph of Figure 4.1. It shows that the mean
value quickly resolves to that value given by the exhaustive evaluation. This value is given by the
black broken line in the graph. The weighted norm of \(\tilde{\phi}\) never exceeds this line. Inspection of the
data obtained for the weighted norm of \(\tilde{\phi}\) indicates that the maximum value that can be found using
this particular genetic algorithm is also 0.018941, the same value that was found using exhaustive evaluation. According to M.J. Hossain [13], the returned values for the maximum weighted norm of $\tilde{\phi}$ are correct.

The fact that the two methods provide identical solutions is important to note. Noting the failure of the two variables to converge to a steady value, it can be seen that the solution stays constant once it has converged, despite the movement of all five variables. Based on this, it is reasonable to conclude that the variables that fail to converge are not significant in the evaluation of the maximum weighted norm of $\tilde{\phi}$, given the initial values and boundary conditions.

**The second function**

Similarly to the evaluation of the first function, the mean value of each of the twelve parameters begin very close to the initial value of that parameter. This can be seen in Figure 4.2. From this figure, the mean values of parameters $\delta_1^*, \delta_2^*, \alpha^*, E_{qr1}^*, V_{dc}^*, V_t^*$ and $K^*$ all converge toward the minimum allowable value of the parameter. This is in agreement with the prediction put forward by the exhaustive evaluation solution.

The mean values of parameters $E_{dr1}^*, E_{dr2}^*, E_{qr2}^*, s_1^*$ and $s_2^*$ all fail to converge. This is similar to behaviour seen in the first function, where two of the five variables did not resolve to a steady value. However, this did not prevent the solution from reaching its maximum value for the first function, nor for the second function. So for this function, as with the first, it can be seen that the solution stays constant once it has converged, despite the initial movement of all twelve variables, and the continued movement of the five variables identified in this paragraph. Therefore it can be said that these five parameters, given the initial values and bounds imposed on the function, do not exert a significant influence on the maximum value of the solution.

The bottom graph of Figure 4.2 shows the mean value of the weighted norm of $\tilde{\phi}$ as it evolves over subsequent generations. It quickly resolves to a value of 0.49521, which is significantly greater than the value given using exhaustive evaluation of 0.38744.

In contrast with the first function, the solution for the maximum weighted norm of $\tilde{\phi}$ is significantly better for genetic algorithm evaluation. It is also a far more complex function than the first function, which is likely a contributing factor for the superior performance of genetic algorithm evaluation compared to that of the first function.

**4.3.3 Computational resources**

Figure 4.3 illustrates the relationship between the time taken and number of iterations achieved. This also compares the two methods, and shows the reader the difference between the time taken to evaluate a given number of chromosomes to a given number of iterations. From earlier results, it is known that the time taken to evaluate a single iteration is very similar to that for a chromosome – this makes sense, because the operation is almost identical, with the exception of the method for choosing variables. The number of iterations is very similar to the number of chromosomes evaluated for a given period, and this gives rise to the question, which of the two give the better solution for the fewest number of operations. The data presented in this graph is not so useful on its own, but becomes very useful when viewed together with the data in Figure 4.5.

Figure 4.5 shows the relationship between the time taken and the solution returned. As time increases, it is expected that the best solution will also increase. The genetic algorithm evaluation (shown as a
broken black line) does improve with time, although the exhaustive evaluation solution does not. It is likely that given sufficient time this solution would improve, but the time constraints of this project do not allow for this. This figure does not illustrate a single run of a genetic algorithm, but rather a number of differently specified versions where the population size and generation length have been altered to take different times to evaluate.

These two figures demonstrate a key part of this thesis. The reader will note that the time axes for both graphs line up, so each graph can be visually compared. By doing so, the relationship between the number of operations and the solution returned can be established. The figures illustrate that for the first function, the exhaustive solution does not change over the period of $10^4$ seconds, or 5.5 hours. However, the genetic algorithm solution takes until approximately 5.1 seconds to reach the same as the exhaustive solution. This means that in this case, using a GA does not give a better solution than using exhaustive solution.

However, the second function presents us with quite different results, as the GA solution is much better than the exhaustive solution. The best solution is reached after running the GA for approximately 42 seconds. However, even running the GA for even a single second will still produce a great result than the predicted value of that returned using exhaustive evaluation over the period of a thousand or a million years. As with the first function, it is possible that a better solution might be found using exhaustive evaluation, but not in a practical time.

It should be noted that a moderate performance personal computer was used to conduct this research, and that in industrial or commercial applications, it is likely that a far more powerful and capable computer could be utilised to reduce the calculation times significantly.
§4.3 Discussion

Figure 4.4: Number of chromosomes and iterations evaluated over time.

Figure 4.5: The solution given for exhaustive and genetic algorithm evaluation over time.
Chapter 5

Conclusion and Recommendations for future Work

5.1 Conclusion

5.1.1 First function

The first function did not show any difference using exhaustive evaluation between 3 and 25 levels of coarseness, nor for any size population or generation time using the genetic algorithm. This result indicates that the time spent setting up the GA was not worthwhile, because the same answer could be achieved faster and with less effort using exhaustive evaluation. Research indicates that this situation is rare though, and that for similarly complex functions, genetic algorithm evaluation should far outperform exhaustive evaluation [5,6]. Subsequently, by altering the bounds or the initial values, the results could have been very different. However, the experimental results have been analysed and the conclusion must be that genetic algorithms are not worth the time and effort required to use them to evaluate a function like the first function in this thesis.

5.1.2 Second function

The second function was a different story. Using exhaustive evaluation, the suggested value of each parameter was -20% of its initial value. This gave a weighted norm of $\bar{\phi}$ of 0.38744, and this was the score to beat. Using genetic algorithms, however, this weighted norm of $\bar{\phi}$ was optimised at 0.4947, which represents a 27% increase from the value obtained using exhaustive evaluation.

This value is significant because it demonstrates the potential for increasing the maximum solution through use of genetic algorithms. It was desired that the maximum solution be as great as possible - the best solution is the largest one - therefore this increase clearly shows that genetic algorithms are a useful tool for optimisation of functions of similar complexity.

It is possible that, given sufficient time, the exhaustive evaluation may produce better results than were achieved in this thesis. In fact, the time taken to achieve the same results using exhaustive evaluation is estimated to require about 400 levels of coarseness, to achieve spacing of 0.1% between increments of variables. It is estimated that such a calculation would take approximately 8711 billion years to evaluate, and even then that is only predicted to match the result obtained about six or seven minutes by using a genetic algorithm. To potentially exceed it would take even longer, and there’s no guarantee that a greater solution could be found even if it were possible to wait this amount of time.
5.2 Future research and work

Alternate functions. One limitation of this thesis was the fact that only two functions were examined. By investigating different functions of varying complexity, sourced from different technical fields, a much more broad baseline could be established as to the advantage of implementing genetic algorithms for general evaluation. Based on the results shown in Figures 4.4 and 4.5, it can be concluded that this thesis did not examine a function that exhibited a crossover point, where the two methods returned the same answer for a specific time period, and before and after this period, different methods gave the better solution.

It should also be noted that it would be advisable to source alternate functions from outside the field of power system control design. This is to show that the technique is equally applicable in different technical fields, not just within power system control design.

Alternate boundary conditions and initial values. The boundary conditions and initial values that were used for the various dynamic parameters in this thesis are based on real life values that have been determined experimentally [12, 13]. By altering the bounds, either slightly or drastically, the results obtained could be dramatically different, but physically impossible and therefore useless. Further research may, however, indicate that the bounds are able to be increased, and this being the case, a return to these functions would be warranted.

Alternate parent selection and reproduction techniques. In this thesis, only basic techniques were applied. This was done with a view for simplicity, rather than accuracy. With greater time to work with, and a better understanding of the techniques that can be used for parent selection and reproduction, it may be possible to increase the efficiency of the evaluation, and potentially remove the problem of variables never converging on a steady value. This may well lead to better solutions, given the right conditions.

Different evaluation methods. This thesis has only examined one alternative method to exhaustive evaluation. Many others exist, that are potentially faster, more efficient and return greater solutions than genetic algorithms are able to. While this thesis did not examine them, they certainly present an interesting area for future research. Further research might compare more techniques against what genetic algorithms have been able to achieve, and make a further recommendation based on those findings.

5.3 Recommendations

Based on the experiments conducted, the data recorded and my conclusions, it is recommended that genetic algorithms be employed for evaluation of functions of a similar or greater complexity as the first and second functions in this thesis. The results that were obtained indicate that genetic algorithms are a potent technique for optimising or otherwise intelligently evaluating functions, and should be considered more widely in the future.