A Genetic Algorithm for Impedance Matching Network Design

by

Warren Paul du Plessis

Submitted in partial fulfilment of the requirements for the degree

Master of Engineering (Electronic)

in the Faculty of Engineering, Built Environment

and Information Technology

University of Pretoria

Pretoria

March 5, 2003

Supervisor: Professor P. L. D. Abrie

Summary

Keywords: Impedance matching, matching network, genetic algorithms, evolutionary computation, messy genetic algorithm, optimisation algorithms, microstrip, microstrip discontinuity, Pareto optimality, multi-objective optimisation

This dissertation considers the development of a new impedance matching algorithm. The main objectives were to develop an algorithm that has similar performance to published results, while overcoming the limitations of published algorithms.

This was achieved by using a genetic algorithm as the basis for the impedance matching algorithm. A number of modifications were made to the genetic algorithm to improve its performance. These modifications include the integration of a local optimiser, the use of Pareto optimality to simultaneously design networks of varying lengths, and the use of both binary and floating-point genetic operators.

The algorithm allows lumped, mixed, and distributed matching networks to be developed. The transmission line elements can either be perfect transmission lines or microstrip lines with both dispersion and discontinuities taken into account. Both the characteristic impedance and length of transmission lines are modified – something that other algorithms cannot do.

A large number of tests were performed on the algorithm to show the effects of the algorithm parameters, and to quantify the algorithm's performance. Comparisons between the new algorithm and published results show that the current algorithm has significant advantages over the real-frequency techniques, although it is not quite as good as the transformation-Q method. This is compensated for by the fact that this algorithm is more versatile than these methods.

Acknowledgements

I would like to thank the following people for their help and support, without which this dissertation would not have been possible:

- Prof. Pieter Abrie, my patient and knowledgeable supervisor.
- Jan Gutter, for tirelessly maintaining the computers used for the simulations in this dissertation despite my best efforts to destroy them.
- Justin Schoeman, for introducing me to the power of a UNIX command line, without which much of this dissertation would have been impossible.
- My family, for never giving up on me.
- My Lord Jesus Christ, without whom I wouldn't even be alive, let alone able to complete this work.

Contents

Chapter 1

Introduction

This document concerns the development of a new impedance matching algorithm. The problem will be placed in context and the motivation for developing a new impedance matching algorithm will be given in this chapter.

Section 1.1 will consider the reasons for the development of a new impedance matching algorithm in light of the current state of the art. Section 1.2 will then present the objectives for the algorithm developed during this dissertation.

1.1 Motivation

A large volume of work has been published on the problem of impedance matching and there has to be some doubt concerning whether yet another technique is really necessary. The motivation for developing a new impedance matching algorithm will be presented in this section with special attention being paid to the limitations of current techniques.

Impedance matching is a very important part of high frequency circuit design. Vendelin *et al.* [1] even go so far as to state "The most important design tool in amplifier and oscillator design is the concept of impedance matching." The importance of impedance matching arises because the gain available at high frequencies is limited, so it is important to ensure that the power loss due to impedance mismatches is kept to a minimum. The impedance presented to an active device also determines the noise, gain, output power, and stability of the device [2, 3].

The current state of the art in impedance matching is embodied in the real-frequency techniques developed by Carlin, Yarman and their co-workers [4–9]. These techniques optimise the transfer function of an impedance matching network and then extract components from the optimised transfer function. The main advantage of the real-frequency techniques is that they use the real measured impedances of the system, rather than an approximation to the system impedances like the earlier analytic techniques. The most important real-frequency techniques were published in 1977 by Carlin [4], 1982 by Yarman and Carlin [6, 7], and in 1983 by Carlin and Yarman [8]. In 1990, Yarman and Fettweis [9] published a simplified version of the algorithm developed by Carlin and Yarman [8]. The most recent real-frequency technique was published by Carlin and Civalleri [10] in 1992 and is an extension to Carlin's original work [4]. The fact that no papers concerning developments to the real-frequency techniques have been published in the last decade seems to indicate that these algorithms are mature and very little further development is possible. The main disadvantage of the real-frequency techniques is that the use of a transfer function means that a constant, corresponding to an ideal transformer, is usually present. Another disadvantage is that component ranges cannot be specified, leading to implementation difficulties because the component ranges that are available at high frequencies are limited by parasitics [2].

Recently, Abrie [2] published an impedance matching algorithm that uses the transformation-Q factor (also known as the node Q) that is applied to narrowband matching [2, 3]. The values of the transformation-Q factors are adjusted by an optimisation algorithm to obtain broadband networks. The main advantages of this technique are that ideal transformers are not required, resonant sections are possible, component ranges can be controlled, and distributed elements can be used.

While these techniques are extremely powerful, they do still have some limitations, limitations that this work seeks to address. These limitations are inadequate consideration of transmission lines, difficulties in handling more complex components like microstrip lines and discontinuities, and limitations in mixed-circuit synthesis.

The real-frequency techniques are limited to problems where commensurate lines (lines with the same fixed length) are used, and the published version of the transformation-Q algorithm does not allow the length and impedance of transmission lines to be varied simultaneously. This is a drawback because only half the available parameters are used, potentially ignoring good solutions.

Microstrip lines are a type of transmission line, but they do not operate in a perfect Transverse Electromagnetic (TEM) mode [11], leading to dispersion. This means that microstrip lines can only be approximated by perfect transmission lines at low frequencies. Additionally, the junctions between microstrip lines form discontinuities that have a significant effect on the network's transfer function. Most impedance matching algorithms assume that these effects are negligible and simply ignore them. Abrie's algorithm does compensate for discontinuities in the sense that attempts are made to limit their effects.

Mixed circuits use both lumped and distributed components. A major problem with this type of circuit is determining which circuit elements should be lumped and which should be distributed. This difficulty means that only a very limited class of mixed problems has been considered in the literature.

The algorithm developed during the course of this research overcomes all of these difficulties. Component ranges can be specified for lumped and distributed components. Transmission line networks can be synthesised by the algorithm, with both the length and impedance of the transmission lines being varied. The parasitic effects of microstrip lines and discontinuities are considered by the algorithm – the only limitation is the accuracy of the models used. Mixed circuits are considered in a completely general way and all combinations of lumped and distributed components are possible within the ladder structure considered.

The new algorithm developed during this dissertation is able to achieve all of these objectives by using a modified genetic algorithm. Many of the modifications used in this dissertation are unique and lead to greatly improved performance of the algorithm.

1.2 Objectives

The objectives for this dissertation are presented below and arise directly from the limitations of current impedance matching algorithms.

The major objective of this dissertation is to develop an impedance matching algorithm that can produce results comparable to the best current algorithms.

The second objective of this dissertation is to eliminate the problems associated with other algorithms. These problems include an inability to specify component ranges, restricted facilities for dealing with transmission lines, inadequate consideration of microstrip elements and discontinuities, and limited mixed-circuit capabilities.

These objectives will be achieved by using a modified genetic algorithm to design impedance matching networks. The main advantage of genetic algorithms is that they can be applied to any problem and should thus be able to overcome the limitations of other approaches.

A known limitation of genetic algorithms is that they converge very slowly, meaning that a large amount of computer time and processing power is required to produce a good result. Every effort will be made to ensure the algorithm is as efficient as possible to limit this problem. However, the quality of the final result will be considered more important than the time required to obtain that result, so this is a secondary objective.

1.3 Summary

The motivation for, and objectives of this dissertation were outlined in this chapter. The overriding objective was to develop an impedance matching network design algorithm that overcomes the limitations of current algorithms.

Chapter 2

Background

This chapter will review the theory used in this dissertation. Much of the work that follows will use information the information in this chapter or be motivated by properties of the techniques outlined below.

The presentation begins with a consideration of optimisation algorithms in Section 2.1 and then moves on to consider the special case of genetic algorithms in Section 2.2. A number of important impedance matching algorithms are reviewed in Section 2.3 and some important discontinuity models are covered in Section 2.4.

2.1 Optimisation Algorithms

The ultimate objective of Computer Aided Design (CAD) is a computer system that can undertake design problems with no input from humans apart from the required specifications. While this objective may still be some way off, the current state of the art is found in optimisation algorithms. Optimisation algorithms are computer programs that modify the parameters of a system to improve its performance. This section will give a very brief overview of some of the most important classes of optimisation algorithm available today. This presentation draws heavily on the review papers by Bandler [12] and Charalambous [13], the very practical presentation by Press *et al.* [14], and the detailed review of the field by Nocedal and Wright [15]. Some of the more advanced algorithms will not be considered here because the extra complexity required is not justified for this dissertation.

Section 2.1.1 will define the most important terms and concepts used in optimisation. Local optimisation in one dimension (line searches) will be covered in Section 2.1.2. Section 2.1.3 considers the most important classes of local optimisation techniques. Section 2.1.4 will present some of the main global optimisation techniques. Section 2.1.5 will give a brief overview of combinatorial optimisation. Some hybrid methods that combine more than one type of optimisation algorithm are considered in Section 2.1.6. Lastly, some methods for dealing with constraints are presented in Section 2.1.7.

2.1.1 Important Concepts

Optimisation is the process of adjusting the values of the parameters of a system so as to minimise or maximise some measure of the system's performance. The function used as a measure of a system's performance is known as the objective function. This section will only consider minimisation problems explicitly. This does not result in a loss of generality because maximisation problems can be formulated as minimisation problems by multiplying the objective function value by -1 [15].

Section 2.1.1.1 presents the basic formulation of an optimisation problem with minimax optimisation being considered in Section 2.1.1.2. Lastly, multi-objective optimisation is covered in Section 2.1.1.3.

2.1.1.1 Problem Formulation

This section will present the basic formulation of an optimisation problem, introduce important terminology, and consider some important special cases. Most of the work in this section is derived from Press *et al.* [14].

The mathematical form of the optimisation problem is

$$f(\mathbf{x}) \to \min \tag{2.1}$$

where $\mathbf{x}$ is some structure (usually a vector or a matrix) whose elements correspond to the system parameters, and $f(\mathbf{x})$ is the objective function which maps the system parameters to some measure of system performance that must be minimised. This case is known as unconstrained optimisation because there are no constraints on how the parameters' values can be assigned.

The simplest differentiable function that can have a well-defined minimum is a quadratic function [13]. The Taylor series expansion of a function is

$$f(\mathbf{x}+\Delta\mathbf{x}) = f(\mathbf{x}) + \sum_i \frac{\partial f}{\partial x_i}\Delta x_i + \frac{1}{2}\sum_{i,j}\frac{\partial^2 f}{\partial x_i \partial x_j}\Delta x_i \Delta x_j + \ldots \tag{2.2}$$

where $\Delta\mathbf{x}$ is the change in $\mathbf{x}$, x_i is the value of component i of $\mathbf{x}$, and Δx_i is the change in x_i. When the higher order terms are ignored and matrix representation is used, (2.2) can be rewritten as

$$f(\mathbf{x}+\Delta\mathbf{x}) \approx c + \mathbf{b}^T \cdot \Delta\mathbf{x} + \frac{1}{2}\Delta\mathbf{x}^T \cdot \mathbf{A} \cdot \Delta\mathbf{x} \tag{2.3}$$

where

$$c \equiv f(\mathbf{x}) \qquad \mathbf{b} \equiv \nabla f|_{\mathbf{x}} \qquad [\mathbf{A}]_{ij} \equiv \left.\frac{\partial^2 f}{\partial x_i \partial x_j}\right|_{\mathbf{x}} \tag{2.4}$$

and all matrices except $\mathbf{A}$ are column matrices. The matrix $\mathbf{A}$, which contains the second partial derivatives of the objective function at $\mathbf{x}$, is known as the Hessian matrix of the function and is symmetrical. If the first and second derivatives of the objective function exist, the gradient vector is zero, and the Hessian matrix is positive definite then there is a minimum at point $\mathbf{x}$ [12]. The quadratic approximation will be used later in connection with conjugate directions, and the Newton and quasi-Newton methods.

Constrained optimisation is the case where parameters or functions of the parameters are constrained to be within some range or equal to a specified value. This can be written as

$$\begin{aligned} g_i(\mathbf{x}) &\geq 0 \\ h_j(\mathbf{x}) &= 0 \end{aligned} \tag{2.5}$$

where $g_i(\mathbf{x})$ and $h_j(\mathbf{x})$ are both functions of the system parameters, and the subscripts indicate that there can be more than one of each type of constraint. Note that no generality is lost by only considering cases with zero because constants can be included in the functions. The situation where a function is less than or equal to a constant can be converted to the case where the function is greater than or equal to a constant by multiplying by -1. The subset of the problem space where all the constraints are satisfied is known as the feasible region.

Various special cases of objective function and its value can now be defined. A local minimum occurs when the value of the objective function is lower than all other objective function values in a subset of the feasible region. The point that has the lowest objective function value in the feasible region is known as the global minimum. Most optimisation algorithms consider the case where the parameters are continuous variables, but optimisation problems that require the best combination or permutation of parameters are also possible. These problems are known as combinatorial optimisation problems. A good example of a combinatorial problem is the classic travelling salesman problem where a salesman has to visit every city in a given area once and only once while minimising the total distance travelled. Other special cases are minimax and multi-objective optimisation which will be considered in the next two sections.

2.1.1.2 Minimax Optimisation

Minimax optimisation considers the problem of minimising the maximum value of an objective function. This is an extremely important problem which is very common in engineering where specifications usually have to be satisfied in a minimax way. The basic formulation of a minimax problem along with an example and some difficulties with minimax objective functions are considered below.

A minimax problem is one with an objective function of the form

$$M(\mathbf{x}) = \max_{i \in I} [f_i(\mathbf{x})] \tag{2.6}$$

where $M(\mathbf{x})$ is the objective function, $f_i(\mathbf{x})$ are sub-functions, and I is the set of all sub-functions. The sub-functions in minimax problems are normally the value of the objective function at a number of values of some independent variable like frequency or temperature.

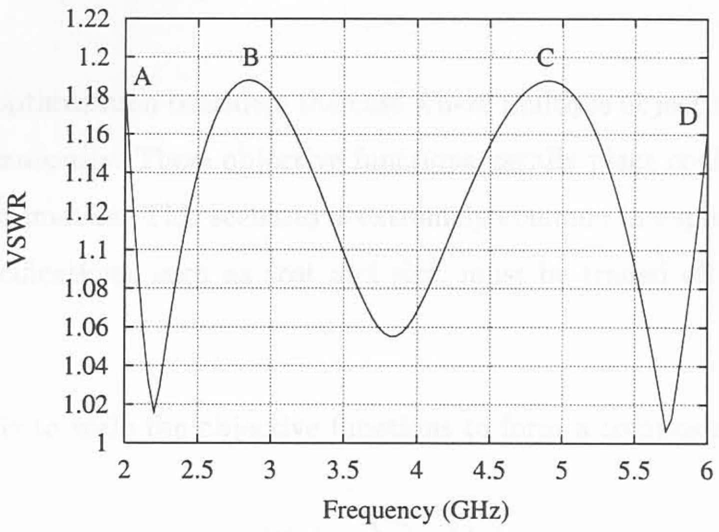
An example of a sixth-order minimax matching problem taken from Abrie [2] is given in Figure 2.1. Figure 2.1(a) shows the how the VSWR of a matching network varies with frequency and Figure 2.1(b) shows the effect of varying one of the component values around its optimum value. One of the four local maxima (A, B, C, and D in Figure 2.1(a)) will determine the minimax VSWR value of the system. This is shown in Figure 2.1(b) where distinct kinks are observed at A and B as different local maxima start to dominate the objective function.

The major difficulty with minimax problems is that the gradient is discontinuous meaning that many useful approximations such as the Taylor expansion are not valid. Gradient discontinuities are clearly seen at points A and B in Figure 2.1(b). It is also clear that a parabolic approximation to the curve in Figure 2.1(b) would be extremely poor, especially if all the points used to form the approximation were before A, between A and B, or after B.

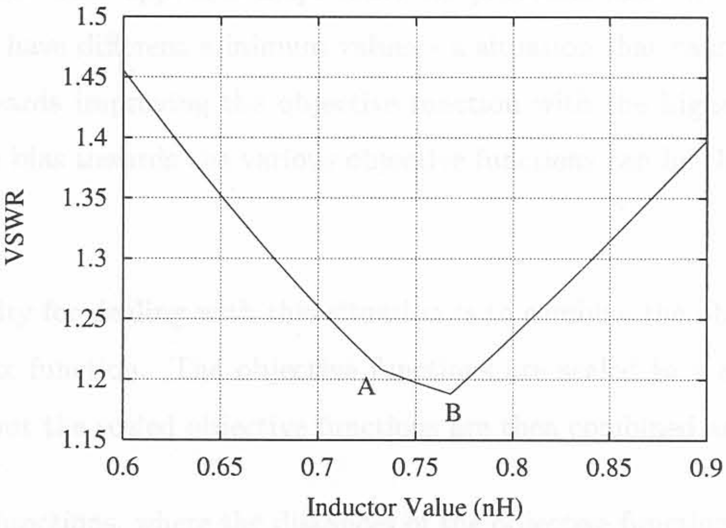
Charalambous [13] reviews a method for converting a minimax problem to a form where the objective function is smooth and has continuous gradients. This conversion is based on

$$M(\mathbf{x}) = \lim_{p \to \infty} \left([f_i(\mathbf{x})]^p\right)^{1/p} \tag{2.7}$$

where $M(\mathbf{x})$ is the minimax error. The main difficulty with the formulation given in (2.7) is that p must be very large for the approximation to be accurate, leading to ill-conditioned problems due to the finite precision of a computer. The techniques reviewed by Charalam-



(a) Frequency response.



(b) Variation with one variable.

Figure 2.1: Minimax Function.

bous [13] are significantly more powerful than (2.7) and relax the requirement for very large values of p.

2.1.1.3 Multi-Objective Optimisation

Multi-objective optimisation considers the case where multiple objective functions must be optimised simultaneously. These objective functions usually place conflicting requirements on the system parameters. This scenario is extremely common in engineering design where a number of specifications, such as cost and size, must be traded off to arrive at a good solution.

The first option is to scale the objective functions to form a composite objective function of the form

$$P(\mathbf{x}) = \sum_{i \in I} a_i f_i(\mathbf{x}) \tag{2.8}$$

where a_i are weights used to scale the objective functions $f_i(\mathbf{x})$, and $P(\mathbf{x})$ is the composite objective function. This approach helps limit the problems that can arise when the objective functions have different minimum values - a situation that causes the optimisation to be biased towards improving the objective function with the highest minimum value. Additionally, the bias towards the various objective functions can be changed by adjusting the weights.

Another possibility for dealing with this situation is to combine the objective functions to create a minimax function. The objective functions are scaled in a similar way to that shown in (2.8), but the scaled objective functions are then combined according to (2.6).

Fuzzy objective functions, where the distances of the objective functions' values from their minima are combined using fuzzy logic, can also be used [16]. This approach requires a knowledge of the optimum value of each objective function, but this is comparatively simple to calculate by solving the problem once for each objective function while ignoring the others.

When specifications for the worst acceptable value of each objective function exist, one of the objective functions can be optimised while the others are considered as constraints. This is a particularly useful approach where the primary requirement is to minimise one of the objective functions, subject to the other objective functions meeting their specifications.

The last approach to multi-objective optimisation that will be considered here uses the concept of Pareto optimality. Pareto optimality [17] provides a way of ranking a number of solutions to an optimisation problem. This means that Pareto optimality is difficult to apply to algorithms that only process one solution, and is much better suited to algorithms where a number of solutions exist such as genetic algorithms. Pareto optimality relies on the concepts of dominated and non-dominated solutions to a problem. A solution to a problem dominates a second solution if the first solution is at least as good as the second solution for every objective function, and the first solution is better than the second for at least one objective function. In mathematical terms this means that solution $\mathbf{x}_1$ dominates solution $\mathbf{x}_2$ if

$$\forall i \in \{1, 2, \ldots, n\}, f_i(\mathbf{x}_1) \leq f_i(\mathbf{x}_2) \wedge \exists i \in \{1, 2, \ldots, n\}, f_i(\mathbf{x}_1) < f_i(\mathbf{x}_2) \,. \qquad (2.9)$$

Pareto optimality is implemented by finding all non-dominated solutions in the current set of solutions and giving them an objective function value. Those individuals are then removed from consideration and the next layer of non-dominated solutions are found. These solutions are given a worse objective function value than the previous layer and the process is then repeated until all solutions have been considered. The major advantage of an algorithm that uses Pareto optimality is that a number of results presenting various compromises between the objective functions are available.

Borghi *et al.* [16] consider multi-objective optimisation applied to Loney's solenoid problem which has two objective functions related to a solenoid's fields. The weighted objective functions, fuzzy, constraint, and Pareto approaches were tested and compared. The best results were obtained in the weighted and Pareto cases with the Pareto case requiring more function evaluations. These extra function evaluations are compensated for by the fact that the Pareto case gives the designer a large number of options to choose from.

2.1.2 Line Searches

Local optimisation is the process of taking a good starting point and finding the best result near that point. Line search algorithms are a special case of local optimisation where the optimisation takes place in only one dimension. This is a very important case that forms an integral part of many of the multi-dimensional algorithms described in Section 2.1.3. Many of the problems encountered in line searches are also applicable to the higher dimension cases considered later, so the study of line searches is extremely useful. This section will review the most important topics dealing with line searches.

2.1.2.1 Bracketing an Optimum

All the line search methods considered in subsequent sections assume that an interval with one and only one optimum is available. This is not always the case and a method is required to initially bracket a unique minimum before a line search can be used. This bracketing problem has not been extensively considered [14], but some algorithms are available.

One bracketing technique is to fit a parabolic function through three points and then to choose a point beyond the parabolic function's optimum. This procedure can be repeated until an optimum is bracketed. Care must be taken to ensure that a maximum is not bracketed when a minimum is required (or vice versa, as the case may be).

Another possibility is to use some heuristic technique to search along a line until an optimum is bracketed. The most common way of doing this is to move in the direction of the line search with steps of increasing size until an optimum is bracketed. An optimum is bracketed when the middle point has a lower objective function value than the first and last points, or two points have gradient values with different signs.

The main problem with these approaches is that it is possible to bracket more than one minimum and the line search methods considered below require an interval with a unique minimum.

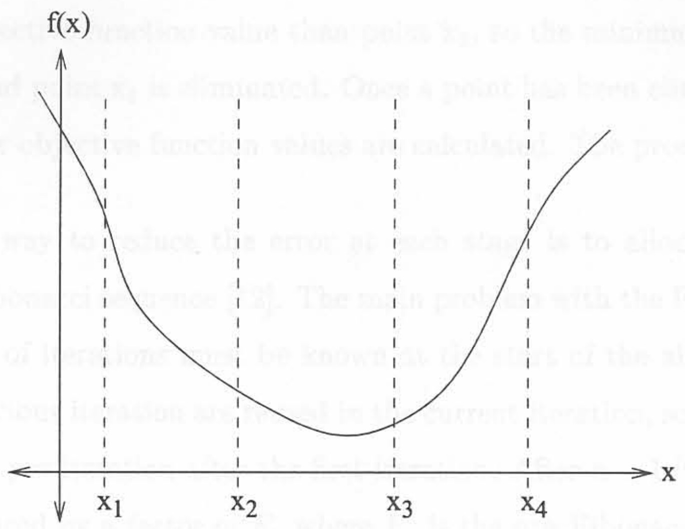


Figure 2.2: Fibonacci line search.

2.1.2.2 Direct Elimination Methods

Direct elimination methods operate by reducing the range around a minimum at each iteration of the algorithm. This means that the algorithm will monotonically converge to an optimum. Direct elimination methods only use the values of the objective function and do not require gradient or any other information about a function. This means that direct elimination methods have the advantage that they can be used on any objective function. The disadvantage of this approach is that useful information such as the gradient of a function is ignored even when it is available.

It is very important that a minimum is bracketed before using a direct elimination method because otherwise the algorithm can converge to an incorrect result. It is also very important that the bracketed range includes only one minima for the same reason.

Direct elimination methods work with four points because it is impossible to determine whether the upper or lower part of a range must be eliminated with only three points. During each iteration the two interior points' values are used to determine whether the first or last point of the current range must be eliminated. For example in Figure 2.2, point

x_2 has a higher objective function value than point x_3, so the minimum must be between points x_2 and x_4, and point x_1 is eliminated. Once a point has been eliminated, new points are chosen and their objective function values are calculated. The procedure then repeats.

The most efficient way to reduce the error at each stage is to allocate the four points according to the Fibonacci sequence [12]. The main problem with the Fibonacci line search is that the number of iterations must be known at the start of the algorithm. The three points from the previous iteration are reused in the current iteration, so only one new point has to be evaluated per iteration after the first iteration. After $n-1$ iterations, the initial search range is reduced by a factor of F_n where F_n is the nth Fibonacci number.

The golden ratio search is very similar to the Fibonacci algorithm except that the points are chosen according to the golden ratio. As with the Fibonacci line search, only one new point has to be evaluated at each iteration after the first iteration. After $n-1$ iterations, the initial search range is reduced by a factor of τ^{n-1} where τ is the golden ratio. The advantage of this formulation is that the number of steps does not have to be known in advance because the use of the golden ratio ensures that iterations can continue forever if rounding errors can be ignored. The penalty is that the golden ratio search is not quite as efficient as the Fibonacci algorithm, and will have an error approximately 17% worse than the Fibonacci algorithm after a large number of iterations [12].

The golden ratio line search is generally favoured over the Fibonacci line search because the number of iterations does not have to be known before the line search commences. This is a major advantage where the line search is done to a specified precision and the initial interval size is not fixed.

2.1.2.3 Interpolation Methods

Interpolation methods approximate the function by a low order polynomial and calculate the minimum of that polynomial to find the next point in the line search. The advantage of this approach is that convergence can be greatly accelerated.

The convergence can become quadratic when a parabolic interpolation function is used; that is the precision of the solution doubles at each iteration [14]. The main difficulty with interpolation methods is that they only work well for smooth functions and can actually be slower than direct elimination methods in some cases. Care must also be taken to ensure that the algorithm converges to a minimum and not a maximum (or vice versa, as the case may be). More information on interpolation methods can be found in Press *et al.* [14] and Jang *et al.* [18].

2.1.2.4 Line Searches with Gradients

The direct elimination and interpolation line search algorithms ignore gradient information. While this approach is justified when gradient information is not available, it does not make sense to ignore useful information about a function when it is available.

Press *et al.* [14] advocate a conservative approach based on the bisection method [19]. Three points are used, with the gradient at the centre point being used to determine whether the upper or lower portion of the range must be eliminated. This technique will halve the range at each step, representing a tremendous advantage over the direct elimination methods considered above. The initial range is reduced by a factor of 2^n after n iterations [19].

Jang *et al.* [18] suggest using Newton's method or the secant method [19] to find the point where the gradient is zero. Newton's method uses the first and second derivatives of the objective function to rapidly converge to the minimum. This approach has the disadvantages that the second derivatives must be calculated, and it can fail to converge. The secant method is similar to Newton's method except that it uses the last two points tested to generate an approximation to the second derivative of the objective function. The secant method's convergence rate can be faster or slower than the bisection method depending on the objective function, and the secant method can fail to converge for some problems [14].

The bisection and secant methods are recommended over Newton's method because they do not require second derivatives. The convergence rate of the secant method will generally, but not always, be better than the bisection method. The bisection method has the advantages that it is simple to implement, and the number of iterations to achieve a specified precision can be calculated.

2.1.3 Local Optimisation

Local optimisation is the process of taking a good starting point and finding the best result near that point. The performance of a local optimisation algorithm thus depends heavily on the starting point chosen. This section will review the most important classes of local optimisation algorithms.

Simplex, direction set, and gradient methods will be considered in Sections 2.1.3.1 to 2.1.3.3. The gradient methods considered include steepest descent, Newton's method and quasi-Newton methods, conjugate gradient methods, and trust region methods.

2.1.3.1 Simplex Methods

Simplex methods are heuristic optimisation algorithms that do not require gradient evaluations or line searches. However this does not mean that they require less function evaluations than other algorithms, in fact the opposite is often true [14], but simplex methods can be used to quickly arrive at a useful solution.

For a problem with n variables, a simplex method requires $n+1$ points chosen so as to give a non-zero volume. The algorithm proceeds by modifying the position of the points in an attempt to improve the objective function value. The most popular simplex method is the Nelder-Mead algorithm [20], but other options do exist [21].

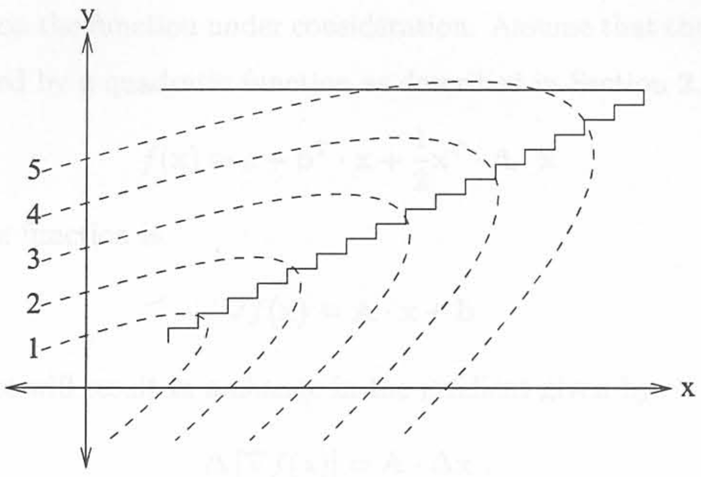


Figure 2.3: Optimisation by line searches along coordinate directions.

2.1.3.2 Direction Set Methods

This section will consider optimisation techniques that use line searches, but do not require gradient information. The most important concept in this section is the notion of conjugate directions. This concept also forms the basis of the conjugate gradient techniques covered later. The information in this section is adapted from Press *et al.* [14].

Line searches are a very powerful method for optimising in one dimension. Unfortunately the principles applied in line searches are not directly applicable to higher order problems. This leads to the possibility of using line searches as part of an optimisation algorithm that is applicable to high order problems. The main difficulty is choosing the line search directions. One possibility is to optimise each variable in turn, but this approach is very inefficient in cases like the one shown in Figure 2.3. The main problem with this approach is that each line search spoils the results of the previous line searches so the algorithm zig-zags down the valley, converging towards the optimum very slowly. This problem can be overcome by using previous results to move down the valley by searching in directions that do not interfere.

Directions that do not interfere are known as conjugate directions. Which directions are conjugate depends on the function under consideration. Assume that the objective function can be approximated by a quadratic function as described in Section 2.1.1.1.

$$f(\mathbf{x}) \approx c + \mathbf{b}^T \cdot \mathbf{x} + \frac{1}{2}\mathbf{x}^T \cdot \mathbf{A} \cdot \mathbf{x} \tag{2.10}$$

The gradient of this function is

$$\nabla f(\mathbf{x}) = \mathbf{A} \cdot \mathbf{x} + \mathbf{b} \tag{2.11}$$

and any change in $\mathbf{x}$ will result in a change in the gradient given by

$$\Delta \left[\nabla f(\mathbf{x})\right] = \mathbf{A} \cdot \Delta\mathbf{x} \ . \tag{2.12}$$

Assume that a line search has found a optimum in some direction $\mathbf{u}$ and a line search in some direction $\mathbf{v}$ must now take place. The directions $\mathbf{u}$ and $\mathbf{v}$ are conjugate if the gradient change due to the second line search is orthogonal to the first line search direction. Two vectors are orthogonal when their scalar product is zero giving

$$0 = \mathbf{u} \cdot \Delta \left[\nabla f(\mathbf{x})\right] \ . \tag{2.13}$$

Now the gradient change can be replaced by the result in (2.12) giving

$$0 = \mathbf{u} \cdot \mathbf{A} \cdot \mathbf{v} \ . \tag{2.14}$$

Thus two directions $\mathbf{u}$ and $\mathbf{v}$ are said to be conjugate when (2.14) holds. Obviously this result is only strictly valid when the objective function is quadratic, but in practice it is a good approximation to most smooth functions.

The main task of a direction set method is thus to generate a set of conjugate directions. Once a set of conjugate directions are found, it can be shown that convergence will be quadratic. A more complete discussion of conjugate direction set methods is given by Powell [21].

2.1.3.3 Gradient Methods

This section will consider optimisation algorithms that use gradient information to improve convergence rate. Obviously these algorithms can only be used where gradient information is available or can be inferred from other calculations. The information in this section is based on Press *et al.* [14].

The presentation will start by considering some methods to obtain the gradient of a function. The optimisation algorithms considered are the steepest descent algorithm, Newton's method and quasi-Newton methods, conjugate gradient algorithms, trust region methods, and algorithms that use gradient information, but do not require line searches.

Calculating the Gradient

Some methods for calculating the gradient will be considered in this section. Obviously the simplest approach is to perform symbolic differentiation of the objective function and supply this gradient function to the algorithm. When this is not possible, finite differences or automatic differentiation can be used. The information in this section is obtained from Nocedal and Wright [15].

Finite difference methods are based on the definition of a derivative given in (2.15).

$$\frac{\partial f}{\partial x_i} = \lim_{h_i \to 0} \frac{f(x_i + h_i) - f(x)}{h_i} \quad \text{for} \quad i = 1, 2, \ldots, n \tag{2.15}$$

where h_i is some deviation in direction x_i, and n is the number of variables. By removing the limit in (2.15), the gradient can be approximated by

$$\frac{\partial f}{\partial x_i} \approx \frac{f(x + \Delta d_i) - f(x)}{\Delta d_i} \quad \text{for} \quad i = 1, 2, \ldots, n \tag{2.16}$$

where Δd_i is some small deviation in direction x_i. This approach in known as the finite difference method because finite differences are used instead of the infinitesimal differences required in (2.15). The main implementation difficulty is to determine the size of Δd_i so that it is small enough to give a good approximation, but not so small that the finite precision of the computer introduces errors. The main problem with finite-difference methods is that a function with n variables requires n function evaluations to calculate the gradient (assuming that the function value at the current point is already known). A more detailed discussion of finite difference methods can be found in Mathews [19].

17224287
b16254806

Automatic differentiation algorithms rely on the fact that any function consists of a number of simple sub-functions. The chain rule can then be used in conjunction with the derivatives of these simple sub-functions to calculate the derivative of the complete function. The result of this approach is a gradient function which is then used to compute gradients.

Steepest Descent Algorithm

The steepest descent algorithm is the simplest gradient-based optimisation algorithm. The principle is to perform a line search in the direction of steepest descent from the current point. It would seem that this algorithm should perform very well, but this is not the case.

The direction of steepest descent is the opposite direction to the gradient vector. When a line search has been completed, the optimum in the direction of the line search has been found, so the component of the gradient in that direction must be zero. This means that the gradient at the end of a line search will be perpendicular to the direction of the line search, and each line search performed by the steepest descent algorithm will thus be perpendicular to the previous line search. The objective function shown in Figure 2.4 has a narrow valley and the steepest descent algorithm performs very poorly because it zig-zags across the valley rather than moving along it. This problem is seen to be very similar to that shown in Figure 2.3. For this reason steepest descent is not a popular optimisation algorithm.

The major advantages of the steepest descent algorithm are that no data has to be stored between steps and second derivatives are not required.

Newton's Method

Newton's method works by finding the minimum of the quadratic approximation to a function. The formulation allows rapid quadratic convergence, but requires a large number of gradient calculations.

Section 2.1.1.1 shows that any function can be approximated as a quadratic function by

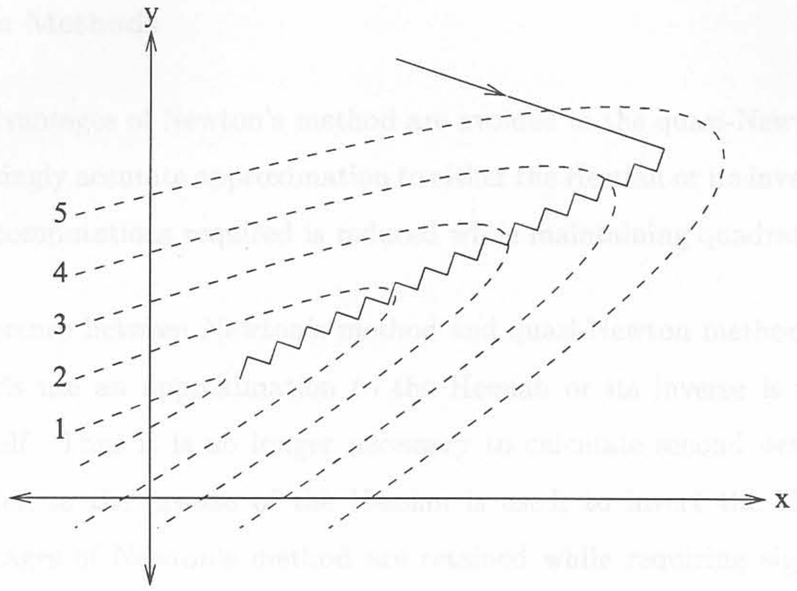


Figure 2.4: Steepest descent algorithm in a narrow valley.

using the Taylor expansion as shown in (2.3). The gradient of this function is

$$\nabla f(\mathbf{x} + \Delta\mathbf{x}) = \mathbf{b} + \mathbf{A} \cdot \Delta\mathbf{x} \ . \tag{2.17}$$

At an optimum the gradient will be equal to zero, so the step to reach the optimum ($\Delta\mathbf{x}$) is given by

$$\Delta\mathbf{x} = -\mathbf{A}^{-1} \cdot \mathbf{b} \ . \tag{2.18}$$

The next optimisation step is thus calculated from the inverse of the Hessian and the gradient.

There are a number of problems with Newton's method. The first problem is that the Hessian and its inverse must be calculated at each step, requiring a large amount of processing power and storage. Newton's method can fail to converge or can converge to an incorrect result when the Hessian is not positive definite. Modifications to the basic version of Newton's method that overcome these difficulties do exist [15], but the quasi-Newton methods are significantly more popular.

Quasi-Newton Methods

The main disadvantages of Newton's method are avoided in the quasi-Newton methods by using an increasingly accurate approximation to either the Hessian or its inverse. In this way the number of computations required is reduced while maintaining quadratic convergence.

The major difference between Newton's method and quasi-Newton methods is that quasi-Newton methods use an approximation to the Hessian or its inverse is used instead of the Hessian itself. Thus it is no longer necessary to calculate second derivatives, and if an approximation to the inverse of the Hessian is used, to invert the Hessian. In this way the advantages of Newton's method are retained while requiring significantly fewer computations.

Once an approximation to the Hessian or its inverse has been formed, quasi-Newton methods use the Newton step given in (2.18) as a line search direction. Unlike other algorithms, the line search does not have to be highly accurate [12, 14].

The major disadvantages of quasi-Newton methods is that a large matrix must be manipulated and this can cause difficulties with problems with surprisingly few variables. The main advantages of quasi-Newton methods are rapid quadratic convergence (faster than the conjugate gradient methods [15]) and the fact that accurate line searches are not required. This means that the number of function evaluations required by the line search algorithm can be reduced. More information on quasi-Newton methods is given by Charalambous [13], Press *et al.* [14], and Nocedal and Wright [15].

Conjugate Gradient Algorithms

The conjugate gradient algorithms extend the notion of conjugate directions discussed in Section 2.1.3.2 to the case where gradient information is available. This results in a dramatic acceleration of the convergence because the extra information provided by the gradient is effectively used.

As with the direction set methods considered in Section 2.1.3.2, the main problem is to establish a set of mutually conjugate directions. The basic principle is to use the previous search direction and the gradient at the current point to determine the next search direction.

The major advantages of conjugate gradient techniques are that only the last search direction needs to be stored, and they converge quadratically (although slower than the quasi-Newton methods [15]). More information on conjugate gradient techniques is given by Press *et al.* [14], Nocedal and Wright [15], and Jang *et al.* [18].

Trust-Region Methods

Trust-region methods use similar principles to Newton's method and quasi-Newton methods. The difference arises because trust-region methods do not use line searches, but instead search within some volume round the current point.

Trust region methods are based on the assumption that any approximation to a problem will be most accurate near the current point. The region where a model is assumed to be accurate, a trust-region, is thus established round the current point. The most common way to do this is to have a radius which is adjusted after each step depending on how accurate the approximation proved to be. The radius is increased when the model proves to be accurate and decreased when the model is inaccurate.

A number of steps are performed for each iteration of a trust-region method. The first step is to calculate the minimum value of the approximation within the trust-region. The value at this point is then compared to the objective function value and the point is accepted if the agreement is satisfactory. If the agreement is unsatisfactory, the new point is discarded, the trust region volume decreased, and the process restarted. The Hessian can be computed directly as in Newton's method, an approximation can be constructed in the same way as the quasi-Newton method, or some other method can be used. Methods for finding the minimum of the approximation within the trust-region are considered by Nocedal and Wright [15].

The most important advantage of trust-region methods is that the Hessian is not always required to be positive-definite. This means that problems that cannot be solved with other methods can be solved by trust-region methods.

Methods That do not Require Line Searches

Apart from the trust region methods, all the methods considered so far require line searches to determine the size of movement in the search direction. Some algorithms that do not require line searches are briefly considered in this section.

Jang *et al.* [18] present a number of methods based on steepest descent that do not require line searches. These methods work by taking a single step in the search direction. The size of the step can be constant or can be adjusted by a heuristic to obtain faster convergence.

The Leap-Frog algorithm developed by Snyman [22–24] is another example of a local optimiser that does not require line searches. The leap-frog optimisation algorithm is based on the movement of a particle in a conservative force field. Such a particle will attempt to minimise its energy by moving to a point with lower energy. Friction is ignored by the algorithm, so a heuristic braking technique is used to ensure that the particle eventually settles at a local optimum. The major advantage of the Leap-Frog algorithm is that it is very robust – it is able to solve problems that other algorithms cannot. Some of the reasons for this robustness are the fact that it is not based on a quadratic approximation to the problem, and the absence of line searches allows it to cope with noisy objective functions.

2.1.4 Global Optimisation

The majority of practical optimisation problems have a number of local optima with the best local optimum being known as the global optimum. Local optimisation techniques can only give a good result if the starting point is near a good solution. For the purposes of this dissertation, global optimisation is defined as the process of finding a starting point for a local optimisation algorithm that is near the global optimum or an optimum with a similar objective function value to the global optimum.

The main problem with global optimisation is that there is no simple approximation like the quadratic approximation for local optimisation. This means that global converge criteria cannot be mathematically derived, and global optimisation algorithms are thus heuristic. The quality of the final result will improve as the number of points sampled increases until the global optimum is found.

A reasonably intuitive approach is to use an exhaustive search. A grid is applied to the problem and each point on the grid is sampled and compared to other points. The best point can the be used either as the global optimum or as the starting point for a local optimiser. Other options are to apply a local optimiser to every grid point or a subset of the grid points. The choice of grid size is critical because a very fine grid will take a long time to sample, but a very coarse grid could easily miss a good solution. The major problem with an exhaustive search is the extremely large number of points that must be sampled. This has led to the development of the heuristic techniques below.

Clustering methods [25] reduce the number of points that have to be passed through a local optimiser by identifying points that are near the same local optimum. The reduces the unnecessary extra computations that result whenever a local optimum is found more than once.

Simulated annealing attempts to emulate the annealing process in materials [14]. A material is heated to a high temperature and then slowly cooled during annealing. Large changes in crystal structure can occur when the temperature is high, but much smaller changes occur at low temperatures. The crystal structure of the material is in a much lower energy state (fewer dislocations and grains) after annealing than before annealing [26] so this approach forms a good basis for an optimisation algorithm. This is done by allowing large changes near the beginning of a run when the current solution is probably far from a minimum, and allowing only small changes near the end of a run when the current solution is assumed to be close to a minimum. The search space is thus efficiently sampled near the start of a run and good solutions are well optimised near the end of a run. This variation in allowable changes is accomplished by slowly reducing the value of a variable analogous to temperature in annealing. The main difference between simulated annealing and the algorithms described above is that changes that produce a deterioration in objective value are sometimes allowed. This means that simulated annealing can escape from local optima and find good results in the global sense. The main differences between various simulated annealing algorithms lie in cooling schemes and the probabilities with which changes that worsen the objective value are accepted [27].

Genetic algorithms imitate evolution and natural selection by maintaining a population of solutions and applying genetic operators such as crossover (breeding). The motivation for this approach is the remarkable way in which natural systems have developed to survive in even the harshest environments. The fact that a population of solutions is maintained means that genetic algorithms sample the entire problem space in a structured manner. Genetic algorithms will be discussed in considerably more detail in Section 2.2.

The use of a population to efficiently explore the problem space has led to the development of a number of other global optimisation methods based on a population of individuals. Particle swarm optimisation [28] attempts to imitate the motion of groups of animals such as a swarm of insects, a flock of birds, or a school of fish. The basic principle is that, while each individual is locally searching for a good solution, the interaction between individuals leads to a global search effect. The major advantage of this approach is that each individual can be very simple without compromising the optimisation performance of the swarm as a whole.

2.1.5 Combinatorial Optimisation

Combinatorial optimisation consists of selecting the best combination or permutation of elements. A good example of a combinatorial optimisation problem is the classic travelling salesman problem where the total distance travelled to visit every node in a given area once and only once must be minimised. This section will consider some of the main combinatorial optimisation algorithms.

Simulated annealing has already been considered in connection with global optimisation. Simulated annealing can be applied to combinatorial optimisation problems because the principle of reducing the size of allowable changes during a run is also applicable to combinatorial optimisation.

Genetic algorithms have also been mentioned above in terms of their global optimisation properties. Genetic algorithms can also be applied to combinatorial optimisation problems by modifying the representation and genetic operators used.

Tabu search is a combinatorial optimisation technique that works by disallowing some modifications to the current solution (making those modifications taboo) [29]. The main motivation for this approach is to avoid revisiting areas of the problem space that have already been tested. A simple example is to ensure that the last modification is not undone, thereby returning to the previous solution. The recent history of the algorithm is used in conjunction with a number of heuristic techniques to establish which modifications will be disallowed.

2.1.6 Hybrid Optimisation Algorithms

The optimisation methods described above apply a number of different techniques to the problem of finding the best solution to a problem. Each of these algorithms has advantages and disadvantages. Recently, hybrid algorithms that combine more than one of the algorithms considered above have been developed in an attempt to accentuate the advantages and reduce the disadvantages of each algorithm. The basis for these developments is the "No Free Lunch" Theorem.

Recently a result known as the "No Free Lunch" Theorem was published by Wolpert and

Macready [30]. The basic premise is that any algorithm that performs very well on a particular type problem pays for this with poor performance on another type of problem. A simple example of this principle is that global optimisation algorithms perform very well on global optimisation problems, but very poorly on local optimisation problems. This suggests that hybrid algorithms should perform better than single algorithms when a wide variety of problem types is considered.

Groenwold and Hindley [31] used this principle in the development of a global optimisation algorithm. Their approach applies a number of optimisation algorithms to a problem, and then uses a Bayesian criterion to combine those results and determine when to terminate the search. The results were superior to those obtained when the sub-algorithms were used independently. The major advantage of this approach is that the algorithms can be run independently in parallel, allowing maximum advantage to be taken of clusters of computers.

Another approach to the development of hybrid optimisation algorithms is to integrate the algorithms so that they no longer function independently. Duch and Korczak [32] list a number of possibilities like the integration of simulated annealing and a genetic algorithm. The benefit of this approach is seen by the fact that Michalewicz [33] found that a genetic mutation operator based on simulated annealing (non-uniform mutation) significantly improved the performance of a genetic algorithm. Duch and Korczak [32] suggest that the integration of local and global optimisation algorithms should produce good results.

Genetic algorithms have been combined with local optimisers by Renders and Flasse [34], and Salomon [35]. Renders and Flasse [34] imitate learning in natural systems by creating a new generation using a genetic algorithm and then running a local optimiser to simulate learning. The results of the hybrid algorithm are significantly more reliable than the local optimiser and more accurate than the genetic algorithm, while requiring fewer function evaluations than the genetic algorithm. Salomon [35] compares genetic algorithms and gradient based local optimisers and gives some similarities and differences. This discussion forms the basis for the development of a new algorithm which is a combination of the two approaches. The performance of the hybrid algorithm is better than more established algorithms on some problems.

2.1.7 Constraints

All the algorithms considered so far do not consider constraints and are known as unconstrained optimisation algorithms. Most practical problems have constraints imposed by physical laws such as the speed of light, and other limitations such as size and cost. The process of finding the values of inputs to a system that satisfy all constraints and give the optimum objective function value is known as constrained optimisation. This section will consider some methods of accounting for constraints. The majority of the information contained in this section was obtained from Nocedal and Wright [15].

A simple method of dealing with constraints is to add some penalty to the objective function value whenever a constraint is violated and then to use an unconstrained optimisation algorithm. This principle can be represented as

$$f_c(\mathbf{x}) = f(\mathbf{x}) + \frac{1}{\mu}\sum_i g_i^m(\mathbf{x}) + \frac{1}{\mu}\sum_j h_j^n(\mathbf{x}) \tag{2.19}$$

where $f_c(\mathbf{x})$ is the new objective function, $f(\mathbf{x})$ is the objective function, μ is the penalty parameter which is greater than zero, $g_i(\mathbf{x})$ are the inequality constraints, $h_j(\mathbf{x})$ are the equality constraints, and m and n are values greater than or equal to one, and i and j are the indices of constraints that are violated. The constraint functions are less than zero when a constraint is satisfied and are greater than zero when a constraint is violated, so they are only included in (2.19) when a constraint is violated. As n is increased or μ is decreased the penalty for violating a constraint increases. The main difficulty with this approach is that the Hessian may become ill-conditioned meaning that most optimisation methods will perform poorly. A further problem is that the penalty function can shift the positions of optima and this can cause solutions that violate the constraints by a small amount to be produced. Penalty functions have however been successfully applied by

Snyman [24] for an algorithm that does not use the Hessian. Nocedal and Wright [15] review the augmented Lagrangian method which reduces the problems with ill-conditioning, but the theory required for this method is beyond the scope of this work.

Another method of dealing with constraints is to transform the problem so that the new independent variables can have any value. A possibility reviewed by Bandler [12] is

$$x_i = x_{li} + \frac{1}{\pi}\left(x_{ui} - x_{li}\right) \operatorname{arccot}\left(x_i'\right) \tag{2.20}$$

where x_i is an independent variable of the original problem, x_{ui} and x_{li} are respectively the upper and lower bounds of the independent variable, and x_i' is the new independent variable. An unconstrained optimisation algorithm can now be applied to the new problem with $\mathbf{x}'$ as the independent variable. Nocedal [15] considers similar algorithms known as barrier function methods. The main problem with barrier function methods is that ill-conditioning of the Hessian can occur.

Other approaches using approximations of both the problem and the constraints in the neighbourhood of the current point have been developed. Nocedal and Wright [15] review the quadratic programming and sequential quadratic programming approaches. Snyman [36, 37] has recently proposed a new algorithm that uses a very simple approximation to the problems yet achieves excellent results. The main difficulty with these approaches is that a complicated sub-problem must be solved at each iteration, limiting the value of these algorithms where the evaluation of the objective function is fast. The main advantage of these approaches is that they typically require fewer objective function evaluations than the algorithms considered above making them very well suited to the case where the evaluation of the objective function is costly.

2.2 Genetic Algorithms

Genetic algorithms are numerical techniques that attempt to imitate evolution and natural selection. The motivation for this approach is the remarkable way in which natural systems

Snyman [24] for an algorithm that does not use the Hessian. Nocedal and Wright [15] review the augmented Lagrangian method which reduces the problems with ill-conditioning, but the theory required for this method is beyond the scope of this work.

Another method of dealing with constraints is to transform the problem so that the new independent variables can have any value. A possibility reviewed by Bandler [12] is

$$x_i = x_{li} + \frac{1}{\pi}\left(x_{ui} - x_{li}\right) \operatorname{arccot}\left(x'_i\right) \tag{2.20}$$

where x_i is an independent variable of the original problem, x_{ui} and x_{li} are respectively the upper and lower bounds of the independent variable, and x'_i is the new independent variable. An unconstrained optimisation algorithm can now be applied to the new problem with $\mathbf{x}'$ as the independent variable. Nocedal [15] considers similar algorithms known as barrier function methods. The main problem with barrier function methods is that ill-conditioning of the Hessian can occur.

Other approaches using approximations of both the problem and the constraints in the neighbourhood of the current point have been developed. Nocedal and Wright [15] review the quadratic programming and sequential quadratic programming approaches. Snyman [36, 37] has recently proposed a new algorithm that uses a very simple approximation to the problems yet achieves excellent results. The main difficulty with these approaches is that a complicated sub-problem must be solved at each iteration, limiting the value of these algorithms where the evaluation of the objective function is fast. The main advantage of these approaches is that they typically require fewer objective function evaluations than the algorithms considered above making them very well suited to the case where the evaluation of the objective function is costly.

2.2 Genetic Algorithms

Genetic algorithms are numerical techniques that attempt to imitate evolution and natural selection. The motivation for this approach is the remarkable way in which natural systems adapt to their environments. Genetic algorithms have been applied to a very wide range of problems including optimisation [17,33], computer programming [38], circuit design [38], the traveling salesman problem [17], and training of neural networks [39]. This document will focus on optimisation, but the conclusions are relevant to all applications of genetic algorithms.

There are some semantics surrounding the use of the term "genetic algorithm." The purists insist that only the binary genetic algorithms proposed by Holland [40] may be called genetic algorithms, with all other evolutionary techniques having different names. Michalewicz [33] uses the term "evolutionary program" to refer to any technique based on evolution and natural selection, but this can lead to confusion with "evolutionary programming," an algorithm for evolving finite state machines. For the purposes of this document the term "genetic algorithm" will be used to denote any technique based on evolution and natural selection, and the term "binary genetic algorithm" will be used to denote the form proposed by Holland [40].

Section 2.2.1 will consider genetic algorithms in more detail. Some well known selection schemes, data representations, and genetic operators will be presented to highlight the most important points. The Schema Theorem is discussed in Section 2.2.5 to show how genetic algorithms work. This result leads to some important observations concerning the implementation of genetic algorithms. Section 2.2.6 gives a brief introduction to messy genetic algorithms.

This is a very short introduction to genetic algorithms so only the most basic points are covered. The interested reader is referred to Whitley [41] for a genetic algorithms tutorial which briefly covers some advanced topics, Goldberg [17] for a detailed consideration of binary genetic algorithms, Michalewicz [33] for motivations for representations other than binary, and Bäck *et al.* [42] for a comparatively recent review of the field with over 200 references.

```
initialise population;
calculate fitnesses;
do
  {
    do
      {
        select two individuals;
        apply crossover with probability pc;
        apply mutation with probability pm;
      } until new generation is full;
    calculate fitnesses;
  } until termination criterion in met;
```

Figure 2.5: Pseudo-code implementation of a genetic algorithm.

2.2.1 Background

This section will review the most important characteristics of a genetic algorithm and two of the best known genetic algorithms will be presented as typical examples. The first is the binary genetic algorithm originally proposed by Holland [40] and covered in great detail by Goldberg [17], and the second is the real number genetic algorithm developed by Michalewicz [33].

As mentioned before, a genetic algorithm works by imitating evolution and natural selection as found in nature. The first step is to generate an initial population of individuals. Next a number of individuals are selected, favouring those with higher fitness, but not completely rejecting those with low fitness. These individuals are then allowed to breed, and some survive to the next generation. Mutation is also applied with a low probability and the whole process is then repeated until some termination criteria is met. The basic algorithm is shown in the pseudo-code implementation given in Figure 2.5.

From this discussion it is clear that five criteria have to be met for a genetic algorithm to be applied to a given problem. A representation of the data has to be established, an initial population of individuals must be created, a selection scheme must be chosen, genetic operators (crossover and mutation) need to be implemented, and a termination criteria has to be established.

The initial population is normally generated randomly, but this is not a requirement and individuals that are known to be good can be used to seed the initial population. The most commonly used termination criterion is to stop the algorithm after a fixed number of generations, but other possibilities do exist. One of these is to stop the algorithm when there has been no improvement in the best fitness for a number of generations. The representation of individuals, selection schemes, and genetic operators are considered in the following sections.

Section 2.2.2 reviews four of the most important selection schemes. The representation of the individuals processed by a genetic algorithm is considered in Section 2.2.3. Genetic operators are considered in Section 2.2.4.

2.2.2 Selection

Once a fitness measure for the problem being considered has been established, a way of selecting individuals must be implemented. This section will start with a description of the most important properties of a selection scheme, move on to address elitism, and will then consider four of the most common selection schemes in Sections 2.2.2.1 to 2.2.2.2. The information in this section is mainly obtained from Bäck [43], and Goldberg and Deb [44]

The selection scheme is what causes a genetic algorithm to converge, and thus has a dramatic effect on the performance of the algorithm. The genetic operators considered in Section 2.2.4 only create new individuals and have a comparatively small effect on the convergence of the algorithm. A selection scheme should favour the selection of good individuals to ensure that the algorithm converges to a good solution. However, poor individuals should still have a small chance of being selected to maintain diversity in the population and prevent premature convergence.

One of the problems with all the selection schemes described in this section is that they select individuals in a structured, but random manner. This means that there is usually no guarantee that the best individual from the current generation will survive to the next generation, so convergence is seldom monotonic. The simplest way to overcome this problem is to copy the best individual to the next generation, a process known as elitism [17]. Elitism is not required by $(\mu + \lambda)$ selection because the best individuals are guaranteed to survive to the next generation.

2.2.2.1 Description of Selection Schemes

Proportional selection is also known as Roulette wheel selection. Individuals are randomly selected with a probability equal to the ratio of the fitness of an individual to the sum of the entire population's fitnesses. This can be written as

$$p(\mathbf{x_i}) = \frac{f(\mathbf{x_i})}{\sum_{j=1}^{n} f(\mathbf{x_j})} \tag{2.21}$$

where $f(\mathbf{x_i})$ is the fitness of individual $\mathbf{x_i}$ and $p(\mathbf{x_i})$ is the probability of selecting individual $\mathbf{x_i}$. Proportional selection can obviously only be directly applied to problems where the fitness must be maximised and all possible fitnesses are strictly positive. However it is often possible to convert the fitness to this form and apply proportional selection to the modified fitness.

A subtle problem with proportional selection is that it becomes little more than random selection when a large number of individuals have similar fitnesses. This can happen near the end of a run when most of the population has converged to essentially the same fitness value. Another problem is that an individual that is much better than all others in a given generation will tend to get selected a large number of times, causing the population to converge to that solution. This is a problem if an individual that is much better than all others, but which is not close to the best solution, is generated near the beginning of a run. Both of these problems can be overcome by scaling the fitness to be between some specified upper and lower bounds [17,42]. Linear scaling is the most common form of fitness scaling, but nonlinear scaling can also be used. Nonlinear fitness scaling can be used to adjust the bias towards individuals with higher fitness [42]. The selection techniques listed below avoid these difficulties.

Selection probability is based on an individual's rank within a population in rank-based selection. Individuals that have a higher rank (better individuals) have a greater chance of being selected than individuals with a lower rank. The main advantages of rank-based selection are that it avoids problems with fitness scaling, and it can be used in cases where there is no absolute measure of fitness and individuals can only be compared. This is the case in game playing problems where the outcome of a match can only be a victory, loss, or draw. Linear ranking is the most common form of rank-based selection, but nonlinear ranking is also possible. As with fitness scaling, the bias towards better individuals can be adjusted in rank-based selection.

Tournament selection works by randomly choosing a number of individuals from the population to participate in a tournament. The individuals with the best fitnesses in the tournament are then chosen as the winners of the tournament, and are the results of the selection. This is usually implemented by randomly choosing two or three individuals at a time and then either selecting the individual with the best fitness or performing proportional selection on the individuals in the tournament. As with rank-based selection, tournament selection avoids fitness scaling problems and the bias towards better individuals can be adjusted. Tournament selection has the additional advantage that it is simple to implement.

The last selection methods considered here work by creating a population larger than that required and eliminating the worst individuals. In (μ, λ) selection the next generation is formed by selecting the best μ individuals from λ offspring, where μ is smaller than λ. The process for $(\mu+\lambda)$ selection is similar except that the next generation is formed from the μ best individuals from the population consisting of both offspring and parents (μ individuals from $\mu+\lambda$ individuals). The main advantage of $(\mu+\lambda)$ selection over (μ, λ) selection is that $(\mu+\lambda)$ selection ensures that the best value improves monotonically. These two selection schemes have similar advantages to tournament selection, but they can have a stronger bias towards better individuals, and finer tuning of the bias can be achieved. These two schemes are common in algorithms based on Evolution Strategies [43].

2.2.2.2 Comparison

Detailed comparisons of these and other selection schemes can be found in the literature [17, 43, 44]. Goldberg and Deb [44] compare these selection schemes in terms of growth ratio, takeover time and time complexity, while Bäck [43] only considers takeover time.

Growth ratio is the ratio of the number of copies of an individual in the next generation to the number of copies of that individual in the current generation. The genetic operators (crossover and mutation) are ignored during a growth ratio calculation so that only the effect of the selection scheme used is seen. Proportional selection was found to have a growth ratio that is heavily dependent on the fitness function, and is high early on and low near the end of a run. Rank-based and tournament selection were found to have similar growth ratios that are much better than proportional selection. Genitor selection (a type of $(\mu+\lambda)$ selection) was found to have a growth ratio that is so high it actually causes problems.

Takeover time is defined as the time taken for the best individual to be copied to every position in a population but one. Again, the effect of crossover and mutation are ignored. Goldberg and Deb [44] found that the takeover times of all the selection schemes were of the same order of magnitude. The more detailed study conducted by Bäck [43] also considered the effect of changing the bias towards better individuals. Proportional selection was found to have a very high takeover time. Linear rank-based selection was shown to have a much better takeover time than proportional selection, and tournament selection was better than linear rank-based selection. The shortest takeover times are obtained by (μ, λ) and $(\mu + \lambda)$ selection. The takeover times for rank-based, tournament, and (μ, λ) and $(\mu + \lambda)$ selection can be varied, with the largest range of variation being achieved by (μ, λ) and $(\mu + \lambda)$ selection followed by tournament selection.

The last parameter considered by Goldberg and Deb [44] is the time complexity (an indication of how much processing time is required) of each of these selection schemes. The most complex is proportional selection with a time complexity of order n^2, where n is the population size, followed by rank-based and Genitor (a type of $(\mu + \lambda)$) selection with time complexities of order $n \log(n)$. The simplest of the selection schemes considered here is tournament selection with a time complexity of order n. It is difficult to imagine a selection scheme with a time complexity of order less than n because n individuals must be selected.

2.2.3 Representation

The representation chosen for the individuals in a population can have a large effect on the performance of a genetic algorithm. Some general points will be considered here and two examples will be highlighted. Obviously different operators will be required for different representations.

The representation used should follow directly from the problem itself wherever possible. Important points to take note of are that the amount of data should be kept to a minimum, and features that are close together in the problem should be close together in the representation. An example of the first point is that a representation with 10 decimal digits accuracy shouldn't be used if only 3 decimal digits are required. The problems with having unnecessary data are that the algorithm takes longer to converge, and the chances that it will converge to a sub-optimum result are increased. An example of the principle that features that are close together in the problem should be close together in the representation is that a two dimensional matrix would be a better representation for a two dimensional problem than a one dimensional vector. Both these points arise from the Schema Theorem presented in Section 2.2.5 and will be discussed further there.

The first genetic algorithms proposed by Holland used a binary string of the form

$$\mathbf{x} = x_{n-1}x_{n-2}\ldots x_3x_2x_1x_0 \tag{2.22}$$

where x_i are binary digits, and n is the length of the string. This representation can then be decoded to a single variable of n bits, two variables of $n/2$ bits, or m variables of n/m bits. The motivation for a binary representation comes from natural systems where genes are either present or absent. While this representation works well for integer variables, a large number of bits is required to encode real numbers which require very high accuracy or large ranges.

Michalewicz [33] overcomes this problem by using a number of floating point values to encode variables that are real numbers. In other words, each individual uses one floating point value to encode each system variable. In terms of (2.22), this means that each x_i is a real number and the problem has n variables.

2.2.4 Operators

After individuals have been selected, genetic operators are applied to create new individuals. The most important properties of genetic operators are presented below, followed by a consideration of mutation in Section 2.2.4.1 and crossover in Section 2.2.4.2.

Genetic operators are applied to modify current solutions in an attempt to produce improved solutions. As mentioned in Section 2.2.2, genetic operators have a comparatively small effect on the convergence of a genetic algorithm, but good genetic operators can lead to improved results.

Genetic operators are applied with a probability less than 1, so not all individuals are affected, with unaffected individuals being copied to the new generation unaltered. This is necessary because genetic operators tend to have a disruptive effect as shown in Section 2.2.5.2. Allowing individuals to survive to the next generation without any modifications ensures that current solutions to the problem are retained.

Operators depend heavily on the representation used and fall into two general categories, mutation and crossover, with mutation being applied after crossover in most implementations. The binary operators originally proposed by Holland [40], and the floating point operators suggested by Michalewicz [33] will be presented below.

2.2.4.1 Mutation

Mutation operators take one individual as an input and randomly change that individual to ensure that genetic diversity is maintained in the population. This is necessary to ensure that the population does not prematurely converge to a poor solution. Mutation is a background operator that has a very small effect on the operation of most genetic algorithms, so it is applied with a very low probability. This does not mean that mutation in unnecessary, and a genetic algorithm that does not have mutation will not give good results. Binary, uniform, boundary, and non-uniform mutation will be presented as examples of typical mutation operators.

Binary mutation was originally suggested by Holland [40] as part of a binary genetic algorithm. Binary mutation works by randomly inverting a bit in the binary representation of an individual, which means that binary mutation can only be used with a binary representation. Binary mutation is usually applied with a probability such that one in a hundred to one in a thousand bits is affected [17].

Uniform mutation was introduced as part of a real number genetic algorithm by Michalewicz [33]. Uniform mutation modifies an individual by changing one of the variables in an individual to a random value uniformly distributed between the maximum and minimum allowable values of that variable.

Boundary mutation was introduced by Michalewicz [33] as part of a real number genetic algorithm to counteract the bias inherent in arithmetic crossover, and to effectively search the extreme values of each variable. Arithmetic crossover will be covered in detail below, but at this point it is sufficient to state that it tends to favour values near the middle of each variable's allowable range. Operators should not introduce any kind of bias because this will affect the operation of the genetic algorithm by favouring regions that do not necessarily produce good results. Boundary mutation randomly assigns either the maximum or minimum allowable value to one variable in an individual.

Non-uniform mutation was also introduced by Michalewicz [33] for use with a real number representation. This operator is similar to simulated annealing [14, 45] in the sense that the amount by which an individual is changed depends on the age of the population. Large changes are possible during the early part of a run when individuals are still far from good results, but only small changes are allowed near the end of a run when individuals are near good values. This is done to ensure that the search space is adequately covered near the beginning of the algorithm by allowing large changes, but good results are not lost near the end of the algorithm. One of the variables in a representation of the form shown in (2.22) will be changed according to

$$x_i' = \begin{cases} x_i + \Delta(t, u_i - x_i) & s = 0 \\ x_i - \Delta(t, x_i - l_i) & s = 1 \end{cases} \tag{2.23}$$

where x_i and x_i' are the old and new values of the variable respectively, l_i and u_i are the minimum and maximum values of the variable respectively, and s is a random bit that has an equal probability of being 0 or 1. The value of Δ used by Michalewicz [33] is given by

$$\Delta(t, y) = y\left[1 - r^{\left(1-\frac{t}{T}\right)^b}\right] \tag{2.24}$$

where r is a random number uniformly distributed between 0 and 1, t and T are the current and final values of the population age (usually numbers of generations) respectively, and b is a parameter used to determine how much the mutation range changes with population age. Michalewicz [33] uses a value of 5 for b.

2.2.4.2 Crossover

Crossover is analogous to breeding in natural systems. The basic principle of crossover is that genetic information from two or more individuals is combined to form a new individual. Crossover is the most important operator in the vast majority of genetic algorithm implementations. This section will consider binary, simple, and arithmetic crossover.

Binary crossover was originally proposed by Holland [40] and is intended for use with a binary representation. Binary crossover works by selecting two parents and a cross point, and then copying the data from one parent before the cross point and from the other parent after the cross point. An example where two 8 bit parents are crossed after bit 5 to produce two offspring is shown below. The parents are given by

$$\begin{array}{ccc|ccccc} x_7 & x_6 & x_5 & x_4 & x_3 & x_2 & x_1 & x_0 \\ y_7 & y_6 & y_5 & y_4 & y_3 & y_2 & y_1 & y_0 \end{array} \tag{2.25}$$

and the offspring by

$$\begin{array}{ccc|ccccc} x_7 & x_6 & x_5 & y_4 & y_3 & y_2 & y_1 & y_0 \\ y_7 & y_6 & y_5 & x_4 & x_3 & x_2 & x_1 & x_0 \end{array} \tag{2.26}$$

where x_i and y_i are the bits of the parents, and | shows the cross point. In this case two offspring are generated for every crossover, but generating one offspring per crossover is also possible. Binary crossover is normally applied with a probability of around 0.6.

Simple crossover was introduced by Michalewicz [33] and is very similar to binary crossover, except that it is applied to representations with real numbers. The only difference to the case shown in the previous paragraph is that x_i and y_i are real numbers.

Simple crossover is very limited because it does not affect the values of the variables, so Michalewicz [33] also uses arithmetic crossover. This operator copies variables from one parent before the cross point, modifies the variable at the cross point using data from both parents, and then copies the remaining variables from the other parent. An example for the case where two parents with eight variables are crossed at variable 4 is shown below.

The parents are given by

$$\begin{array}{ccc|c|cccc} x_7 & x_6 & x_5 & x_4 & x_3 & x_2 & x_1 & x_0 \\ y_7 & y_6 & y_5 & y_4 & y_3 & y_2 & y_1 & y_0 \end{array} \tag{2.27}$$

and the offspring is given by

$$\begin{array}{ccc|c|cccc} x_7 & x_6 & x_5 & a_4 & y_3 & y_2 & y_1 & y_0 \end{array} \tag{2.28}$$

where x_i and y_i are real numbers, | shows the cross point, and a_i is given by

$$a_i = rx_i + (1 - r)y_i \tag{2.29}$$

where r is a random number between 0 and 1. It is also possible to generate two offspring from two parents as with binary crossover.

2.2.5 Schema Theorem

So far this work has concentrated on what genetic algorithms are and how to implement them, but nothing has been said about how and why genetic algorithms work. This is addressed by the Schema Theorem which explains the operation of genetic algorithms in terms of schemata (singular: schema). The Schema Theorem is covered by most books and tutorials that deal with genetic algorithms including [17, 33, 41]. While the Schema Theorem is only valid for binary genetic algorithms, the results can be applied to any genetic algorithm. This section will give a brief overview of the Schema Theorem to allow the reader to gain an understanding of how and why genetic algorithms work.

2.2.5.1 Schema Definition

The first step towards deriving the Schema Theorem is defining a schema. A schema is a template used to show similarities between different individuals. This is done by specifying the value of some of the bits in an individual, and leaving the values of all the other bits unspecified. An asterisk is typically used as a "don't care" symbol to show which bits' values are unspecified. For example, all individuals which have the first bit of an eight bit string set to 1 are represented by the schema 1*** ****. Another example is that the strings 1010 1110 and 1110 1101 both contain the schema 1*1* ****.

Schemata have two fundamental properties, order and length. The order of a schema, denoted by $o(H)$, is the number of bits whose values are specified (they are not "don't care" values). The schemata 1*** ****, ***0 ****, and **** **1* all have order 1 because only one bit's value is specified. Examples of second order schemata are *1** *0**, *00* ***, and 1*** ***1. High order schemata match fewer strings than low order schemata, so the order of a schema is an indication of how general a schema is. The length of a schema, denoted by $\delta(H)$, is the number of bit positions from the first specified bit to the last specified bit. For example, the schema **1* 0*0* has a length of 4 because the first specified bit is in position 1 and the last specified bit is in position 5, where the bits are numbered from right to left as shown in (2.22).

2.2.5.2 Derivation

The fundamental result of the Schema Theorem is a lower bound on the quantity of a given schema which will be present in a new generation. This value is affected by the number of copies of that schema in the current generation, the fitness of the schema relative to the average schema fitness, and the probability that crossover and mutation will disrupt the schema.

The first important step is thus to define the fitness of a schema, denoted by $f(H)$. A schema's fitness is the average of the fitnesses of all strings that contain that schema. For example, the fitness of the schema 1101 011* is the average of the fitnesses of the strings 1101 0110 and 1101 0111, and the fitness of *101 *001 is the average of the fitnesses of the strings 0101 0001, 1101 0001, 0101 1001, and 1101 1001. The fitness of a schema is implicitly processed by a genetic algorithm and need not be explicitly calculated. Schema fitness is processed by fact that the individuals in a population which contain good schemata should have higher fitnesses than individuals that contain poor schemata. The accuracy of this assumption will improve as the population size increases because the number of copies of each schema will increase.

The number of copies of a given schema that are selected is now derived assuming that proportional selection (see Section 2.2.4.2) is used. Similar growth measure results for other selection schemes have been derived by Goldberg and Deb [44], but are not considered here because the results are similar. The probability that an individual, and thus a particular instance of a schema, will be selected is given by (2.21). The probability that any copy of a schema is selected is this probability multiplied by the number of copies of that schema in the current generation. This process is repeated until one individual has been selected for every space in the new generation. The number of copies of a schema that are selected for the next generation is thus given by

$$m(H,t+1) = m(H,t)n\frac{f(H)}{\sum_{j=1}^{n} f(\mathbf{x_j})} \tag{2.30}$$

where $m(H,t)$ is the number of copies of schema H in generation t, and n is the population size. This can be simplified by noting that the sum of all fitnesses divided by the population size is the average fitness of all individuals in the current population. So (2.30) becomes

$$m(H,t+1) = m(H,t)\frac{f(H)}{\bar{f}} \tag{2.31}$$

where $\bar{f}$ is the current population's average fitness.

The result in (2.31) shows that the number of good schemata (those with a fitness greater than the population average) will increase and the number of bad schemata (those with a fitness lower than the population average) will decrease. While this is desirable, it serves no purpose unless those schemata are processed to produce better results, and this requirement is met by the genetic operators described previously. Some schemata are destroyed and others are created during the application of genetic operators. When binary crossover takes place, schemata are destroyed when the cross point falls between bits whose values are specified as shown below.

$$\begin{array}{rclll|lllll}
\mathrm{x_i} & = & x_7 & x_6 & x_5 & x_4 & x_3 & x_2 & x_1 & x_0 \\
H_1 & = & 1 & * & * & 1 & * & * & * & 0 \\
H_2 & = & * & * & * & 1 & * & * & 0 & *
\end{array} \tag{2.32}$$

In this example the cross point is chosen between bits 4 and 5. This is between two specified bits of schema H_1 so it will be destroyed unless the other individual taking part in the crossover has the same values in those bit positions. The cross point does not lie between any specified bits of schema H_2, so it is not affected by this crossover. Binary mutation will disrupt any schema that has a specified value that is mutated. Obviously new schemata that were not present in the individuals processed by the genetic operators will also be created.

The probability with which schemata are created and destroyed needs to be included in (2.31) to account for the effect of genetic operators. This analysis will only consider the probability that a schema will survive crossover and mutation, ignoring the creation of new schemata. This is a worst case analysis which establishes a lower bound on the probability that a schema will survive to the next generation.

A schema is disrupted by binary crossover if the cross point falls between the first and last specified bits of the schema. The cross point for binary crossover can be between any two bits, which means that an individual of length n has $n-1$ potential cross points. Every potential cross point has the same probability of being used, so the probability of using a specific cross point is $1/(n-1)$. The number of cross points between the first and last specified bits of a schema is equal to the length of the schema $\delta(H)$ so the probability of a schema being disrupted by binary crossover is thus $\delta(H)/(n-1)$. Binary crossover is not applied in every case, so this value must be multiplied by the probability that binary crossover is applied. The probability that a schema will survive crossover is given by

$$p_s = 1 - p_c\frac{\delta(H)}{n-1} \tag{2.33}$$

where p_c is the probability that crossover is applied. Equation (2.31) can now be modified to account for the effect of crossover, giving

$$m(H,t+1) = m(H,t)\frac{f(H)}{\bar{f}}\left[1 - p_c\frac{\delta(H)}{n-1}\right] \;. \tag{2.34}$$

Binary mutation can disrupt a schema by inverting any of the specified bits in a schema. The probability that any particular bit will be mutated is equal to the mutation probability p_m. The number of bits specified by a schema is given by the order of the schema $o(H)$ so the probability that a schema will be disrupted is given by $p_m o(H)$. This can be included in (2.34) to give the number of copies of a particular schema that can be expected in the next generation:

$$m(H,t+1) = m(H,t)\frac{f(H)}{\bar{f}}\left[1 - p_c\frac{\delta(H)}{n-1} - p_m o(H)\right] \tag{2.35}$$

As mentioned above, (2.35) ignores the fact that schemata are also created by crossover and mutation. This means that the value given above is actually a lower bound on the quantity of a schema that can be expected in the next generation, and thus represents the worst case. This can be shown explicitly by modifying (2.35) to use an inequality, giving

$$m(H,t+1) \geq m(H,t)\frac{f(H)}{\bar{f}}\left[1 - p_c\frac{\delta(H)}{n-1} - p_m o(H)\right] \;. \tag{2.36}$$

2.2.5.3 Implications of the Schema Theorem

This section will consider some of the implications of the Schema Theorem and how they can be used to design better genetic algorithms.

The first important result is given in (2.31) and shows the expected number of copies of a schema in a new generation. The factor $f(H)/\bar{f}$ means that the number of copies of a given schema grows with the ratio of the fitness of the schema to the average population fitness. If the ratio of the schema fitness to the average population fitness is assumed to be a constant, (2.31) becomes a geometric progression. In other words, a schema with a fitness higher than the average fitness will get an exponentially increasing number of trials assigned to it over time, assuming it is not disrupted by the genetic operators. A schema with fitness lower than the average fitness will get an exponentially decreasing number of trials assigned to it. This means that a population should very rapidly converge to a good result. Section 2.2.2.2 notes that proportional selection has the lowest growth ratio of the selection schemes considered, so the other selection schemes will do even better.

The final result given in (2.36) adds the effect of crossover and mutation to the result in (2.31). This equation shows that short schemata have a greater chance of surviving crossover than long schemata, and low order schemata have a higher probability of surviving mutation than high order schemata.

These conclusions lead to a result known as the building block hypothesis. A building block is simply a short, low order schema. The main result of the Schema Theorem is that good building blocks will be combined in such a way as to improve the population's fitness. This means that the representation chosen should be such that important parts of the problem are close together to minimise the possibility of being disrupted by crossover, and the amount of data required should be kept to a minimum to reduce the chances of disruption by mutation. Both these points were mentioned in Section 2.2.3.

The last issue that needs to be considered is the number of schemata that are processed by the algorithm per generation that contribute useful information about the problem. Not every schema that is present in the population will be usefully processed because long, high order schemata have a high probability of being disrupted by genetic operators. Goldberg [17] shows that the number of usefully processed schemata is of order n^3. So despite only explicitly processing n individuals per generation, a genetic algorithm implicitly processes on the order of n^3 schemata. This important result is known as implicit parallelism.

While the Schema Theorem is a very useful result which gives an insight into the operation of genetic algorithms, it is only valid for binary genetic algorithms and does not constitute a rigorous mathematical proof. The search for a generally applicable, mathematically rigorous proof of the convergence of genetic algorithms is ongoing. Leung *et al.* [46] give a brief review of the most important results that have been achieved, and suggest a new model for analysing genetic algorithms.

2.2.6 Messy Genetic Algorithms

Genetic algorithms are a very powerful technique for solving difficult problems, but they are not perfect. One of the most difficult aspects of implementing a useful genetic algorithm is determining the representation. This is complicated by the fact that it is usually not possible to know in advance which ordering of variables will produce good results. Messy genetic algorithms [47,48] overcome this problem by modifying genetic algorithms to reduce ordering problems.

Section 2.2.6.1 will consider the motivation for messy genetic algorithms in more detail. The unique initialisation of messy genetic algorithms is presented in Section 2.2.6.2. Section 2.2.6.3 considers the representation used by messy genetic algorithms and highlights the difficulties this representation causes for fitness calculation. Section 2.2.6.4 considers the modifications that have to be made to conventional selection algorithms before they can be used for messy genetic algorithms. Section 2.2.6.5 presents the operators used in messy genetic algorithms. Lastly, Section 2.2.6.6 briefly considers the Schema Theorem as applied to messy genetic algorithms.

2.2.6.1 Motivation

The performance of a genetic algorithm can depend very strongly on the representation used. This is particularly evident in deceptive problems – problems that are known to be difficult for genetic algorithms. This section will give a brief overview of this difficulty.

The Schema Theorem states that short, low order, high fitness building blocks are combined to improve the population fitness. This means that features of a function must be tightly linked (close together) in the representation so that the length of the important schemata are as short as possible. Unfortunately, it is not always possible to know in advance which ordering of the data will produce good results. This effect is accentuated when deceptive problems are considered. A number of techniques have been suggested for overcoming this problem, but none is satisfactory.

The first possibility is just to use a random ordering for every problem, but this is not a good solution because the probability of obtaining a good ordering is low. The other possibility is to introduce an operator which changes the ordering used in the representation during a run of the algorithm. This can be considered as a mutation operator for representation ordering. Goldberg *et al.* [47] give a number of difficulties with this approach. The most important objection to a reordering operator is that the genetic algorithm will then be trying to optimise both the ordering and fitness at the same time, producing a much more difficult problem than just improving fitness.

The problems highlighted above suggest that a new approach to genetic algorithm representation ordering is required. The messy genetic algorithm proposed by Goldberg *et al.* [47, 48] overcomes these ordering problems.

The inspiration for messy genetic algorithms, as with conventional genetic algorithms, comes from nature. Conventional genetic algorithms use the assumption that evolution takes place with a fixed number of genes which are either absent or present, but this is only true over comparatively short periods in evolution. Over longer periods of time this approximation is seen to be incorrect. In nature, individuals frequently have genes which are present more than once (overspecification), genes which are required but absent (underspecification), and chromosones which can vary in length. Messy genetic algorithms are unique because they allow varying length representations, underspecification, and overspecification.

2.2.6.2 Initialisation

The first major difference between messy genetic algorithms and conventional genetic algorithms is in the initialisation of the population.

The messy genetic algorithm starts with a large number of short individuals. These individuals are initialised to contain all possible building blocks of a specified, short length. The first phase of a messy genetic algorithm, known as the primordial phase, proceeds by only applying selection and gradually reducing the population size. This is necessary because a large number of individuals are produced during initialisation and it would be impractical to consider all these individuals in the complete algorithm. An additional advantage is that the use of selection means that mostly good individuals survive to the next phase, known as the juxtapositional phase, giving a very good starting point. In the juxtapositional phase both selection and genetic operators are applied.

2.2.6.3 Representation

The next important characteristic of any genetic algorithm is the representation used, and the unique aspects of the messy genetic algorithm representation are presented below.

In the case of messy genetic algorithms, the representation is of the form

$$\mathbf{x} = (n-1, x_{n-1})(n-2, x_{n-2})\ldots(3, x_3)(2, x_2)(1, x_1)(0, x_0) \tag{2.37}$$

where the first number in each bracket is an index (i), and x_i is the value of bit i. The most important difference between this case and the binary genetic algorithm case given in (2.22) is the fact that each bit carries a label in this case. This is necessary because a messy genetic algorithm allows underspecification and overspecification in its representation. Another important difference is that the order of the bits in an individual is not important in a messy genetic algorithm. A typical individual is

$$(2, 1) \quad (4, 1) \quad (5, 0) \quad (4, 0) \quad (7, 1) \quad (1, 1) \quad . \tag{2.38}$$

Note that bits are not in order, bits 6, 3 and 0 are not present, and bit 4 is present more than once. The major challenge arising from this representation is finding a sensible way to calculate fitness.

There are two cases that cause difficulties with fitness calculation in a messy genetic algorithm: overspecification and underspecification. Overspecification is the simpler of the two cases to deal with, and Goldberg *et al.* [47, 48] simply use the first instance of any bit, so bit 4 would be 1 in the example in (2.38). The problem of underspecification is significantly more complex. Goldberg *et al.* [47] consider a number of possibilities and show that taking the missing bits from a locally optimal solution produces excellent results.

2.2.6.4 Selection

Once the fitness is calculated, the next step is to select individuals to create the next generation. Messy genetic algorithms can use the same selection algorithms as conventional genetic algorithms (see Section 2.2.2), but the fact that not all variables will be present in every individual does result in a few difficulties.

The main problem is to ensure that comparisons between individuals are valid. Comparing two individuals that do not have any variables in common is obviously not a good approach, but requiring too many variables to be common will limit the number of possible comparisons. Goldberg *et al.* [48] overcome this difficulty by only comparing individuals where the number of variables common to both individuals is greater than the number of variables that are expected to be common to both individuals due to the randomness inherent in the system. Goldberg *et al.* [48] have conducted a number of tests to prove the viability of this approach.

2.2.6.5 Genetic Operators

The last important characteristic of any genetic algorithm is the implementation of genetic operators. The messy genetic operators described by Goldberg *et al.* [47] are presented below.

The mutation operator is essentially the same as for the binary genetic algorithm with bits in the representation being randomly inverted with some low probability.

Crossover is also very similar to the binary genetic algorithm case except that the variable length of individuals must be accounted for. This is done by choosing the cross point for each parent independently. The offspring is then generated by copying data from one parent before its cross point and then from the other parent after its cross point. For example, if the parents are given by

$$\begin{array}{rrr|rr} (1,1) & (5,0) & (3,1) & (6,1) & (1,0) \\ & (5,0) & (6,1) & (3,0) & \end{array} \tag{2.39}$$

the offspring will be

$$\begin{array}{rrr|r} (1,1) & (5,0) & (3,1) & (3,0) \end{array} \tag{2.40}$$

where | shows the cross points. Note that the length of the offspring is different to that of its parents, and it is possible to produce more than one offspring.

2.2.6.6 Schema Theorem

Goldberg *et al.* [47] have extended the Schema Theorem to consider messy genetic algorithms. The most important differences from the form derived in Section 2.2.5 are that individual length is not a constant, and that variables can be masked if they are present more than once in an individual. The extension is significant because it shows that, despite the differences between messy genetic algorithms and conventional genetic algorithms their operation is similar. This is a very important result because the Schema Theorem was used to justify the development of messy genetic algorithms in the first place.

2.3 Impedance Matching

This section will present a brief review of some of the more important impedance matching network design techniques. The strengths and weaknesses of each type of approach will be considered to justify the development of a new algorithm.

Graphical impedance matching is considered in Section 2.3.1, and analytic approaches are reviewed in Section 2.3.2. The real-frequency techniques are presented in Section 2.3.3 and represent the best techniques available today. The application of genetic algorithms to impedance matching network design is considered in Section 2.3.4.

2.3.1 Graphical Techniques

A number of graphical techniques for impedance matching network design are available in the literature. This section will consider some of these techniques.

The vast majority of graphical techniques use the Smith Chart which is described in most electromagnetics and high frequency design textbooks such as Gonzalez [3], Pozar [11], and Marshall and Skitek [49]. There are however some graphical techniques that do not use the Smith Chart such as the technique proposed by Hamid and Yunik [50].

The simplest of these graphical techniques are the single and double stub matching techniques considered by a large number of textbooks including Gonzalez [3], Marshall and Skitek [49], and Pozar [11]. These techniques are usually only considered for single matching problems, but they can be generalised to the double matching case. The most significant limitation of these techniques is that the impedances of the transmission lines used are fixed, removing one degree of freedom from the design procedure for each section of transmission line. This can however be an advantage where transmission lines are only available with a few values of characteristic impedance.

Another well-known graphical technique is the design of lumped element matching networks using an immitance Smith Chart (a Smith Chart with both impedance and admittance circles). This technique is considered by many authors including Gonzalez [3]. The process is essentially to move along a constant resistance circle when a reactive element is added in series, and along a constant conductance circle when a reactive element is added in parallel. This technique can be used for double matching problems, but is usually only explicitly presented for the single matching case. The bandwidth of these matching networks can be estimated from the transformation-Q values obtained during synthesis.

The paper by French and Fooks [51] deals with the design of a matching network with two transmission lines in series. The problem considered is that of single matching. The characteristic impedances of the transmission lines is fixed in advance, removing two degrees of freedom from the design algorithm. As before this does have the advantage of allowing transmission lines with readily available characteristic impedances to be used. The major advantage of this technique is that it is possible to plot the impedances that can be matched using two series transmission lines of given characteristic impedances on the Smith Chart, allowing insight into the problem to be gained.

The problem of matching two impedances using a series transmission line has been considered by a number of authors. The techniques are all based on the fact that the locus of the input impedances of a series transmission line as the length varies is a circle and the characteristic impedance is given by the geometric mean of the intercepts with the real axis. French and Fooks [52] describe a single matching technique that uses a Smith Chart normalised to the desired impedance to find the characteristic impedance of the series transmission line. Somlo [53] describes a more general double matching technique where a circle with its centre on the real axis of the Smith Chart is constructed through the two impedances to be matched to find the characteristic impedance of the series transmission line. Both these techniques then re-normalise the Smith Chart to the transmission line's characteristic impedance to find the line length. Arnold [54] proposes a technique that can find the line length without re-normalisation, but the geometric constructions are more complex and often have to be repeated to obtain a good result. Day [55] combines Somlo's method of determining the characteristic impedance with Arnold's method of determining the line length to obtain a double matching technique that does not require re-normalisation. The major advantage of this technique is the impedances that can be matched to a given impedance using a series transmission line can be plotted on the Smith Chart allowing a greater understanding of the problem to be gained.

The major advantages of graphical techniques are that they require very few calculations and they provide insight into the matching procedure that cannot be obtained with other techniques. The major disadvantages are that these techniques only consider a single frequency explicitly, the geometric constructions on the graph can be complex, and inaccuracies in reading values from the graph can limit accuracy.

2.3.2 Analytic Methods

This section will consider impedance matching methods that use analytic theory. These methods allowed the formulation of gain-bandwidth theory that sets theoretical limits on the quality of match that can be obtained for a given load. Analytic impedance matching methods were essential before the advent of low cost, high performance computers. This section will give a brief history of analytic impedance matching.

The first significant work on impedance matching was published by Bode [56]. Bode analysed RC and RL loads and proved the existence of gain-bandwidth limits. Fano [57, 58] extended Bode's work on gain-bandwidth theory and matching network design to consider more complex cases including RLC loads. Youla [59] simplified Fano's work by using the principle of complex normalisation. Later a number of approaches to consider the case where both the source and load are networks were developed, including those proposed by Carlin and Yarman [8], and Chen and Satyanarayana [60].

These results were still extremely complex and difficult to use, so a number of papers containing simplified results were published. The approach used by most early papers, including Bode's [56] classic work, was to include graphs that allowed the design parameters to be read from the graph. Another approach to the problem was to publish design tables. This approach was used by Matthaei [61] and Cristal [62] to publish optimum networks for matching two resistances using Chebyshev and Butterworth responses respectively. The last approach involved deriving formulae for the values of the circuit elements required to design an impedance matching network. As an example, Chen [63], and Chen and Kourounis [64] have published formulae to match a load consisting of a RLC network to a resistor.

As wideband circuits began to gain popularity, the need to level the gain of a circuit over a frequency band became apparent. The is necessary because an active device's gain decreases with increasing frequency at high frequencies. Pitzalis and Gilson [65] used a computer to generate tables of networks with 4, 5, and 6 dB per decade frequency slopes to compensate for active device gain rolloff. Ku and Peterson [66] extended this work by publishing an algorithm for obtaining matching networks with specified frequency slopes without the use of a computer.

Transmission lines are considerably more complex to analyse than lumped elements, so much less work has been done on the analysis of impedance matching networks with distributed elements, and results tend to be incremental improvements of previous work. Most of the work has concentrated on the use of a number of series transmission lines of the same length (commensurate lines). An equiripple frequency response gives much wider bandwidth than the maximally flat case, so the equiripple frequency response has been extensively considered in the literature. The first results were published by Collin [67] and Cohn [68], with a more rigorous formulation being developed by Riblet [69], and later, extensions by Young [70] were made available. Young [71] used these results to publish tables of quarter-wave series matching networks. Gledhill and Issa [72], and Chang and Mott [73] derived simpler equations for determining the characteristic impedance of each transmission line. All these techniques have the limitation of only being applicable to the case where two resistances that do not vary with frequency are matched.

Another approach to designing distributed impedance matching networks is to transform a lumped circuit to a distributed approximation. This has the advantage that all the theory relevant to the design of lumped matching networks can be applied to distributed networks. The major result in obtaining distributed equivalents to lumped networks was published by Richards [74] and allows lumped elements to be replaced by open or short circuited transmission line stubs. Richards' transform is usually used in conjunction with Kuroda's identities and Norton's identities [2] to obtain a practically realisable circuit. Carlin and Kohler [75] have used Richards' transform to derive transfer functions that can be realised by transmission lines, and Kuroda's identities to eliminate series stubs. Richards' transformation and Kuroda's identities were also used by van der Walt [76] to obtain very small distributed matching networks. Other possibilities for approximation of lumped elements exist such as using a transmission line with a high impedance to approximate a series inductor and a transmission line with a low impedance to approximate a parallel capacitor. This approach was used by Matthaei [77] to produce tables of matching networks using transmission lines shorter than a quarter wavelength.

The approach described by Matthaei [77] reduces the length of each transmission line, but increases the range of characteristic impedances required. This problem led Levy [78,79] to the develop a class of mixed lumped and distributed matching networks. Levy's approach differs from Matthaei's in that the parallel capacitors are not replaced by series transmission lines. The capacitances can be adjusted to compensate for discontinuities before being implemented. Possible implementations for the capacitances include capacitors, irises, and parallel stubs.

Recently, Drozd and Joines [80] have published a paper which considers parallel stubs as resonators. This technique has the advantage of allowing the design of transmission line impedance matching networks that use stubs without requiring transformations from lumped circuits.

The main advantage of analytic impedance matching techniques is that fundamental limits have been derived in the form of gain-bandwidth theory. The other advantage of analytic methods is that a computer is not required to compute the results – an essential requirement of any design technique before the advent of low cost, high performance computers. The main disadvantage of analytic impedance matching techniques is that they only consider cases that can be analysed analytically. This means that most of these methods only consider very simple cases where the load and source are simple networks, the frequency response is either flat or rolls off at some constant rate, the resulting network is either low-pass or high-pass even if a band-pass structure would be more suitable, and discontinuities are often not adequately compensated for.

The development of the real-frequency techniques has meant that analytic techniques are hardly ever used today because the real-frequency techniques use the measured data of the system rather than an approximation, and the results are significantly better than analytic methods [5, 8].

2.3.3 Real-Frequency Techniques

Real-frequency techniques are those impedance matching techniques that use the actual data of a system to design a matching network. This is in contrast to analytic techniques which rely on some model of the system rather than the system data itself, and optimisation techniques which cannot be considered as design techniques because they require a good starting point. This section will give a brief overview of some of these techniques.

The first real-frequency technique was proposed by Carlin [4]. This is a single-matching technique which can design for specified transducer gains. The basic principle is that the transducer gain of the matching network is derived in terms of the load and the input impedances of the network. Only the input resistance or conductance as a function of frequency has to be considered because the imaginary part of the input impedance or admittance can be found from the real part using a Hilbert transform. A piecewise linear approximation of the input resistance or conductance is used to simplify the implementation, but other approximations such as the Wiener-Lee mapping can also be used [10]. The optimisation of the resistance or conductance function is then achieved using a standard optimisation technique. There are many possibilities for initialising the algorithm. Carlin [4] suggests that the gain could be levelled at its DC value over the frequency band and Abrie [2] suggests using the load resistance value, with good results being obtained in both cases. Carlin and Komiak [81] show how this technique can be expanded to consider other factors such as amplifier stability by including constraints during optimisation. This technique is compared to analytic impedance matching algorithms by Carlin and Amstutz [5] and even this simple technique is seen to be significantly better than analytic techniques.

The next major advance in real-frequency impedance matching techniques came with the extension of the real-frequency principle to the double matching problem. The first double matching real-frequency technique was published by Yarman and Carlin [7]. The basic premise of the technique is that the transducer gain can be calculated from the input reflection coefficient. Further, it can be shown that the denominator of the reflection coefficient can be calculated from the numerator of the reflection coefficient, so the technique only explicitly considers the numerator of the input reflection coefficient. The major advantages of this technique over the technique considered above are that it can do double matching problems and the Hilbert transform is not required to calculate the reactance. The most important problem with this technique is that it requires a good starting point. Yarman and Carlin [7] suggest that an initial point could be obtained from the results of Carlin's [4] single matching technique applied to the case where one of the impedances is assumed to be purely real.

Another double matching technique was proposed by Carlin and Yarman [8]. This technique is based on the single matching technique considered above. The basic principle is to transform the double matching problem into an equivalent single matching problem and then to apply Carlin's [4] single matching technique. The major advantage over the double matching technique considered above is that an initial point can be generated, but this technique is more computationally intensive. An analytic double matching technique is also derived by Carlin and Yarman [8], and as before, the results of the real-frequency technique are significantly better than those obtained with the analytic technique.

More recently, Yarman and Fettweis [9] have proposed a simplified version of the Carlin and Yarman's [8] double matching technique. The basic formulation is the same, but Brune functions are used in this case to simplify the numerical calculations by avoiding the factorisation of polynomials. Numerical stability is also improved. Carlin's [4] single matching technique is again used to initialise this algorithm.

Abrie [2] has proposed a technique based on the transformation-Q of an element. The transformation-Q is defined as the ratio of the imaginary part of an impedance of susceptance to the real part of that impedance or susceptance at any point in a ladder network. The transformation of the impedance or susceptance by each element can be related to the transformation-Q. The transformation-Q is only accurate at one frequency, but broadband matching can be accomplished by using an optimisation algorithm. The algorithm is initialised using an exhaustive search, so this algorithm can be considered to be an optimisation algorithm applied to impedance matching. This technique has the advantages that it can have series and parallel resonant sections, and it can be generalised to consider transmission lines.

Many of the published computer-based methods of impedance matching network design are little more than general purpose optimisation algorithms applied to this problem. A good example of this is the paper by Dedieu *et al.* [82] where a new optimisation algorithm was used to optimise the component values of a given network. The review papers by Bandler [12] and Charalambous [13] give an overview of some of the more important optimisation theory that has been applied to circuit design and supply some examples. The most significant disadvantage of this approach is that the circuit configuration must be supplied to the algorithm. The most important benefit of this approach is that optimisation methods specifically related to impedance matching network design have been developed. An example of this is the work on minimax optimisation that has been done by Madsen *et al.* [83], Bandler *et al.* [84, 85], and Bandler and Charalambous [13], among others.

As in the analytic case, the problem of using transmission lines for impedance matching has not been as well addressed as the lumped problem. However, useful results do exist.

One approach is to start with a lumped prototype, transform some of the elements to transmission lines using a technique such as Richards' transform [74], and optimise the result. Mahdi and Macnee [86] used this approach to transform low pass lumped prototypes to distributed networks, and then optimise the results. Yarman and Aksen [87] applied a similar approach with new transformations that consider more than one frequency. The biggest problem associated with these transformation techniques is that the resonant effects of transmission lines are not accounted for in the lumped prototype, so the final distributed network is often suboptimal.

Another approach to the design of distributed networks is to modify the frequency variable to account for the frequency response of a transmission line. Richard's transform is used to modify lumped techniques by Gibson [88], and by Pandel and Fettweis [89]. Gibson recommends adjusting the normalising frequency (the frequency where the lines are 90° long) to obtain realisable transmission line impedance values, while Pandel and Fettweis allow the normalisation frequency to be adjusted by their algorithm. Both these approaches require all the transmission lines to have the same length, and the number of parallel elements to be specified.

The real-frequency techniques listed above have a number of advantages over analytic and graphical techniques. The first major advantage is that the real data of the systems to be matched over the entire frequency range of interest are used. The analytic techniques use some approximation to the real data and graphical techniques only consider one frequency. This means that problems which cannot be solved using other techniques are possible with the real-frequency techniques. The second major advantage is that the circuit configuration does not have to be selected, it is generated by the algorithm. The last major advantage of the real-frequency techniques is that they produce much better results than analytic techniques [5, 8].

The most important disadvantage of these techniques is that they all (except for the transformation-Q technique) actually design a transfer function rather than the circuit itself. The circuit then has to be extracted from the transfer function using algorithms such as those described by Yarman and Fettweis [9]. This means that these techniques do not allow the element values to be constrained between minimum and maximum values, and often require coupled coils and ideal transformers. A minor problem with most of these real-frequency techniques is that initialisation is obtained from another algorithm, so two algorithms have to be implemented.

2.3.4 Genetic Algorithms

This section will consider the application of genetic algorithms to impedance matching. Significantly, none of the review papers that consider genetic algorithms in electromagnetics give any examples of impedance matching applications [90–92]. However there has been some work done in the field and it will be reviewed here.

The main application of genetic algorithms in impedance matching to date has been the optimisation of tapered transmission line matching sections. Günel [93] derived equations for the input impedance of a non-reciprocal and non-symmetric exponential tapered transmission line and then used a genetic algorithm to optimise the result. Later Günel [94] expanded the procedure to consider lossy parabolic and exponential tapered transmission lines, and to use a hybrid algorithm to speed convergence. Bornholdt [95] considered the more general case where the tapered line can be represented by B-spline curves. The resulting tapered transmission line is then simulated using a three-dimensional method of moments algorithm. The main problem associated with tapered transmission lines is that analysis is complex and time consuming in all but the simplest case. Günel [93] takes the approach of considering comparatively simple shapes that can be analytically analysed to shorten simulation time, whereas Bornholdt [95] considers potentially much more complex shapes, but has to run long simulations in each case.

Raychowdhury *et al.* [96] overcome this difficulty by using an impedance matching network consisting of series transmission lines. The transmission lines are all a quarter of a wavelength long at the centre frequency and a genetic algorithm is used to determine the

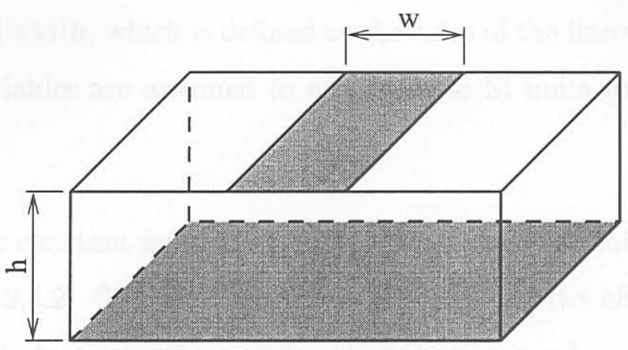


Figure 2.6: Microstrip line.

characteristic impedance of each transmission line. The structure used is very simple and can easily be analysed.

Sun and Lau [97, 98] consider an even simpler situation, a Π-section matching network. The values of the elements are tuned to match an antenna to a resistance using a genetic algorithm.

Genetic algorithms have tremendous potential for use in impedance matching network design algorithms. The papers presented above have not realised this potential because they either take too long to run or they only consider special cases which can be solved with simpler techniques.

2.4 Discontinuity Models

A number of discontinuity models have been published and this section will as give a brief overview of some of these. The models used here were obtained from the references given in the books by Hoffmann [99] and Gupta *et al.* [100], and the manuals for Agilent's ADS microwave design software [101].

A diagram of a microstrip line is shown in Figure 2.6 with the symbols for the width of the line and the height of the substrate indicated. In this document the symbol u is used to denote the normalised width, which is defined as the ratio of the line width to the substrate height (w/h). All variables are assumed to use the base SI units (e.g. m and Hz) unless otherwise stated.

The effective dielectric constant and characteristic impedance of a microstrip line are given in Sections 2.4.1 and 2.4.2. Section 2.4.3 gives equations for the effect of an open-ended microstrip stub. The inductance of a circular via hole to ground is given in Section 2.4.4. A model for the effect of a step in microstrip line width is given in Section 2.4.5. A microstrip T-junction model is given in Section 2.4.6 and the microstrip cross is considered in Section 2.4.7.

2.4.1 Effective Dielectric Constant

The effective dielectric constant is a very important factor that is used in a number of the models developed in the following sections. The first part of this section will present the formula for relative dielectric constant proposed by Hammerstad and Jensen [102]. Then the corrections proposed by Kirschning and Jansen [103] to account for high frequency dispersion will be covered.

At low frequencies the effective dielectric constant of a microstrip line can be given by the following equation proposed by Hammerstad and Jensen [102]:

$$\varepsilon_{eff} = \frac{\varepsilon_r + 1}{2} + \frac{\varepsilon_r - 1}{2}\left(1 + \frac{10}{u}\right)^{-ab} \tag{2.41}$$

where

$$a = 1 + \frac{1}{49}\ln\left[\frac{u^4 + (u/52)^2}{u^4 + 0.432}\right] + \frac{1}{18.7}\ln\left[1 + \left(\frac{u}{18.1}\right)^3\right], \tag{2.42}$$

$$b = 0.564\left(\frac{\varepsilon_r - 0.9}{\varepsilon_r + 3}\right)^{0.053} \tag{2.43}$$

and ε_r is the relative dielectric constant of the substrate.

The original paper states that the accuracy of this model is better than 0.2% for at least

$\varepsilon_r \leq 128$ and $0.01 \leq u \leq 100$. However Kirschning and Jansen [103] show that this model is only valid at low frequencies.

To overcome this limitation, Kirschning and Jansen [103] proposed the following set of equations that account for dispersion in the relative dielectric constant:

$$\varepsilon_{eff}(f) = \varepsilon_r - \frac{\varepsilon_r - \varepsilon_{eff}(f=0)}{1+P(f)} \tag{2.44}$$

where

$$P(f) = P_1 P_2[(0.1844 + P_3 P_4)10fh]^{1.5763} \tag{2.45}$$

with

$$P_1 = 0.27488 + [0.6315 + 0.525/(1+0.157fh)^{20}] \times u - 0.065683\exp(-8.7513u), \tag{2.46}$$

$$P_2 = 0.33622[1 - \exp(-0.03442\varepsilon_r)], \tag{2.47}$$

$$P_3 = 0.0363\exp(-4.6u) \times \{1 - \exp[-(fh/3.87)^{4.97}]\}, \tag{2.48}$$

and

$$P_4 = 1 + 2.751\{1 - \exp[-(\varepsilon_r/15.916)^8]\}, \tag{2.49}$$

and where f is the frequency in GHz and h is the substrate height in cm.

This model is accurate to better than 0.6% for $0.1 \leq u \leq 100$, $1 \leq \varepsilon_r \leq 20$, and $0 \leq h/\lambda_0 \leq 0.13$. Unfortunately, Kirschning and Jansen do not state which model they used for $\varepsilon_{eff}(f=0)$ in (2.44), but the model given above is the most accurate one referenced, so the accuracy claimed above should apply.

At this point the finite thickness of the conductor and losses in the line have been ignored because these effects would introduce a large amount of extra complexity for small accuracy gains. March [104] gives an equation which modifies the line width to account for finite conductor thickness and a comprehensive overview of microstrip losses is given by Denlinger [105].

2.4.2 Characteristic Impedance

A model for the characteristic impedance of microstrip lines is given in this section. The formulae presented here were proposed by Hammerstad and Jensen [102].

The following equation gives the value of the characteristic impedance for a microstrip line with a free space substrate:

$$Z_{0l} = \frac{\eta_0}{2\pi} \ln \left[\frac{f}{u} + \sqrt{1 + \left(\frac{2}{u} \right)^2} \right] \tag{2.50}$$

where

$$f = 6 + (2\pi - 6) \exp \left[- \left(\frac{30.666}{u} \right)^{0.7528} \right] \tag{2.51}$$

and η_0 is the free space wave impedance ($\sqrt{\mu_0/\varepsilon_0}$).

Hammerstad and Jensen [102] claim that this equation is accurate to less than 0.01% for $u \leq 1$, and to less than 0.03% for $u \leq 1000$.

When a substrate other than free space is used, the characteristic impedance is obtained by dividing Z_{0l} by the square root of the effective dielectric constant:

$$Z_0 = \frac{Z_{0l}}{\sqrt{\varepsilon_{eff}}} . \tag{2.52}$$

As in Section 2.4.1 the effects of the finite thickness of the conductor have been ignored, with correction factors for this effect being given by March [104].

2.4.3 Open End

This section will consider an open-ended microstrip line as shown in Figure 2.7. The model used here was proposed by Kirschning *et al.* [106] and consists of an extension of the line length.

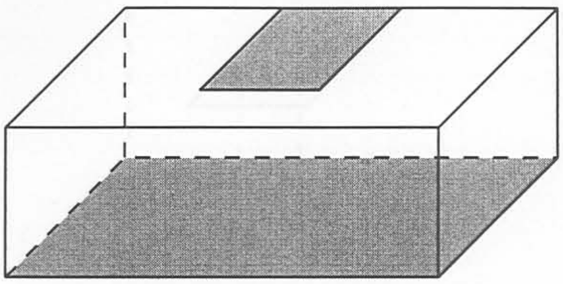


Figure 2.7: Open-ended microstrip line.

The length increase as a factor of the substrate height is given by

$$\Delta l/h = (\xi_1 \xi_3 \xi_5 / \xi_4) \tag{2.53}$$

where

$$\xi_1 = 0.434907 \frac{\varepsilon_{eff}^{0.81} + 0.26}{\varepsilon_{eff}^{0.81} - 0.189} \times \frac{u^{0.8544} + 0.236}{u^{0.8544} + 0.87}, \tag{2.54}$$

$$\xi_2 = 1 + \frac{u^{0.371}}{2.358\varepsilon_r + 1}, \tag{2.55}$$

$$\xi_3 = 1 + \frac{0.5274 \arctan[0.084 u^{1.9413/\xi_2}]}{\varepsilon_{eff}^{0.9236}}, \tag{2.56}$$

$$\xi_4 = 1 + 0.0377 \arctan[0.067 u^{1.456}] \times \{6 - 5\exp[0.036(1 - \varepsilon_r)]\}, \tag{2.57}$$

$$\xi_5 = 1 - 0.218 \exp(-7.5u), \tag{2.58}$$

and Δl is the line length extension.

Kirschning *et al.* [106] claim that this model is accurate to within 2.5% for $0.01 \leq u \leq 100$ and $\varepsilon_r < 50$ at 1 GHz when compared to numerical results listed in the paper. The effective dielectric constant (ε_{eff}) model used by Kirschning *et al.* is the one given by Hammerstad and Jensen [102], and is supplied in Section 2.4.1.

2.4.4 Via Holes

A via hole is present whenever a microstrip line is shorted to ground as shown in Figure 2.8. The model presented here is due to Goldfarb and Pucel [107].

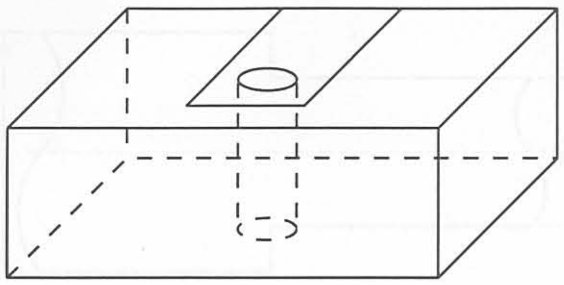


Figure 2.8: Via hole to ground.

The model consists of a parasitic inductance to ground with a value of

$$L_{via} = \frac{\mu_0}{2\pi}\left[h \cdot \ln\left(\frac{h+\sqrt{r^2+h^2}}{r}\right) + \frac{3}{2}\left(r - \sqrt{r^2+h^2}\right)\right] \tag{2.59}$$

where r is the radius of the via.

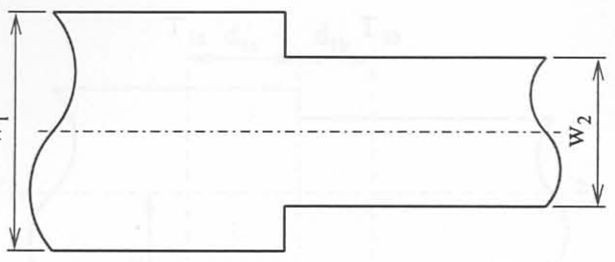
The accuracy of this model is not explicitly given by Goldfarb and Pucel [107]. All that is stated is that equation agrees very closely with simulations of vias in 100–635 μm substrates with $0.2 < d/h < 1.5$, $2.2 < \varepsilon_r < 20$, and $1 < u < 2.2$. The dependency of the via inductance of the width of the pad is specified as being less than 4% for a very large range of u ratios.

2.4.5 Width Step

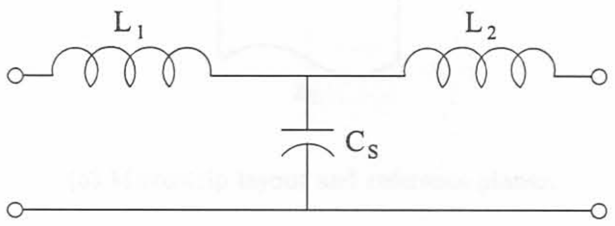
An abrupt junction between two microstrip lines leads to a step in the line width as shown in Figure 2.9(a). The model presented here is adapted from Hammerstad [102] and Gupta *et al.* [108]. The model is a T network which consists of inductors in series with each of the lines and a parallel capacitor between them as shown in Figure 2.9(b). Both symmetrical and asymmetrical steps are considered by Hammerstad [102], but only the symmetrical forms are given here.

The capacitance is given by

$$C_s = \left(\frac{\sqrt{\varepsilon_{eff1}}}{Z_{L1}c_0} - \varepsilon_0\varepsilon_r u_1\right)\frac{w_1 - w_2}{2} \tag{2.60}$$



(a) Microstrip layout.



(b) Equivalent circuit.

Figure 2.9: Microstrip width step.

where C_s is the capacitance value, Z_{Ln} is the characteristic impedance of line n, w_n is the width of line n, and the subscripts 1 and 2 denote the wider and narrower lines respectively.

Hammerstad [102] gives two models for the total series inductance in the model. The simpler of the two forms gives better results than the more complex form and is given by

$$L_s = \frac{\mu_0}{\pi} \ln \left[1/\sin \left(\frac{\pi Z_{L1}}{2 Z_{L2}} \sqrt{\frac{\varepsilon_{eff1}}{\varepsilon_{eff2}}} \right) \right] h \,. \tag{2.61}$$

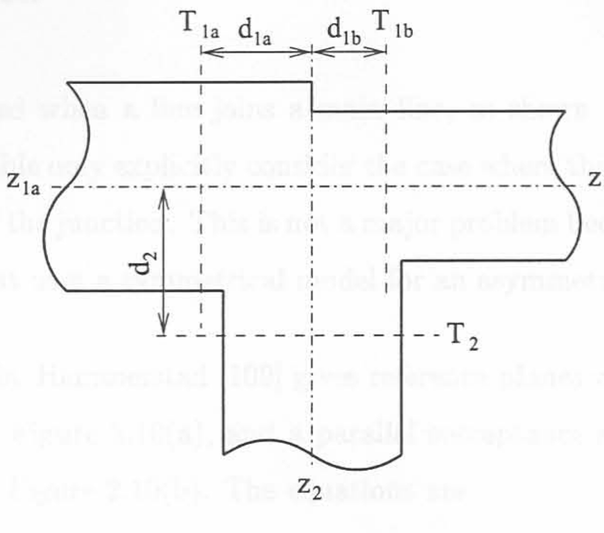
This inductance then has to be split to form the two series inductors. This is done using the following equation:

$$L_n = \frac{L_{wn}}{L_{w1} + L_{w2}} L_s \tag{2.62}$$

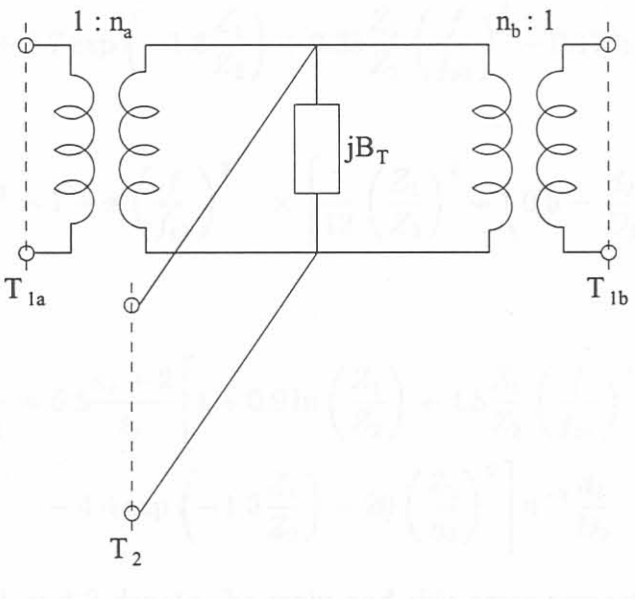
where L_n is the inductance in series with line n, and L_{wn} is the inductance per unit length of the transmission line given by

$$L_{wn} = \frac{Z_{0n} \sqrt{\varepsilon_{eff\ n}}}{c_0} \tag{2.63}$$

where c_0 is the speed of light in a vacuum.



(a) Microstrip layout and reference planes.



(b) Equivalent circuit.

Figure 2.10: Microstrip T-junction.

Unfortunately no error bounds are specified for these equations.

2.4.6 T-Junction

A T-junction is formed when a line joins a main line, as shown in Figure 2.10(a). The models that are available only explicitly consider the case where the main line has the same width on both sides of the junction. This is not a major problem because Hammerstad [109] suggests a method that uses a symmetrical model for an asymmetrical T-junction.

The model proposed by Hammerstad [109] gives reference planes relative to the middle of each line as shown in Figure 2.10(a), and a parallel susceptance and transformers in the main line as shown in Figure 2.10(b). The equations are

$$\frac{d_1}{D_2} = 0.055\left[1 - 2\frac{Z_1}{Z_2}\left(\frac{f}{f_{p1}}\right)^2\right]\frac{Z_1}{Z_2}, \tag{2.64}$$

$$\frac{d_2}{D_1} = 0.5 - \left[0.05 + 0.7\exp\left(-1.6\frac{Z_1}{Z_2}\right) + 0.25\frac{Z_1}{Z_2}\left(\frac{f}{f_{p1}}\right)^2 - 0.17\ln\left(\frac{Z_1}{Z_2}\right)\right]\frac{Z_1}{Z_2}, \tag{2.65}$$

$$n^2 = 1 - \pi\left(\frac{f}{f_{p1}}\right)^2 \times \left[\frac{1}{12}\left(\frac{Z_1}{Z_2}\right)^2 + \left(0.5 - \frac{d_2}{D_1}\right)^2\right], \tag{2.66}$$

and

$$\frac{B_T}{Y_2}\frac{\lambda_1}{D_1} = 5.5\frac{\varepsilon_r + 2}{\varepsilon_r}\left[1 + 0.9\ln\left(\frac{Z_1}{Z_2}\right) + 4.5\frac{Z_1}{Z_2}\left(\frac{f}{f_{p1}}\right)^2 \right. \tag{2.67}$$

$$\left. - 4.4\exp\left(-1.3\frac{Z_1}{Z_2}\right) - 20\left(\frac{Z_2}{\eta_0}\right)^2\right]n^{-2}\frac{d_1}{D_2} \tag{2.68}$$

where the subscripts 1 and 2 denote the main and side arms respectively, d_n is the offset from the centre line, n is the number of turns in the main arm transformer, B_T is the parallel susceptance, and Y_n is the inverse of the line characteristic impedance Z_n. The remaining parameters are:

$$\lambda_n = \frac{c_0}{f\sqrt{\varepsilon_{effn}}}, \tag{2.69}$$

$$f_{pn} = \frac{c_0 Z_n}{2\eta_0 h}, \tag{2.70}$$

and

$$D_n = \frac{\eta_0 h}{Z_n \sqrt{\varepsilon_{effn}}} \tag{2.71}$$

where λ_n is the wavelength in the line, f_{pn} is the first higher mode cutoff frequency of the equivalent parallel plate transmision line, and D_n is the line width of the equivalent parallel plate transmission line.

While these formulae are only valid for the case where the main arm has the same line width on both sides of the discontinuity, Hammerstad [109] proposes the following procedure to allow for an asymmetric main arm. Use the impedance of the main arm under consideration when calculating the main arm parameters. Use the geometric mean of the main line impedances when calculating the side arm parameters.

Unfortunately, no error bounds are supplied for this model.

2.4.7 Cross

This discontinuity is formed when two lines cross as shown in Figure 2.11. All the models that have been published are only valid for the case where the two lines that cross are the same width on both sides of the discontinuity. This dramatically limits these models' applicability and accuracy. Additionally, the model given by Garg and Bahl [110] only gives capacitance values for an alumina substrate ($\varepsilon_r = 9.9$). For these reasons, two parallel T-junctions will be used to approximate a cross as suggested by Abrie [2].

When a cross is approximated by two parallel T-junctions, the calculations for the side arms are performed in exactly the same way as for a T-junction by using the parameters of the side arm under consideration and the geometric mean of the main arm parameters. The calculations for the main arm parameters are accomplished using the parameters of the main arm under consideration and the geometric mean of the side arm parameters. The susceptance in the T-junction model is only used once. This is obviously not a particularly satisfying solution to the problem, but the other options are just as bad.

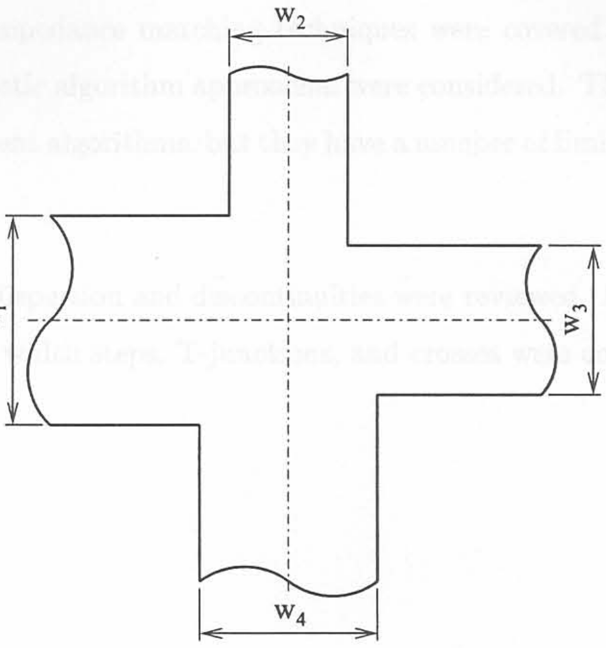


Figure 2.11: Microstrip Cross.

2.5 Summary

The theory used in this dissertation was briefly reviewed in this chapter. A reasonably detailed overview of each topic was given because the information in this chapter was used to motivate choices made later.

Optimisation algorithms form the basis of most CAD tools. The formulation of an optimisation problem and some special cases were presented. A number of local, global, and combinatorial optimisation techniques were considered. Hybrid algorithms that combine a number of different algorithms were briefly discussed.

An introduction to genetic algorithms, with the emphasis on the underlying principles rather than a specific paradigm, was given. The most important parts of a genetic algorithm were presented with selection schemes, representation, and genetic operators being considered in detail. Messy genetic algorithms were presented as a means to overcome some of the difficulties faced when applying genetic algorithms to practical problems.

The most important impedance matching techniques were covered. Graphical, analytic, real-frequency, and genetic algorithm approaches were considered. The real-frequency techniques are the best current algorithms, but they have a number of limitations that motivated this work.

Models for microstrip dispersion and discontinuities were reviewed. Microstrip lines, open-ended stubs, via holes, width steps, T-junctions, and crosses were considered.

Chapter 3

Implementation

This chapter will consider the implementation of the algorithm developed during this dissertation. Most of the work involves bringing together a number of the techniques and theories discussed in Chapter 2. The principle of this dissertation is to assume that an impedance matching problem is simply an optimisation problem and to develop an algorithm to solve that problem.

Section 3.1 will consider the approach used during the development of the algorithm. Section 3.2 will then present the genetic algorithm used. The calculation of the fitness function values used by the genetic algorithm is covered in Section 3.3. Lastly, the local optimiser used is presented in Section 3.4.

3.1 Approach

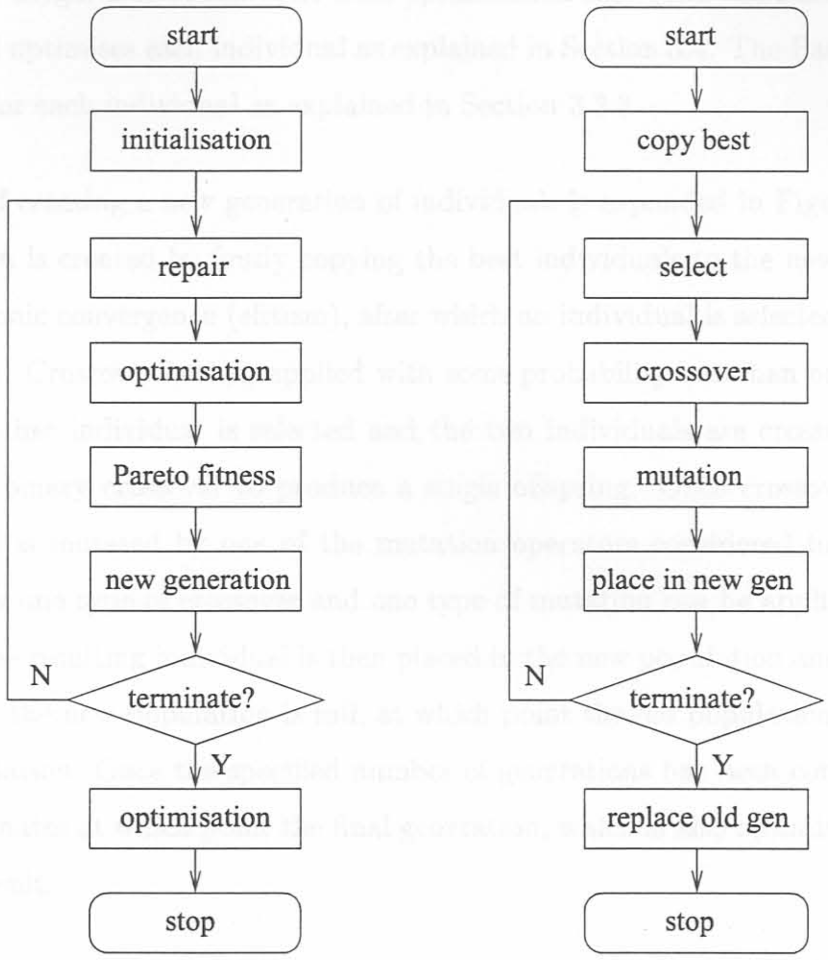
The approach used to develop the algorithm that is the basis of this dissertation is given in this section. This includes how an impedance matching problem can be considered as an optimisation problem, and why a genetic algorithm coupled with a local optimiser was used to solve that problem.

Impedance matching network design is the process of designing a circuit that matches one impedance to another with a specified gain. This can be viewed as an optimisation problem where the objective is to minimise the gain error given the source and load impedances, and the desired gain.

The problem of optimising a circuit for any purpose, including impedance matching, can be considered to consist of combinatorial, global, and local optimisation problems. The combinatorial part of the problem involves the structure of the circuit. This entails deciding between a number of circuit elements including inductors, capacitors and transmission lines, and whether a component should be in series or parallel. The global and local parts of the problem consider the component values. There are a number of local optima in an impedance matching problem, so a global optimisation algorithm is necessary to find the best of these. Once a global optimiser has found a point near the global optimum, a local optimiser is used to rapidly converge to the global optimum.

Of the optimisation algorithms considered in Section 2.1, genetic algorithms and simulated annealing can be used for both combinatorial and global optimisation. Using one of these algorithms would thus eliminate the need to implement separate combinatorial and global optimisation algorithms. Genetic algorithms were chosen ahead of simulated annealing because genetic algorithms use a population of individuals allowing more than one solution to a problem to be obtained. While genetic algorithms are very useful for combinatorial and global optimisation, their local optimisation properties are poor, so the hybrid system proposed by Renders and Flasse [34] was used to speed convergence. The implementation of the local optimiser is discussed further in Section 3.4.

A number of constants and options were implemented in this algorithm. Examples of constants include minimum and maximum component values, and crossover and mutation probabilities. Examples of options include the three different fitness functions discussed in Section 3.3. All the constants and options were implemented as constants that cannot be changed by the program and this has the drawback that all the constants and options must be set before the program is compiled. The major advantage of this approach is



(a) Complete algorithm.

(b) New generation creation.

Figure 3.1: Flow charts.

that it results in faster code than the case where constants and options can be changed after compilation because the compiler can apply more optimisations to the code. A large number of tests were run during this dissertation, so program speed was considered more important than the problems involved with re-compiling the program every time a constant is changed.

A flow chart of the algorithm implemented here is given in Figure 3.1(a). Initialisation is the initialisation of the algorithm and is done randomly in this case. The repair stage eliminates any illegal individuals. The local optimisation algorithm calculates fitnesses and gradients, and optimises each individual as explained in Section 3.4. The Pareto-like fitness is calculated for each individual as explained in Section 3.3.3.

The process of creating a new generation of individuals is expanded in Figure 3.1(b). The new generation is created by firstly copying the best individuals to the new generation to ensure monotonic convergence (elitism), after which an individual is selected using tournament selection. Crossover is then applied with some probability less than one. If crossover is applied another individual is selected and the two individuals are crossed using either arithmetic or binary crossover to produce a single offspring. Once crossover is complete the individual is mutated by one of the mutation operators considered in Section 3.2.2. Note that only one type of crossover and one type of mutation can be applied to each new individual. The resulting individual is then placed in the new population and the process is repeated until the new population is full, at which point the old population is replaced by the new population. Once the specified number of generations has been considered the algorithm terminates at which point the final generation, which is also optimised, is returned as the final result.

Each of the steps shown in the flow chart in Figure 3.1 will be considered in more detail in the following sections.

3.2 Genetic Algorithm

The implementation of the genetic algorithm portion of this dissertation will be presented in this section. The initialisation, selection scheme, termination criterion, elitism, representation, and genetic operators that were used are covered.

The usual genetic algorithm practice of randomly initialising the population was used. Other possibilities considered were to initialise the population using a grid search or to seed the population with the results of other algorithms. The populations used are too small to make a grid initialisation practical because, for example, if an individual consists of six lumped elements there are $4^6 = 4096$ possible ways to choose those elements. While some of those combinations are redundant (for example two inductors in series), this type of initialisation is still impractical for reasonable population sizes. Seeding the population with the results of other algorithms would require the implementation of a large number of other algorithms, and while this is possible, it would be too time consuming for this dissertation.

Two options for determining the initial length of individuals were considered. The first option generates individuals with equal probabilities of being any length from one element to the maximum number of elements allowed, but the removal of illegal and redundant elements means that this scheme will actually favour the production of short individuals. The second possibility is to make all individuals as long as possible prior to removal of illegal and redundant elements. This is useful because the algorithm tends to favour the creation of short individuals due to removal of elements, and longer individuals represent more complex solutions which require more samples to arrive at a good result. For this reason, the second option was implemented.

Of the selection schemes considered in Section 2.2.2 tournament selection was used in this dissertation because it is very simple to implement and the bias towards good individuals can be adjusted. The algorithm was terminated after a fixed number of generations. Elitism was implemented as an option that can be set before compilation. When elitism is used the best individual (or individuals in the Pareto case) is copied to the new population unaltered.

The default genetic algorithm parameters are given in Table 3.1.

The representation used is considered in Section 3.2.1 and the genetic operators used are presented in Section 3.2.2.

Table 3.1: Genetic algorithm operator default probabilities.

Genetic Operator	Value
Arithmetic crossover	0.4
Binary crossover	0.3
Boundary mutation	0.03
Uniform mutation	0.02
Non-uniform mutation	0.1
Binary mutation	0.1

3.2.1 Representation

The representation used for this dissertation is a unique combination of the discrete and continuous parts of the problem. Additionally, the representation used for the continuous variables allows both binary and floating point operators to be used – something that has not been achieved before. This section will consider this representation with the emphasis on the new features.

The circuit configuration used was limited to ladder networks. This simplifies the representation because a ladder network can be represented as a string of circuit elements. Each individual thus consists of a number of circuit elements in order from load to source. The maximum number of circuit elements in each individual is set when the program is compiled. A ladder network can only have two elements in parallel at any point (a cross structure), so more than two parallel elements at a point are considered redundant and are eliminated. Similarly, in the purely lumped case, two capacitors or inductors in series or parallel are redundant and the second element is removed. Series elements are not eliminated in the distributed case and when discontinuities are used because there is no series redundancy in these cases. Another approach to dealing with redundant elements would have been to ignore them, but this can lead to difficulties in calculating a network's length and can cause poor combinations of elements to exist in abeyance (present in an individual,

Table 3.2: Element Representation.

Circuit element	Bits				
	29	28	27	26–13	12–0
parallel inductor	0	0	0	value (log)	
parallel capacitor	0	0	1	value (log)	
series inductor	0	1	0	value (log)	
series capacitor	0	1	1	value (log)	
parallel shorted stub	1	0	0	width (log)	length (linear)
parallel open stub	1	0	1	width (log)	length (linear)
series transmission line	1	1	0	width (log)	length (linear)
illegal	1	1	1		

but not used in that individual, although it might be used in future generations).

The full representation consists of 30 bits. Most desktop computers today use a 32 bit word length, so the use of 30 bits allows 2 bits for future expansion of the number of element types. The first three of these bits are used to encode the type of element and the remaining 27 bits encode the element's parameter values as shown in Table 3.2.

From Table 3.2 it is clear that bit 27 is 0 for inductive elements and 1 for capacitive elements, bit 28 is 0 for parallel elements and 1 for series elements, and bit 29 is 0 for lumped elements and 1 for distributed elements. The lumped elements only require one variable to represent their value, so they use the full remaining 27 bits for this purpose. Distributed elements require two values for both characteristic impedance (determined by the width in a microstrip system) and length, so 14 bits are used to represent the line width and 13 bits are used to represent the line length. The transmission line lengths are linearly distributed between the minimum and maximum specified values. The values of lumped elements vary logarithmically and the relationship between the width of a microstrip line and its characteristic impedance is approximately logarithmic, so a logarithmic representation was used in these cases. Binary values are distributed linearly, so a transformation was necessary.

The uniformly distributed binary values are converted to logarithmically distributed values using

$$r = y_{\min}\left(\frac{y_{\max}}{y_{\min}}\right)^{x} = y_{\min}e^{\left[x\ln\left(\frac{y_{\max}}{y_{\min}}\right)\right]} \tag{3.1}$$

where r is the representation of the variable y, $y_{\max}$ and $y_{\min}$ are the maximum and minimum values of y, and x is a uniformly distributed variable over the range [0, 1].

The first advantage of this representation is that the value of the variable y is constrained to the range $[y_{\min}, y_{\max}]$ allowing the algorithm to be constrained to realisable parameter ranges. The second advantage is that this allows the use of uniformly distributed variables to represent logarithmic parameters. This means that the value of x in (3.1) can be implemented as a binary string (a linear representation), allowing both the binary and floating-point genetic operators considered in Section 2.2.4 to be used on problems with floating-point variables.

An important consideration at this point is the decimal precision of the representation where values of $y_{\max}$, $y_{\min}$ and the number of bits used to represent x are given. The decimal precision of an arbitrary representation is

$$d = \left\lfloor \log\left(\frac{5}{err}\right)\right\rfloor + \lfloor \log(y)\rfloor \tag{3.2}$$

where

$$err = |y - r|, \tag{3.3}$$

d is the precision in number of decimal digits, and the $\lfloor\cdot\rfloor$ (floor) operator indicates that the argument must be rounded to the greatest integer value less than or equal to the argument. The first term of (3.2) gives the number of decimal digits precision after the decimal point and the second term gives the position of the most significant decimal digit. The value of the representation is given in (3.1), and the correct value of the variable is

$$y = y_{\min}\left(\frac{y_{\max}}{y_{\min}}\right)^{x_e} = y_{\min}e^{\left[x_e\ln\left(\frac{y_{\max}}{y_{\min}}\right)\right]} \tag{3.4}$$

where x_e is the value of x that would make the representation r exactly equal to the variable

y. The representation can be rewritten in terms of x and x_e as shown below.

$$x = \frac{\ln\left(\frac{r}{y_{\min}}\right)}{\ln\left(\frac{y_{\max}}{y_{\min}}\right)} \tag{3.5}$$

$$x_e = \frac{\ln\left(\frac{y}{y_{\min}}\right)}{\ln\left(\frac{y_{\max}}{y_{\min}}\right)} \tag{3.6}$$

The maximum difference between x and x_e is half of the smallest step in x. A binary value with n bits divides a range into $2^n - 1$ steps, so the maximum error is half this value:

$$|x_e - x| = \left|\frac{\ln\left(\frac{y}{y_{\min}}\right)}{\ln\left(\frac{y_{\max}}{y_{\min}}\right)} - \frac{\ln\left(\frac{r}{y_{\min}}\right)}{\ln\left(\frac{y_{\max}}{y_{\min}}\right)}\right| \leq \frac{1}{2\left(2^n - 1\right)} \; . \tag{3.7}$$

This result can now be used to obtain the limits on the value of r as shown below.

$$\left|\ln\left(\frac{y}{y_{\min}}\right) - \ln\left(\frac{r}{y_{\min}}\right)\right| \leq \frac{\ln\left(\frac{y_{\max}}{y_{\min}}\right)}{2\left(2^n - 1\right)} \tag{3.8}$$

$$\left|\ln\left(\frac{y}{r}\right)\right| \leq \frac{\ln\left(\frac{y_{\max}}{y_{\min}}\right)}{2\left(2^n - 1\right)} \tag{3.9}$$

$$y \exp\left[-\frac{\ln\left(\frac{y_{\max}}{y_{\min}}\right)}{2\left(2^n - 1\right)}\right] \leq r \leq y \exp\left[\frac{\ln\left(\frac{y_{\max}}{y_{\min}}\right)}{2\left(2^n - 1\right)}\right] \tag{3.10}$$

$$y\left(\frac{y_{\max}}{y_{\min}}\right)^{\frac{-1}{2(2^n-1)}} \leq r \leq y\left(\frac{y_{\max}}{y_{\min}}\right)^{\frac{1}{2(2^n-1)}} \tag{3.11}$$

The values of the range limits do not differ much as can be seen by the fact that the exponents are both very close to zero when n is large. These values of r can now be substituted into (3.3) to give

$$err = \begin{cases} y - r \geq y - y\left(\frac{y_{\max}}{y_{\min}}\right)^{\frac{-1}{2(2^n-1)}} & \text{if } y \geq r \\ r - y \leq y\left(\frac{y_{\max}}{y_{\min}}\right)^{\frac{1}{2(2^n-1)}} - y & \text{if } y < r \end{cases} \tag{3.12}$$

and

$$y\left[1 - \left(\frac{y_{\max}}{y_{\min}}\right)^{\frac{-1}{2(2^n-1)}}\right] \leq err \leq y\left[\left(\frac{y_{\max}}{y_{\min}}\right)^{\frac{1}{2(2^n-1)}} - 1\right] \; . \tag{3.13}$$

As above, the limits are very similar when n is large, so this result can be approximated by the upper limit:

$$err \approx y\left[\left(\frac{y_{\max}}{y_{\min}}\right)^{\frac{1}{2(2^n-1)}} - 1\right]. \tag{3.14}$$

This value can now be substituted into (3.2) to obtain the decimal precision of the result:

$$d \approx \left\lfloor \log\left(\frac{5}{y\left[\left(\frac{y_{\max}}{y_{\min}}\right)^{\frac{1}{2(2^n-1)}} - 1\right]}\right)\right\rfloor + \lfloor \log(y) \rfloor \tag{3.15}$$

where the result is approximate because the value of *err* is approximate. In (3.15) the precision depends on the value of the variable y. The value of y can be removed by making the following approximation:

$$d \approx \left\lfloor \log\left(\frac{5}{y\left[\left(\frac{y_{\max}}{y_{\min}}\right)^{\frac{1}{2(2^n-1)}} - 1\right]}\right) + \log(y) \right\rfloor - 1 \tag{3.16}$$

$$\approx \left\lfloor \log\left(\frac{5}{\left[\left(\frac{y_{\max}}{y_{\min}}\right)^{\frac{1}{2(2^n-1)}} - 1\right]}\right)\right\rfloor - 1\,. \tag{3.17}$$

The subtraction of one is necessary because there will be some value of y where the result is greater than the correct value given by (3.15) if n is large enough. This error cannot be greater than one because the sum of the fractional parts of both terms in the floor operator in (3.16) must be less than two.

The component ranges used are given in Table 3.3, where λ is the wavelength of a transmission line and h is the height of a microstrip substrate. The component ranges considered during parameter tuning were intentionally chosen to be very large to give worst-case results when the algorithm was tested in Chapter 4. The transmission line impedances are the minimum and maximum impedances obtained when the effective dielectric constant of a microstrip line can vary from 1 to 10, and the width of the microstrip line can vary from 0.01 to 100 times the substrate height. The number of decimal digits' precision for the logarithmically distributed variables is at least 3 in all cases and the linearly distributed variables have step sizes that are smaller than 1/8000 of the total range, so the accuracy of

Table 3.3: Component Ranges.

Component Type	Minimum Value	Maximum Value	Precision (decimal digits)
Lumped	10^{-15} F or H	10^{-6} F or H	6
Transmission line length	0.01 λ	3 λ	step: $3.650 \times 10^{-6}\lambda$
Transmission line impedance	1 Ω	400 Ω	3
Transmission line length	2 h	100 h	step: $1.196 \times 10^{-2}h$
Transmission line width	0.1 h	10 h	3

Table 3.4: Microstrip substrate parameters.

Problems	Relative dielectric constant (ε_r)	Height (mm)
2, 3, 4, 5	1	3
7, 8	2	0.1
1, 6	5	0.25
9, 10	10	0.5

the representation is seen to be good. The minimum microstrip line length was chosen to ensure that there is a reasonable separation between parallel elements to minimise coupling. A transmission line length of zero was not used because this leads to infinite impedances for parallel open stubs.

The substrates used for the microstrip problems are given in Table 3.4 for the ten test problems considered in Section 4.1 on page 106. The substrate relative dielectric constants were chosen to give a range of typical values, and the substrate heights were chosen to be approximately one hundredth of a wavelength in the substrate at the highest frequency. All elements are assumed to be lossless and the conductor is assumed to be infinitesimally thin. The via diameter was equal to half the substrate height for each problem.

3.2.2 Genetic Operators

The genetic operators used in this dissertation are covered in this section. One of the unique features of this dissertation is that all the operators in Section 2.2.4 except simple crossover are used. A description of how the operators are applied is given below followed by the implementation details of the crossover and mutation operators.

A flow chart of the algorithm is given in Figure 3.1 on page 79. An individual is selected before the genetic operators are applied. Two individuals are required for crossover so a second individual is selected if crossover is applied. Mutation is applied to the individual generated by crossover. Crossover and mutation are applied with some probability less than one so not all individuals are modified by these operators. Individuals that are not modified simply pass through the operator unaltered and are applied to the next stage of the algorithm. While a number of crossover and mutation operators are implemented, only one crossover and one mutation operator can be applied to each individual.

3.2.2.1 Crossover

Binary and arithmetic crossover were implemented and their operation is described below. The two parents used were not allowed to be identical in this dissertation to increase the genetic diversity of the population.

The representation consists of a number of 30 bit elements in each individual. Variable elements were allowed by using a crossover scheme similar to that used in the messy genetic algorithms considered in Section 2.2.6. If the parents are given by

$$\begin{array}{cccc|c|cccc} & & x_5 & x_4 & x_3 & x_2 & x_1 & x_0 \\ y_6 & y_5 & y_4 & y_3 & y_2 & y_1 & y_0 & \end{array} \tag{3.18}$$

where x_i and y_j represent circuit elements and the cross point is marked by $|\cdot|$, then the offspring is given by

$$x_5 \; \Big| \; a \; \Big| \; y_2 \quad y_1 \quad y_0 \tag{3.19}$$

where a represents some combination of x_4 and y_3. To implement this process a cross point must be chosen for each parent. Care must be taken to ensure that the offspring is not longer than the maximum allowable length by limiting the position of the cross point in the second parent.

An option to always choose the cross point in the second parent so as to make the offspring as long as possible was implemented. This option allows the variable length genetic algorithm to function as a normal fixed length genetic algorithm by making all individuals as long as possible when Pareto optimality is not used. This option also helps to ensure that there are more long individuals than short individuals. This is necessary because longer individuals represent a more difficult problem than short individuals and thus require more processing.

Binary crossover is implemented as above with a being generated by performing binary crossover on the two parents' elements at that position. The cross point is chosen uniformly from before the first bit to after the last bit of the element representation. This allows a to be generated solely by the one parent, eliminating the need for a simple crossover operator. It is possible for binary crossover to generate the illegal elements, so a repair operator that eliminates illegal elements was implemented.

Arithmetic crossover is very similar to binary crossover except that the generation of a is different. The circuit element type of a is the same as the circuit element type of the first parent. The element values are then crossed using (2.29) on page 44. If two transmission lines are crossed, then the length and the width are crossed independently using the same weight (r). In any other case bits 26–0 of the representation are assumed to contain one value and are crossed accordingly. The weight used for crossover can vary from 0 to 1 so a can be generated by only one of the parents as with binary crossover. This again eliminates the need for a simple crossover operator.

3.2.2.2 Mutation

The mutation operators in this case all work in the same basic way. An element in the individual is chosen and is then mutated. As with crossover, only one type of mutation can operate on each individual.

Binary mutation works by inverting bits in an individual. One of the elements of the individual is chosen, and then one of the bits of that element is inverted. Binary mutation can generate illegal individuals, but these are deleted by the repair algorithm.

Non-uniform mutation is described in Section 2.2.4.1. An element of an individual is chosen and then mutated. Both the width and length of a distributed element are mutated independently, but with the same parameters.

Boundary mutation sets an element's value to either its minimum or maximum value with equal probability. In this case it is not necessary to consider lumped and distributed elements separately.

Uniform mutation reinitialises an element of an individual by assigning a random element type and value to that element.

3.3 Fitness Calculation

The calculation of the fitness of each individual will be considered in this section. Lumped, distributed, and mixed lumped and distributed networks are possible. Further a number of discontinuity models were implemented and a type of multi-objective ranking similar to Pareto ranking was used. Each of these points is considered below starting with a discussion of the error functions used.

3.3.1 Error Functions

The calculation of the fitness starts at the load and calculates the input impedance by taking each element into account. The final result of a fitness calculation in this dissertation uses one of four types of error, namely Voltage Standing Wave Ratio (VSWR), Maximum Relative Deviation (MRD), or the gain error in decibels (dB). In each case the maximum value of the error with frequency is used as the fitness value, so this is a minimax problem.

VSWR is a well known error function for measuring the quality of a match, and is calculated from the reflection coefficient:

$$\text{VSWR} = \frac{1+|\Gamma|}{1-|\Gamma|} \tag{3.20}$$

where Γ is the reflection coefficient calculated using

$$\Gamma = \frac{Z_L - Z_0^*}{Z_L + Z_0} \tag{3.21}$$

where Z_0 is the complex normalising impedance, and Z_L is the other impedance at the junction under consideration [3]. The highest VSWR value over the frequency range of interest is used as the fitness. VSWR can only be used where the objective is to perfectly match the load and source impedances because its definition does not include a gain specification. The MRD and decibel errors overcome this limitation by using the transducer power gain (G_T) to determine the error.

The transducer power gain is defined as the ratio of the power delivered to the load to the power available from the source [3]. The magnitude of S_{11} squared is equal to the ratio of the power reflected by the network to the power available from the source [3] when the network is terminated by its normalising impedances. All the power entering a lossless network must emerge at the output, and a passive network cannot add any power to a signal. This means that the transducer gain of a passive, lossless network is the proportion of the power available from the source that is not reflected back to the source, and is given by

$$G_T = 1 - |S_{11}|^2 \tag{3.22}$$

$$= 1 - |\Gamma|^2 \,. \tag{3.23}$$

Both ports are guaranteed to be terminated in their normalising impedances in this algorithm because the source and load impedances of the required design are used as the normalising impedances. The equivalence of $|S_{11}|$ and $|\Gamma|$ can be shown by using the definition of Γ given in (3.21) and the equation for S_{11}:

$$S_{11} = \frac{Z_L - Z_0^*}{Z_L + Z_0} \,. \tag{3.24}$$

The difference between the MRD and decibel errors lies in how they use the transducer gain.

MRD is calculated using

$$\text{MRD} = \left| \frac{G_T(f)}{G_{Topt}(f)} - 1 \right| \tag{3.25}$$

where $G_T(f)$ and $G_{Topt}(f)$ are the actual and desired values of transducer gain at frequency f respectively [2]. The drawback of this error function is that gain values above and below the desired gain are treated differently because of the nonlinearity introduced by the absolute value function. For example values of 0.5 and 1.5 for $G_T(f)/G_{Topt}(f)$ both give a MRD value of 0.5 despite the fact that the first case represents a gain error of 3 dB and the second only 1.76 dB. This nonuniformity means that errors below the desired gain will be better optimised than errors above the desired gain although the difference will decrease when the error is small.

The bias inherent in MRD errors is avoided by using the gain error in decibels. This error is calculated using

$$dB_{err} = \left| 10 \log \left(\frac{G_T(f)}{G_{Topt}(f)} \right) \right| \tag{3.26}$$

where dB_{err} is the gain error in decibels.

Conversions between these error functions will now be derived so that results with different error functions can be compared. The case where $G_T(f)/G_{Topt}(f)$ is greater than one has to be considered separately from the case where it is less than one because of the absolute value functions in the definitions of MRD and gain error.

The derivation of decibel error in terms of MRD proceeds by solving (3.25) in terms of $G_T(f)/G_{T_{opt}}(f)$ and the substituting these into (3.26) to obtain the decibel error in terms of MRD:

$$dB_{err} = \begin{cases} 10\log(1+\text{MRD}) & \text{if } \frac{G_T(f)}{G_{Topt}(f)} \geq 1 \\ 10\log(1-\text{MRD}) & \text{if } \frac{G_T(f)}{G_{Topt}(f)} < 1 \end{cases} . \tag{3.27}$$

This result allows conversions from MRD to decibel error.

The conversion from decibel error to MRD also involves two cases. The equation for decibel error (3.26) is solved for $G_T(f)/G_{T_{opt}}(f)$ and the result is substituted into (3.25) to give

$$\text{MRD} = \begin{cases} 10^{\frac{dB_{err}}{10}} - 1 & \text{if } \frac{G_T(f)}{G_{Topt}(f)} \geq 1 \\ 1 - 10^{\frac{-dB_{err}}{10}} & \text{if } \frac{G_T(f)}{G_{Topt}(f)} < 1 \end{cases} . \tag{3.28}$$

This result allows conversions from decibel error to MRD.

VSWR is only defined for the case where a perfect match is attempted, so $G_{T_{opt}}(f)$ is always equal to one because a perfect match for a passive network corresponds to a gain of one. This means that $G_T(f)/G_{T_{opt}}(f)$ is always less than or equal to one because $G_T(f)$ cannot be greater than one. Thus, only one case needs to be considered in the conversions between VSWR, and MRD and decibel error.

The conversion from decibel error to VSWR involves converting the decibel error to a reflection coefficient using (3.23) and (3.26) to obtain

$$|\Gamma| = \sqrt{1 - 10^{\frac{-dB_{err}}{10}}} . \tag{3.29}$$

This value is then substituted into (3.20) to obtain VSWR in terms of decibel error.

The reflection coefficient is obtained in terms of MRD using (3.23) and (3.25) and this result is then substituted into (3.20) to obtain

$$\text{VSWR} = \frac{1+\sqrt{\text{MRD}}}{1-\sqrt{\text{MRD}}} . \tag{3.30}$$

This result allows conversions from MRD to VSWR.

Converting from VSWR to MRD and decibel error involves solving (3.20) in terms of $|\Gamma|$, using this value in (3.23) to calculate $G_T(f)$, and substituting the result into (3.25) and (3.26). The results after simplification are:

$$\text{MRD} = 1 - \left(\frac{\text{VSWR} - 1}{\text{VSWR} + 1}\right)^2 \tag{3.31}$$

and

$$dB_{err} = -10 \log \left[\frac{4\text{VSWR}}{(\text{VSWR} + 1)^2}\right] \tag{3.32}$$

where the minus sign in (3.32) is due to the fact that the logarithm is negative because its argument is less than one.

3.3.2 Impedance Calculation

This section will consider the calculation of the input impedance used to calculate the values of the error functions given in Section 3.3.1. The calculation of the input impedance, the effect of losses, lumped and distributed elements, and discontinuities are presented below.

The impedances required to calculate the values of the error functions described in Section 3.3.1 are calculated starting at the load impedance. Each element is then used to modify the impedance until the input impedance of the system is obtained. This input impedance is then used to calculate the input reflection coefficient from which the fitness is calculated. The only major change to this approach occurs when discontinuities are used. In this case the effect of parallel elements cannot simply be included when they are encountered because the discontinuities affect the parallel elements' contributions to the circuit, and the discontinuity parameters depend on all circuit elements forming the discontinuity. The solution used in this dissertation is to only modify the impedance when a series element is encountered. The effects of any parallel elements and discontinuities are then simultaneously accounted for.

All elements used are assumed to be lossless. The main reason for this approach is that it allows simplified calculation of the gain of the matching network as discussed in Section 3.3.1. The error encountered by using the assumption that all elements are lossless is usually very small, and is adequate for a synthesis technique such as the one presented here.

The lumped elements used in this dissertation are simple capacitors and inductors which can be connected either in series or in parallel. The models used do not include losses or resonant effects which arise due to the parasitic elements that are present in all components [2]. The main reason for this is that the magnitude of the parasitic effects change with component value and type. Accounting for parasitics is however an extremely important process and would have to be done for any practical design.

The distributed elements in this dissertation are either perfect transmission lines or microstrip transmission lines. Series lines, and open and short circuited stubs are allowed. In the case of perfect transmission lines, the same characteristic impedance is used at all frequencies and dispersion is not present, allowing the characteristic impedance of the line to be specified directly. Microstrip lines are not perfect TEM transmission lines leading to dispersion which causes the characteristic impedance to vary with frequency. This means that the characteristic impedance of a microstrip line cannot be uniquely specified and the line widths have be specified instead.

One of the most important properties of the algorithm developed during this dissertation is the inclusion of dispersion and discontinuity effects during synthesis. This allows these effects to be used by the algorithm to obtain better results than algorithms that attempt to minimise these effects or ignore them altogether. The discontinuity models used can thus have a large effect on the quality of the results obtained. The discontinuities considered are open-ends, via holes to ground, width steps, T-junctions, and crosses.

3.3.3 Pareto-Like Fitness

One of the major advantages of maintaining a population of solutions is that a number of solutions to a problem can be obtained. The problem is to find a way of identifying multiple solutions that are of high quality. Pareto optimality, which was considered in Section 2.1.1.3, is one solution to this problem because it allows a number of objectives to be optimised simultaneously. This dissertation uses a simplified implementation of Pareto optimality to rank solutions.

The two objectives used for the Pareto-like optimality criterion are the fitness (error) and the length of an individual. The error functions used in this dissertation are considered in Section 3.3.1, and the length of an individual is the number of circuit elements comprising that individual. The use of the length of an individual as one of the objectives is intuitive because a good five element solution can usually be created by adding one element to a good four element solution and optimising the result. Using a multi-objective fitness function thus means that a number of solutions can be compared, and the performance of the algorithm should improve.

Pareto optimality works on the principle of dominated and non-dominated solutions. The fitness is then assigned by giving each layer of non-dominated individuals a new fitness as described in Section 2.1.1.3. The implementation of this process can be complicated, so a simpler Pareto-like ranking system was used in this dissertation.

The population is first sorted according to the two objective functions' values so that the final order has the lengths from shortest to longest, and each length further is sorted so that the error values vary from low to high. The fitness is then assigned by using a fitness of zero for the first individual of each length, and increasing the fitness by one for each subsequent individual of the same length. This has the effect of ranking each length independently to ensure that comparisons between individuals of different lengths are fair.

The main difference between this approach and the Pareto approach is that solutions are only considered to be dominated by other solutions of the same length. The largest differences between the fitnesses assigned by this approach and the Pareto approach will be seen near the beginning of a run, when some individuals will have fitnesses that differ from the pure Pareto case because their Pareto ranking is affected by individuals of other lengths. Near the end of a run these fitness differences should be negligible because longer individuals will give smaller errors than shorter individuals meaning that the Pareto ranking of an individual is almost always only affected by individuals of the same length.

This approach only works because there are a small number of lengths, and each length generally gives better results than shorter lengths. This approach will not work where two continuous variables are used.

3.4 Local Optimiser

This section will consider the implementation of the local optimiser used in this dissertation. The decision as to which local optimisation algorithm to use will be considered first because this determines whether gradient calculations and line searches are necessary or not. Then gradient calculation is considered followed by a description of the line search algorithm used.

3.4.1 Algorithm

The choice of local optimisation algorithm is very important to this dissertation because it determines whether gradient calculations and line searches are necessary, and it can affect the performance of the algorithm. This section will consider the choice of local optimisation algorithm.

The first major consideration is that the objective function used in this dissertation is a minimax function. This leads to gradient discontinuities and limits the usefulness of methods that assume that the objective function is quadratic. Another consideration is that the genetic algorithm will probably find a point near an optimum, so the local optimiser does not have to be exceptionally good. The calculation of the gradient is difficult because of the presence of discontinuity models which do not have simple gradient functions, so an algorithm that does not require many gradient calculations would be preferable.

Simplex methods have the disadvantage that they are known to be slow for large problems [12]. The direction set and conjugate gradient methods assume the function is quadratic which is a poor approximation to a minimax problem. The quasi-Newton methods also use a quadratic approximation, but have the additional disadvantage that the calculations can involve large matrices. The leap-frog algorithm has the limitation that a large number of gradient calculations are required.

Eventually the steepest descent algorithm was chosen. Steepest descent is known to be a poor local optimiser, but the problems usually only arise when the current solution is far from an optimum. As mentioned above, the genetic algorithm should provide a solution that is close to an optimum, so these problems should not be significant. Steepest descent has the additional advantage that it is very simple to implement, but it has the drawback that gradient information is required.

3.4.2 Gradient Calculation

Gradient calculation is required for the implementation of most local optimisation algorithms including the steepest descent algorithm used here. The accurate calculation of the gradient is essential to obtain good results and this section describes the gradient calculation used in this dissertation.

The main problem encountered with gradient calculations in this dissertation is the presence of discontinuities. The models for these discontinuities are complex precluding the calculation of closed form gradient functions. This meant that finite differences had to be used.

Finite differences were used to calculate the gradient as described in Section 2.1.3.3. The values of distributed elements parameters were changed by one to obtain the maximum accuracy. An adjustment of 10^{-6} of the maximum value was used for lumped elements because the extremely high precision of the representation could lead to problems with rounding errors if the step size used is too small.

3.4.3 Line Search

The implementation of the line search for this dissertation is considered in this section. The Fibonacci line search algorithm was used. The motivations for this choice, and the modifications made are considered in this section.

As mentioned previously, a polynomial approximation is a poor fit to minimax functions, and this means that interpolation methods considered in Section 2.1.2.3 will not produce good results. This narrows the choice to the Fibonacci and golden ratio line search algorithms. The only advantage of the golden ratio line search is that the number of steps does not have to be set in advance. The number of line search steps was implemented as a constant in this algorithm to minimise the number of optimisation steps wasted on poor solutions. Good solutions will tend to survive to the next generation of a genetic algorithm allowing them to be optimised further, while poor solutions will die off, limiting the computations wasted on that solution. Thus the Fibonacci line search was chosen because it gives a smaller error than the golden ratio search for a given number of steps.

The next problem is bracketing a minimum. The quadratic interpolation method considered in Section 2.1.2.1 will not produce good results because a minimax function is used. A heuristic technique was thus implemented. Three points are required to determine whether an optimum has been bracketed, and these three points should preferably be chosen so that they form three of the four points used by the Fibonacci search.

The initial step size in the direction of the search in a constant. A third point is now

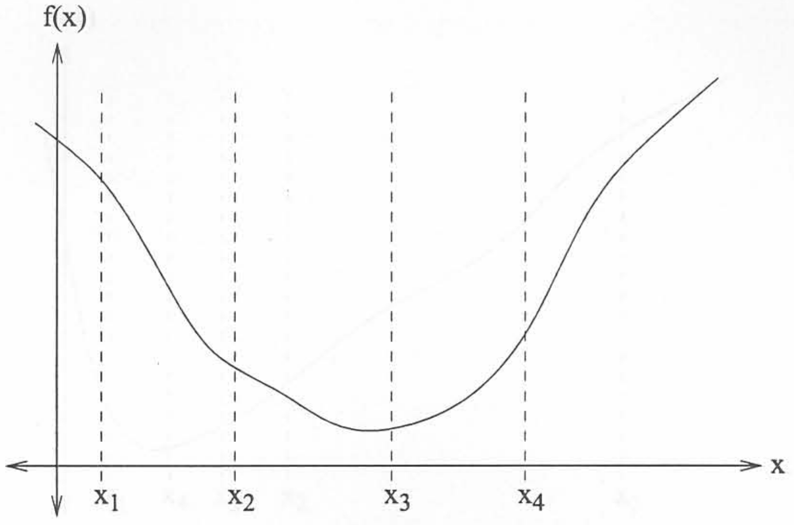


Figure 3.2: Line search range extension.

required between the starting and ending points of the range. This point is generated by using the point from the Fibonacci search that is closest to the starting point of the current range, giving points x_1, x_2, and x_3 in Figure 3.2. The range must be extended if the middle point has an objective function value that is worse than the objective function value at the end point of the range. The range extension is accomplished by using the previous range middle point as the new starting point and the previous end point as the new middle point, giving points x_2 and x_3 in Figure 3.2. A new end point must now be determined to ascertain whether an optimum is now bracketed (point x_4 in Figure 3.2). The maximum number of times this process may be repeated is set as a constant. Each range extension step only requires one calculation and the last three points generated from three of the four points required by the Fibonacci line search. In other words, point x_5 would have to be added to points x_2, x_3, and x_4 in Figure 3.2 to start a Fibonacci line search.

It is possible that the procedure described above can bracket more than one optimum, but the Fibonacci search only works in the case where a single optimum is bracketed. This problem is overcome by checking that the two middle points of the Fibonacci line search are better than the endpoints. If this is not true the line search is terminated and the best point found up to that point is returned. Note that only one of the middle points has to be

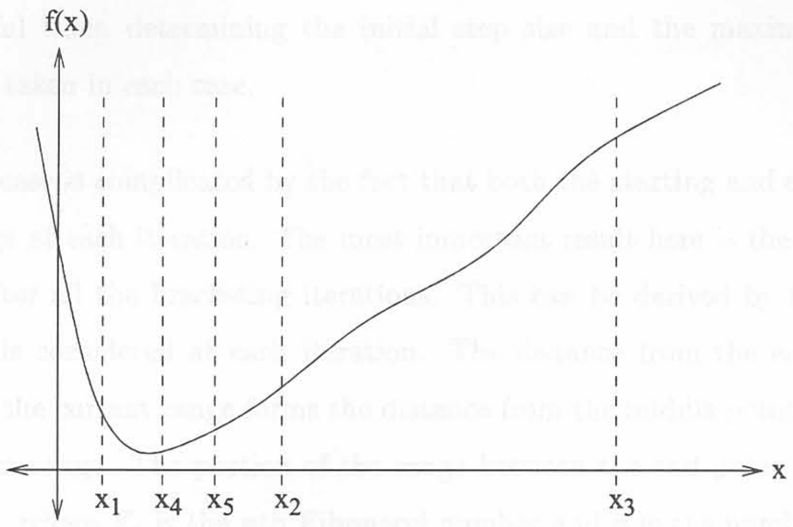


Figure 3.3: Line search range reduction.

checked at each step because the other middle point is carried over from the previous step.

A similar procedure was implemented to reduce the range before the commencement of the Fibonacci line search because the genetic algorithm and repeated optimisations should ensure that the current point is close to the local optimum. The range reduction search starts in the same way as the bracketing search with a step being taken in the direction of the line search and a middle point being determined, giving points x_1, x_2, and x_3 in Figure 3.2. If the middle point is worse than the range starting point then the current middle point is taken as the new range end point and the procedure is repeated, giving points x_1, x_2, and x_4 in Figure 3.2. As with the bracketing search, each iteration only requires one calculation and the last three points generated form three of the four points used by the Fibonacci search. In other words, point x_5 would have to be added to points x_1, x_2, and x_4 in Figure 3.2 to start a Fibonacci line search. The maximum number of times this procedure may be repeated is limited. This range reduction algorithm decreases the range faster than a Fibonacci line search when the optimum is very close to the starting point.

The size of the final ranges and the steps taken by the bracketing and range reduction searches is useful when determining the initial step size and the maximum number of iterations to be taken in each case.

The bracketing case is complicated by the fact that both the starting and ending points of the range change at each iteration. The most important result here is the maximum step that is taken after all the bracketing iterations. This can be derived by first calculating the range that is considered at each iteration. The distance from the end point to the middle point of the current range forms the distance from the middle point to the starting point of the next range. The portion of the range between the end point and the middle point is F_{p-1}/F_p where F_n is the nth Fibonacci number and p is the number of iterations in the Fibonacci search [12]. The ratio of the of the total range to the distance from the middle point to the starting point is F_p/F_{p-2}. These results can then be used to find the relationship between the current range and the previous range:

$$I_{i+1} = \frac{F_p}{F_{p-2}} \frac{F_{p-1}}{F_p} I_i \tag{3.33}$$

$$= \frac{F_{p-1}}{F_{p-2}} I_i \tag{3.34}$$

where I_i denotes the range at iteration i. The range at any iteration can be calculated by applying (3.35) recursively:

$$I_i = \left(\frac{F_{p-1}}{F_{p-2}}\right)^i I_0 \ . \tag{3.35}$$

The total step size can now be calculated using this result and ignoring overlaps between iterations. The portion of each range that is not an overlap is F_{p-1}/F_p. The total step size is now given by

$$I_i = I_0 + I_0 \frac{F_{p-1}}{F_p} \sum_{i=1}^{n} \left(\frac{F_{p-1}}{F_{p-2}}\right)^i . \tag{3.36}$$

The first term is the size of the initial step that is always taken. The second term is the sum of the step sizes at each iteration which are calculated from the portion of the range that is not an overlap at each iteration.

The range reduction case is a simple application of the formulae that are used by the Fibonacci line search. The portion of the total range that is between the starting and middle points of the range for the first iteration of a Fibonacci search is given by F_{p-2}/F_p

Table 3.5: Default local optimiser parameters.

Parameter	Value
Initial step size	$\frac{1}{8000}$
Local optimiser steps	2
Line search steps	3
Range extension steps (maximum)	5
Range reduction steps (maximum)	5

where F_n is the nth Fibonacci number and p is the number of iterations in the Fibonacci search [12]. At each iteration of the range reduction algorithm the previous range will be reduced by this factor. The final range as a fraction of the initial range is thus given by

$$I_m = \left(\frac{F_{p-2}}{F_p}\right)^m I_0 \tag{3.37}$$

where I_m is the range after m range reduction iterations. This is seen to be much faster than the Fibonacci search in this case because the Fibonacci search reduces the range by a factor of F_p after p iterations.

The parameters used in the final algorithm are given in Table 3.5. The initial step size was chosen to give a change of approximately one in the smallest range used in the representation (transmission line length). The number of line search steps was chosen to be the smallest allowable value for a Fibonacci line search because the genetic algorithm is the primary search algorithm and accurate line searches are thus not required. The number of range reduction steps were chosen to give a minimum step size of 3.91×10^{-6} of the maximum value used in the representation, which corresponds to an 18 bit accuracy. This accuracy might appear to be too low, but it corresponds to four significant digits when the component range is 10^{-15} to 10^{-6}. The total range after range extension with five range extension steps is 2.44×10^{-3} of the maximum value used in the representation.

The last important consideration is that a line search can generate points that violate the constraints. In this case the constraints specify the maximum and minimum values of the variables and are thus inequality constraints. If a line search generates a point that violates constraints, all the variables whose constraints are violated are set to either their maximum or minimum values as required. The main drawback of this approach is that it changes the line search direction. A way of avoiding this problem is to move backwards along the search direction until all constraints are satisfied. However this will mean that points on a constraint boundary that have a gradient that points into the infeasible region will not be optimised by the line search. This problem is avoided by the approach described above.

3.5 Summary

The implementation of the impedance matching algorithm developed during the course of this work was given in this chapter.

Firstly, an overview of the approach followed was presented. The reasons for using a hybrid genetic algorithm, and a flow chart of the algorithm were given.

The genetic algorithm implemented was then considered in detail with special attention being paid to the representation and operators used. The representation used is unique because it allows both binary and floating-point genetic operators to be used, and equations for calculating the decimal precision of the representation were derived.

The error functions, impedance calculation, and Pareto-like optimality were reviewed to show how the fitness of an individual was calculated. Equations for the conversions between VSWR, MRD, and decibel errors were derived, and the default algorithm parameters were given. The calculation of the input impedance of a network includes microstrip dispersion and discontinuity effects. A new type of Pareto-like optimality was used to allow multiple solutions with different lengths to be synthesised simultaneously.

The choice of local optimiser, and the gradient calculation and line search algorithms used were discussed. Steepest descent was used because it is very simple and robust, and the combination with a genetic algorithm should overcome its limitations. Gradients were calculated using finite differences because the complex discontinuity models used mean that the derivation of analytic expressions would be impractical. A Fibonacci line search was used as the basis for the line search algorithm, with extensions being made to allow a minimum to be bracketed.

The performance of the algorithm presented above will be considered in Chapter 4.

Chapter 4

Algorithm Testing

A large number of tests were conducted on this algorithm. These tests were conducted to ensure that the fitness calculation is correct, determine good values for the parameters, compare various options, and to compare this algorithm with published results.

Ten test problems were used to evaluate the algorithm and are presented in Section 4.1. The tests performed include accuracy verification in Section 4.2, parameter value tests in Section 4.3, and run time evaluation in Section 4.4. The results obtained are summarised in Section 4.5 and comparisons to published results are given in Section 4.5.1. The implications of the results obtained here will be considered in Chapter 5.

4.1 Test Problems

The problems used to test this algorithm are presented in this section. Ten problems from a number of different sources and circumstances were used to test the algorithm as thoroughly as possible. A number of active devices, manufacturers, and types of problem were considered to provide as wide a variety of test cases as possible.

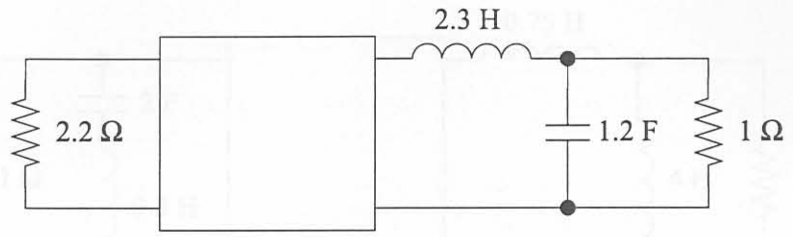


Figure 4.1: Test Problem 2.

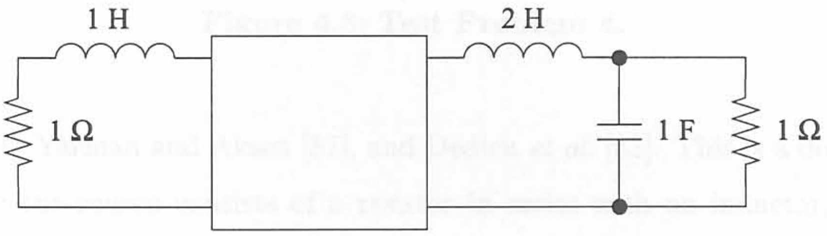


Figure 4.2: Test Problem 3.

Problem 1 is taken from Abrie [2] and involves matching a source of resistance of 25 Ω to a load resistance of 100 Ω from 2 to 6 GHz. A frequency spacing of 0.25 GHz was used for this problem.

Problem 2 has been used extensively in the literature to test impedance matching algorithms. Papers that use this problem include Fano [57, 58], Carlin [4], Carlin and Amstutz [5], and Dedieu *et al.* [82]. This is a single matching problem where a resistive source is matched to a network consisting of the parallel combination of a capacitor and a resistor in series with an inductor as shown in Figure 4.1. The value of the source resistor is not specified, but Carlin [4] found that a value of 2.2 Ω gives the best results, and experiments with this algorithm indicate that this is true in this case as well. The matching network must have a low-pass form with a maximum frequency of 1 rad/s. The circuit was scaled in frequency so that the maximum frequency is 1 GHz, and a frequency spacing of 0.1 GHz was used. The impedances were scaled by 50 Ω for the distributed, mixed, and microstrip tests to ensure sensible results are obtained.

Problem 3 is also taken from the literature and is used by Carlin and Yarman [8], Yarman

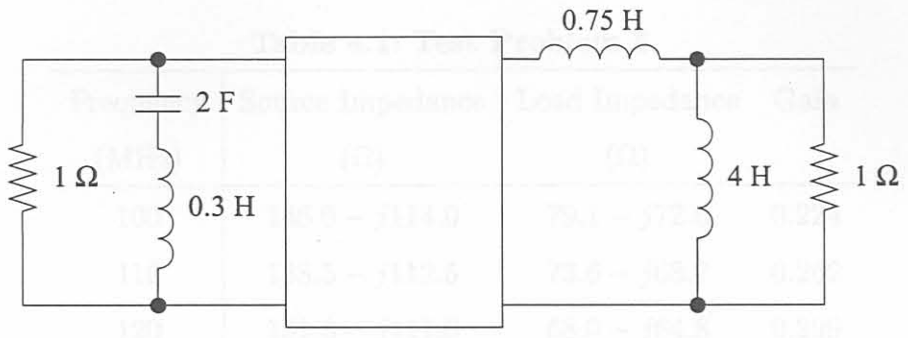


Figure 4.3: Test Problem 4.

and Fettweis [9], Yarman and Aksen [87], and Dedieu *et al.* [82]. This is a double matching problem where the source consists of a resistor in series with an inductor, and the load consists of the parallel combination of a capacitor and a resistor in series with an inductor as shown in Figure 4.2. The network must match the load to the source from 0 to 1 rad/s, so this is a low-pass problem. The problem was again scaled in frequency so that the frequency range is from 0 to 1 GHz, and a frequency spacing of 0.1 GHz was used. The impedances were scaled by 50 Ω for the distributed, mixed, and microstrip tests to ensure sensible results are obtained.

Problem 4 appears in the papers by Carlin and Yarman [8], Yarman and Fettweis [9], and Dedieu *et al.* [82]. The problem involves matching a source consisting of a resistor in parallel with the series combination of a capacitor and an inductor to a load consisting of the parallel combination of an inductor and a resistor is series with an inductor as shown in Figure 4.3. The frequency range for this problem extends from 0.3 to 1 rad/s, but this range was scaled to be from 0.3 to 1 GHz with frequency steps of 0.1 GHz being used. The impedances were scaled by 50 Ω for the distributed, mixed, and microstrip tests to ensure sensible results are obtained.

The number of points used for Problems 1 to 4 was chosen to allow fast simulations while still producing good results. Doubling the frequency spacing for Problem 1 produces slightly better results, while halving the frequency spacing produces better results for Problems 2 and 3, and essentially the same results for Problem 4. This means that the frequency

Table 4.1: Test Problem 5.

Frequency	Source Impedance	Load Impedance	Gain
(MHz)	(Ω)	(Ω)	
100	$146.0 - j114.0$	$79.1 - j72.6$	0.224
110	$138.5 - j112.5$	$73.6 - j68.7$	0.262
120	$131.0 - j111.0$	$68.0 - j64.8$	0.299
140	$137.0 - j103.0$	$63.2 - j56.8$	0.400
160	$144.0 - j88.0$	$59.6 - j47.9$	0.559
180	$140.0 - j88.0$	$57.5 - j47.3$	0.709
190	$136.5 - j92.0$	$55.0 - j41.9$	0.764
200	$133.9 - j96.0$	$53.5 - j40.4$	0.818

Table 4.2: Test Problem 6.

Frequency	Source Impedance	Load Impedance	Gain
(GHz)	(Ω)	(Ω)	
2	$75.08 + j0.84$	$83.16 - j135.9$	0.7462
3	$81.22 + j2.98$	$53.02 - j102.9$	0.8874
4	$81.94 - j1.52$	$35.56 - j77.55$	0.8802
5	$85.15 - j1.40$	$39.93 - j68.64$	1.0000
6	$81.44 - j1.19$	$22.69 - j46.11$	0.8605

spacings used above are conservative in the sense that different frequency spacings typically lead to better results.

Problem 5 is a double matching problem taken from Abrie [2]. Impedances and transducer gains are specified at a number of frequency points from 100 to 200 MHz as shown in Table 4.1.

Problem 6 considers an interstage match between two transistors and is also taken from Abrie [2]. The problem is part of the design of a two-stage amplifier. Impedances and

Table 4.3: Test Problem 7.

Frequency (GHz)	Impedance (Ω)	Gain
9.5	$1.932 - j10.43$	0.07468
10.0	$5.860 - j7.550$	0.2468
10.5	$7.200 - j4.424$	0.3334
11.0	$8.199 - j1.178$	0.4288
11.5	$9.976 + j1.985$	0.6111
12.0	$11.03 + j4.857$	0.8252
12.5	$10.02 + j7.515$	0.7809
13.0	$9.265 + j9.974$	0.7816
13.5	$8.671 + j12.47$	0.7978
14.0	$8.215 + j15.27$	0.8397
14.5	$6.952 + j18.36$	0.7546
15.0	$6.906 + j21.62$	0.8803
15.5	$6.281 + j25.21$	0.9430

transducer gains are specified at a number of frequency points from 2 to 6 GHz as shown in Table 4.2.

Problem 7 involves mismatching the output of a transistor to level the gain. The transistor chosen was the CFY35 GaAs FET from Infineon biased at 2.5 V and 10 mA. The gain was levelled at 9 dB from 9.5 to 15.5 GHz in steps of 0.5 GHz. The problem was converted to a passive matching problem using the equations derived by Abrie [2]. The impedances and transducer gains required by the transistor are given in Table 4.3. The transistor impedances in Table 4.3 were used as the load impedance, and the source impedance is 50 Ω at all frequencies.

Problem 8 requires the input of a transistor to be matched so as to obtain a specified noise figure. The device used is an Agilent ATF-36163 PHEMT biased at 2 V and 10 mA. The

Table 4.4: Test Problem 8.

Frequecy (GHz)	Impedance (Ω)	Gain
5	$32.9 + j60.0$	0.321
6	$27.6 + j43.5$	0.335
7	$26.0 + j32.7$	0.355
8	$27.9 + j22.7$	0.371
9	$28.3 + j14.8$	0.335
10	$31.6 + j4.0$	0.319
11	$37.1 - j7.5$	0.304
12	$46.0 - j17.1$	0.321
13	$64.5 - j24.0$	0.402
14	$97.5 - j20.3$	0.515
15	$136.9 + j14.4$	0.609
16	$138.5 + j78.8$	0.704
17	$87.6 + j91.9$	0.829
18	$48.1 + j86.1$	0.962

noise figure was levelled at 2 dB from 5 to 8 GHz in steps of 1 GHz. This would not necessarily be a good approach to designing a real amplifier because the problem should rather be set up to ensure that the noise figure is less than a specified value. However, the objective here is to set up a difficult test problem and designing a matching network for a specified gain is more complex than designing for a minimum gain. The problem was converted to a passive matching problem using the equations derived by Abrie [2]. The impedances and transducer gains required by the transistor are given in Table 4.4. The transistor impedances in Table 4.4 were used as the load impedance, and the source impedance is 50 Ω at all frequencies.

Problem 9 considers matching the output of a transistor for maximum output power. The active device is an Ericsson PTF10112 LDMOS biased at 28 V and 580 mA. The optimum

Table 4.5: Test Problem 9.

Frequency (GHz)	Impedance (Ω)
1.75	$1.48 - j0.25$
1.80	$1.56 + j0.20$
1.85	$1.66 + j0.50$
1.90	$1.32 + j0.80$
1.95	$1.16 + j0.60$
2.00	$1.10 + j0.45$
2.05	$1.18 + j0.30$

Table 4.6: Test Problem 10.

Frequency (GHz)	Impedance (Ω)
1.775	$34.27 + j36.07$
1.780	$41.33 + j33.14$
1.785	$48.17 + j27.34$
1.790	$53.12 + j18.53$
1.795	$54.45 + j7.97$
1.800	$51.68 - j2.07$
1.805	$46.01 - j9.68$
1.810	$39.24 - j14.31$
1.815	$32.73 - j16.47$
1.820	$27.09 - j16.96$
1.825	$22.47 - j16.47$

power match is required by the transistor given in the datasheet and is repeated in Table 4.5. The transistor impedances in Table 4.5 were used as the load impedance, with a source impedance of 50 Ω being used at all frequencies.

Problem 10 requires an antenna to be matched to 50 Ω. The antenna is a probe-fed patch antenna on a GIL MC3D substrate, and the data were supplied by Mr David de Haaij. Mr de Haaij is working towards a masters degree at the University of Pretoria and is considering patch antenna impedance matching. The antenna is designed for a 50 Ω system and a centre frequency of 1.8 GHz. Its dimensions are a length of 43.3 mm, a width of 47 mm, and a feed inset of 11.6 mm. The original data supplied was from 1.6 to 2 GHz in steps of 1 MHz. Using this full range would be impractical because the wide bandwidth would lead to poor results, and the algorithm would be slow because of the large number of frequency points. The data was thus reduced to be from 1.775 GHz to 1.825 GHz (a 2.78% bandwidth with a centre frequency of 1.8 GHz) in steps of 5 MHz as shown in Table 4.6. The antenna impedances in Table 4.6 were used as the load impedance, and the source impedance is 50 Ω at all frequencies. This is an extremely useful problem because an antenna is a resonant structure unlike the devices in the problems considered above.

This selection of test problems provides a large number of different cases which can be used to evaluate the performance of this algorithm. The problems include low-pass and band-pass networks, gain and noise matches, a GaAs FET and a PHEMT, single and double matching problems, resonant structures, purely real loads and sources, and established and new problems.

4.2 Accuracy Verification

The first important stage of testing involves confirming that the error function calculations are correct. This was done using Hewlett-Packard's Touchstone microwave circuit simulation software. Only single elements are considered here because complete circuits will be accurate if all the element models are accurate.

Version 2.000.104.019 of EEsof was used to verify the accuracy of the error calculations. This version of EEsof was released in 1994 and is thus a little old. This is not a great problem because the major changes that have been made are to the user interface and this is obviously not relevant here. While the EESof models are not perfect they do provide a basis for evaluating the accuracy of the models implemented here. Using comparisons to measurements or full-wave electromagnetic simulations would be desirable, but were considered to be beyond the scope of this work. The agreement between EESof and the models implemented here should be good because the EESof models are, in most cases, based on the published models used here.

4.2.1 Single Elements

The first round of tests was conducted by considering only one type of element to determine the accuracy of each element. The input impedance of a 50 Ω system was calculated and then compared to the results obtained using EEsof.

The error was calculated from the magnitude of a type of reflection coefficient defined by

$$E = \left| \frac{Z - Z_{EEsof}}{Z + Z_{EEsof}} \right| \tag{4.1}$$

where E is the error, Z is the input impedance calculated, and Z_{EEsof} is the input impedance calculated by EEsof. The difference between the definition of reflection coefficient given in (3.21) and (4.1) is that conjugates are not used here. This change is necessary because the objective is to find the error between two impedances rather than a conjugate match.

Rounding errors caused some difficulties. The highest error for a parallel inductor was 0.227 and occurred with impedances of $2.2 \times 10^{-14} + j10^{-6}$ for EEsof and $7.9 \times 10^{-15} + j6.3 \times 10^{-7}$ for the current algorithm. This is obviously not a true error because the impedances are very small so all points where the real and imaginary parts of both impedances were less than 10^{-6} were ignored.

The first tests were run for purely lumped components. The EEsof models for an ideal inductor (IND) and capacitor (CAP) were used. The component values were logarithmically swept from 10^{-15} to 10^{-3}. The highest error was less than 10^{-8}, so the lumped component models used in this algorithm are exact to within rounding errors.

Perfect transmission lines were tested next. The EEsof models for a series transmission line (TLIN), a parallel shorted stub (TLSC), and a parallel open stub (TLOC) were used here. These models require the length in degrees at a specified frequency and the characteristic impedance of the line. The minimum and maximum values of the characteristic impedance were 1 Ω and 400 Ω respectively. These values are the rounded minimum and maximum impedances obtained when the effective dielectric constant of a microstrip line can vary from 1 to 10, and the width of the microstrip line can vary from $0.01h$ to $100h$, where h is the substrate height. The length was varied from 0.1λ to 1λ, where λ is the wavelength of the transmission line. The highest error obtained was less than or equal to 10^{-4} in all cases, so the transmission line models used in this algorithm are exact to within rounding errors.

The next step was to test the system using microstrip lines using the full model which includes dispersion effects. The EEsof models for a series microstrip line (MLIN), a parallel shorted stub (MLSC), and a parallel open stub (MLOC) all use the full microstrip model including dispersion. These models require the substrate to be specified, and the length and width of the lines to be given. Only the relative dielectric constant and height of the substrate were specified with all other values being set to zero. The substrate used has a height of 1 mm and a relative dielectric constant of 1, 2, 5, and 10. This substrate was used for all the microstrip and discontinuity tests in this section. The minimum and maximum line widths were $0.01h$ and $100h$ respectively, and the line length was varied from 1 mm to 10 cm. The worst results for a series microstrip line are shown in Table 4.7, and are typical of all three cases. The results are very good and show that the model used here is accurate when the substrate relative dielectric constant is low and the frequency is low.

The open-end model was tested next using the EEsof microstrip line model that includes the end effect (MLEF) with the length of the lines set to zero to isolate the end effect. The average and worst results are shown in Table 4.8 and the agreement is good even at high

Table 4.7: **Microstrip Line Worst Results.**

Frequency (GHz)	$\varepsilon_r = 1$	$\varepsilon_r = 2$	$\varepsilon_r = 5$	$\varepsilon_r = 10$
0.1	0	1.33×10^{-4}	3.41×10^{-4}	5.23×10^{-4}
0.2	0	2.90×10^{-4}	7.67×10^{-4}	1.52×10^{-3}
0.3	0	5.03×10^{-4}	1.32×10^{-3}	2.40×10^{-3}
0.4	0	1.04×10^{-3}	2.10×10^{-3}	4.08×10^{-3}
0.5	1.15×10^{-6}	1.22×10^{-3}	3.85×10^{-3}	4.99×10^{-3}
0.6	2.01×10^{-6}	1.35×10^{-3}	3.31×10^{-3}	6.11×10^{-3}
0.7	4.05×10^{-7}	1.52×10^{-3}	4.06×10^{-3}	7.43×10^{-3}
0.8	4.18×10^{-7}	2.06×10^{-3}	5.22×10^{-3}	8.68×10^{-3}
0.9	1.67×10^{-6}	2.18×10^{-3}	5.20×10^{-3}	9.37×10^{-3}
1.0	1.15×10^{-6}	2.63×10^{-3}	6.27×10^{-3}	1.10×10^{-2}
2.0	1.02×10^{-5}	5.41×10^{-3}	1.11×10^{-2}	1.62×10^{-2}
3.0	1.00×10^{-5}	7.82×10^{-3}	1.41×10^{-2}	3.03×10^{-2}
4.0	1.50×10^{-5}	9.47×10^{-3}	2.17×10^{-2}	2.05×10^{-2}
5.0	2.11×10^{-5}	1.34×10^{-2}	2.44×10^{-2}	3.78×10^{-2}
6.0	1.01×10^{-5}	1.40×10^{-2}	3.41×10^{-2}	3.91×10^{-2}
7.0	4.40×10^{-6}	1.53×10^{-2}	4.35×10^{-2}	4.32×10^{-2}
8.0	1.22×10^{-5}	2.16×10^{-2}	5.24×10^{-2}	4.91×10^{-2}
9.0	1.02×10^{-5}	1.58×10^{-2}	5.70×10^{-2}	7.31×10^{-2}
10.0	1.33×10^{-5}	2.41×10^{-2}	5.75×10^{-2}	9.13×10^{-2}

frequencies.

The model used for the inductance of a via hole was compared to the VIA2 model used by EEsof. The results were identical indicating that the model implemented here is the same as the model implemented by EEsof.

Microstrip width steps were compared to the EEsof microstrip step model (MSTEP). The width of each of the lines was varied from $0.1h$ to $10h$, and the width ratio between the

Table 4.8: Microstrip Open End Results.

Frequency	$\varepsilon_r = 1$		$\varepsilon_r = 2$		$\varepsilon_r = 5$		$\varepsilon_r = 10$	
(GHz)	Mean	Worst	Mean	Worst	Mean	Worst	Mean	Worst
0.1	$< 10^{-3}$	0.002	$< 10^{-3}$	0.001	$< 10^{-3}$	0.001	$< 10^{-3}$	0.001
0.2	0.001	0.004	$< 10^{-3}$	0.003	$< 10^{-3}$	0.002	$< 10^{-3}$	0.002
0.3	0.001	0.006	0.001	0.004	$< 10^{-3}$	0.003	$< 10^{-3}$	0.002
0.4	0.001	0.008	0.001	0.006	0.001	0.004	$< 10^{-3}$	0.002
0.5	0.001	0.010	0.001	0.007	0.001	0.004	$< 10^{-3}$	0.002
0.6	0.002	0.012	0.001	0.008	0.001	0.005	$< 10^{-3}$	0.001
0.7	0.002	0.014	0.001	0.009	0.001	0.005	$< 10^{-3}$	0.001
0.8	0.002	0.016	0.001	0.010	0.001	0.005	$< 10^{-3}$	0.001
0.9	0.002	0.018	0.002	0.011	0.001	0.005	$< 10^{-3}$	0.001
1.0	0.003	0.020	0.002	0.012	0.001	0.005	$< 10^{-3}$	0.002
2.0	0.005	0.035	0.003	0.018	0.001	0.003	0.002	0.009
3.0	0.007	0.046	0.004	0.020	0.001	0.002	0.003	0.014
4.0	0.009	0.052	0.004	0.021	0.001	0.004	0.004	0.019
5.0	0.010	0.056	0.005	0.021	0.001	0.007	0.005	0.023
6.0	0.012	0.059	0.005	0.020	0.002	0.010	0.007	0.026
7.0	0.013	0.061	0.005	0.020	0.003	0.012	0.008	0.029
8.0	0.014	0.063	0.005	0.020	0.003	0.014	0.010	0.033
9.0	0.015	0.064	0.005	0.019	0.004	0.016	0.011	0.036
10.0	0.015	0.065	0.005	0.019	0.005	0.019	0.013	0.038

lines was varied from a factor of 2 to a factor of 10. A number of options for the step discontinuity model were considered with the model presented in Section 2.4.5 giving the best results. The results are given in Table 4.9 where the data are presented as the ratio between the substrate height and the wavelength in the substrate when the error is 1% and 5%. The model is seen to perform best when the relative dielectric constant is low and the ratio of the two line thicknesses is small.

Table 4.9: Microstrip Width Step Results.

w_1/w_2	$\varepsilon_r = 1$		$\varepsilon_r = 2$		$\varepsilon_r = 5$		$\varepsilon_r = 10$	
	1%	5%	1%	5%	1%	5%	1%	5%
2	0.006	0.033	0.006	0.027	0.005	0.022	0.004	0.017
3	0.009	0.037	0.009	0.038	0.006	0.024	0.005	0.018
5	0.004	0.019	0.008	0.036	0.007	0.024	0.005	0.017
10	0.002	0.010	0.005	0.020	0.010	0.030	0.007	0.020

Table 4.10: Microstrip T-Junction Results.

w_1/w_2	$\varepsilon_r = 1$		$\varepsilon_r = 2$		$\varepsilon_r = 5$		$\varepsilon_r = 10$	
	1%	5%	1%	5%	1%	5%	1%	5%
1	0.011	0.017	0.011	0.017	0.010	0.016	0.010	0.016
2	0.006	0.025	0.006	0.024	0.005	0.021	0.004	0.018
3	0.002	0.008	0.002	0.009	0.002	0.008	0.001	0.006
5	0.002	0.009	0.001	0.008	0.001	0.007	0.001	0.006
10	0.001	0.005	0.001	0.005	0.002	0.005	0.001	0.004

The microstrip T-junction model was compared to the EEsof T-junction model (MTEE). The width of each of the lines was varied from $0.1h$ to $10h$, and the width ratio between the lines was varied from a factor of 1 to a factor of 10. The results are given in Table 4.10 where the data is presented as the ratio between the substrate height and the wavelength in the substrate when the error is 1% and 5%. The model is seen to perform best when the relative dielectric constant is low, and the ratio of the line thicknesses is small.

The microstrip cross model was compared to the EEsof cross model (MCROS). The width of each of the lines was varied from $0.4h$ to $2h$, and the width ratio between the lines was varied from a factor of 1 to a factor of 5. This is the maximum range of values over which the EEsof cross model is accurate. The results are given in Table 4.11 where the data is presented as the ratio between the substrate height and the wavelength in the substrate

Table 4.11: Microstrip Cross Results.

w_1/w_2	$\varepsilon_r = 1$		$\varepsilon_r = 2$		$\varepsilon_r = 5$		$\varepsilon_r = 10$	
	1%	5%	1%	5%	1%	5%	1%	5%
1	0.002	0.011	0.004	0.017	0.007	0.036	0.013	0.054
2	0.001	0.005	0.001	0.007	0.002	0.012	0.003	0.016
3	0.001	0.004	0.003	0.013	0.002	0.008	0.002	0.011
5	0.001	0.002	0.001	0.003	0.001	0.005	0.001	0.007

when the error is 1% and 5%. The model is seen to be poor, but this is expected because this is a modified T-junction model rather than a true cross model.

4.2.2 Circuits

The accuracy of complete circuits is considered here. The microstrip results obtained in the tests conducted for Section 4.5 include comparisons to the results obtained using EEsof.

The results in Tables A.4, A.8, A.12, A.16, A.20, A.24, A.28, A.32, A.36, and A.40 on pages 160 to 173 include both the results obtained with the current algorithm and the results obtained using EEsof. The first results are those obtained using the current algorithm and the second results are those obtained using EEsof. Crosses were not allowed because of the comparatively poor performance of the cross model.

The agreement is seen to be very good in most cases with the only exceptions being Problems 2 to 4. These problems all use an air substrate with a height of 3 mm. Problem 5 uses the same substrate, but does not have large errors, probably because it has a much lower maximum frequency than Problems 2 to 4.

4.3 Parameter Values

The algorithm developed here has a number of parameters that can be adjusted to improve the performance of the system. This section details the tests that were run to determine good values for the parameters.

The first important consideration is the component ranges that were used. Smaller component ranges will lead to faster convergence because a smaller range of values has to be searched. The component ranges were previously given in Table 3.3 on page 87.

Unless otherwise stated, each test was run 100 times to obtain statistics about the effect of each parameter. Individuals with one to six elements were considered. Each test was run for 40 generations and had a population size of 30000 for the pure genetic algorithm case and 3000 for the hybrid genetic algorithm case. The default genetic algorithm parameter values given in Table 3.1 on page 82, the default local optimiser values given in Table 3.5 on page 103, and the default component ranges given in Table 3.3 on page 87, were used.

The six individual length results are combined by normalising each length's parameters to its maximum value and adding the results. This result is then also normalised to its maximum value. This procedure is followed to ensure that large values, like those obtained with shorter individuals, do not dominate the results.

The minimum, median, maximum, mean, and standard deviation were determined for each test. The median is favoured over the mean as many of the results are heavily skewed towards good results because the algorithm usually obtains good results, and outliers can have a disproportionate effect on the mean. In most cases the difference between the mean and median is negligible, but there are some cases where the difference is significant. For example, running the hybrid algorithm on Problem 5 produced the cumulative distribution shown in Figure 4.4 for a two-element network. The median of the distribution is 0.194, but the mean is 0.237 despite the fact that 86% of the results are better than this value.

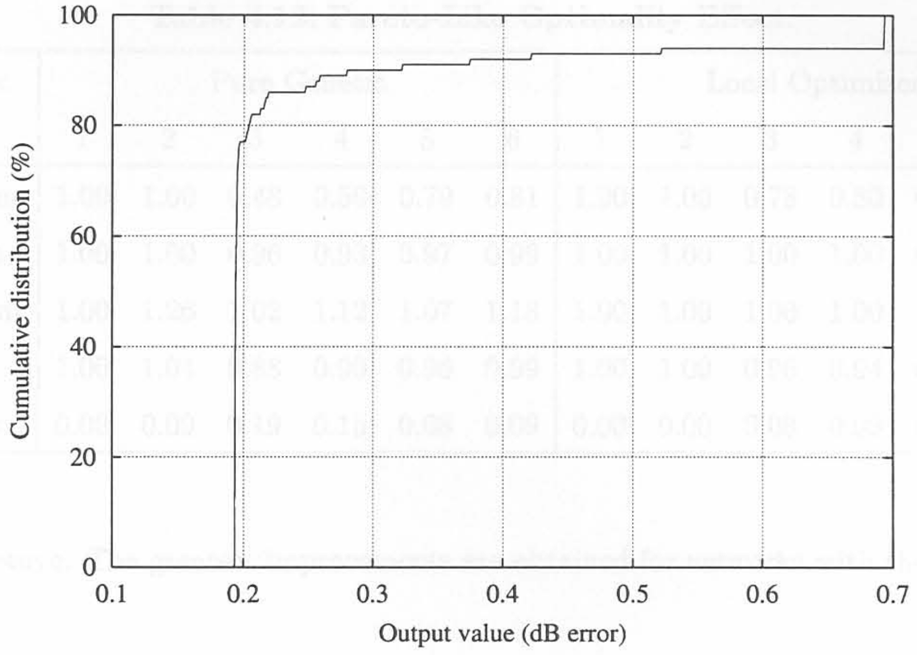


Figure 4.4: Example of a highly skewed result.

4.3.1 Pareto Effect

The effect of the Pareto-like optimality criterion discussed in Section 3.3.3 is considered here. The results with and without the Pareto-like optimality are compared to show how this approach affects the results.

The algorithm was firstly tested with the Pareto-like optimality enabled, and then with the Pareto-like optimality disabled. The algorithm had to be run once for each of the six lengths when Pareto-like optimality was disabled because each run only considers one length. Obviously this is not an entirely fair comparison because the algorithm is run six times when Pareto-like optimality is disabled, but only once when Pareto-like optimality is enabled. The results in Table 4.12 are determined from the ratio of the values obtained using Pareto-like optimality to the values that do not use Pareto-like optimality. The results are remarkable with the Pareto-like optimality actually improving the results in most cases despite requiring significantly fewer calculations. The addition of a local optimiser to the genetic algorithm decreases the effect of the Pareto-like optimality, but the results are

Table 4.12: Pareto-Like Optimality Effect.

Statistic	Pure Genetic						Local Optimiser					
	1	2	3	4	5	6	1	2	3	4	5	6
Minimum	1.00	1.00	0.48	0.59	0.79	0.81	1.00	1.00	0.78	0.80	0.89	0.72
Median	1.00	1.00	0.96	0.93	0.97	0.99	1.00	1.00	1.00	1.00	0.98	1.00
Maximum	1.00	1.26	1.02	1.12	1.07	1.18	1.00	1.00	1.00	1.00	1.02	1.07
Mean	1.00	1.04	0.88	0.90	0.96	0.99	1.00	1.00	0.96	0.94	0.97	0.97
Std Dev.	0.00	0.09	0.19	0.15	0.08	0.09	0.00	0.00	0.08	0.09	0.04	0.10

still impressive. The greatest improvements are obtained for networks with three and four elements.

4.3.2 Local Optimiser Effect

The effect of the local optimiser discussed in Section 3.4 is considered here. The results with and without the local optimiser are compared to show how it affects the results.

The algorithm was run with the default parameters in both cases. The only exception to this is that the pure genetic algorithm used a population of 50,000 and was run for 60 generations to ensure that similar numbers of fitness evaluations were used in both cases (approximately 2.6 million in the local optimiser case and approximately 2.3 million in the pure genetic algorithm case). The results in Table 4.13 are obtained from the ratio of the values using the local optimiser to the values that only use the genetic algorithm. The results obtained are better when the local optimiser is used and this in agreement with the results obtained by Renders and Flasse [34]. The effect of the local optimiser increases as the individuals become longer with significant differences being obtained for individuals with lengths of five and six elements.

Table 4.13: Local Optimiser Effect.

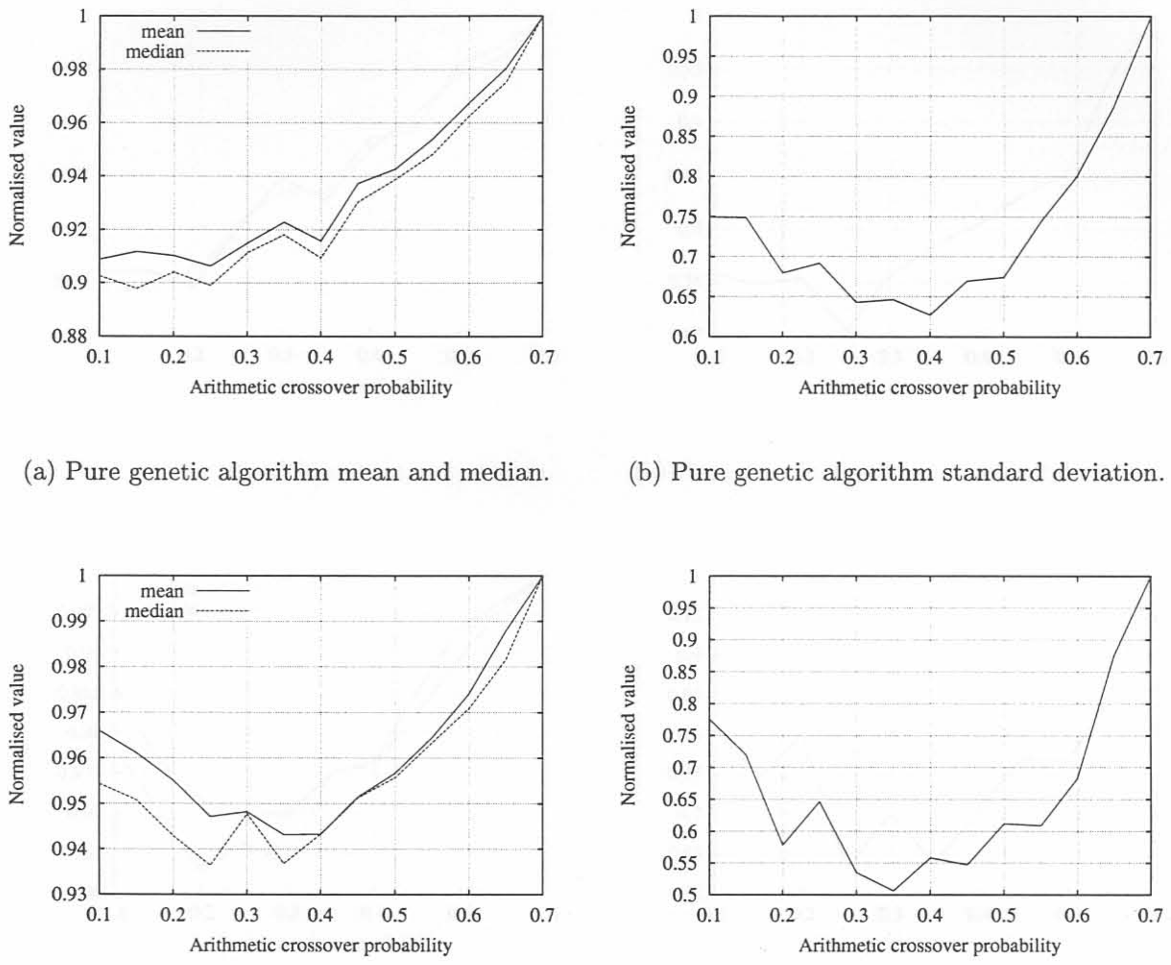
Statistic	Individual Length					
	1	2	3	4	5	6
Minimum	0.99	0.86	0.83	0.60	0.50	0.29
Median	1.00	0.99	0.99	0.96	0.88	0.80
Maximum	1.00	0.99	0.99	0.99	0.98	0.98
Mean	1.00	0.98	0.96	0.91	0.85	0.74
Std Dev.	0.00	0.04	0.05	0.12	0.14	0.23

4.3.3 Genetic Algorithm Operators

This section will consider the effect of changing the various genetic algorithm parameters. Only lumped elements were used for this round of testing because the results will be similar in the distributed and mixed cases, and lumped element tests are much faster. The genetic operators considered are arithmetic crossover, binary crossover, boundary mutation, uniform mutation, non-uniform mutation, and binary mutation.

The default values for the genetic operator probabilities were determined during the development of the algorithm and are given in Table 3.1. The probabilities were then varied round their default values to determine the effect of such variations. Both a pure genetic algorithm and a hybrid genetic algorithm with an integrated local optimiser were used for these tests, allowing the effect of the local optimiser to be observed.

The arithmetic and binary crossover results are given in Figures 4.5 and 4.6 respectively. The arithmetic crossover probability is varied to maximum value of 0.7 because the probability of crossover (arithmetic and binary) is one at this value. The maximum probability used for binary crossover is 0.6 for the same reason. The average results show that the error initially decreases to a minimum and then increases to a maximum as the crossover probability increases. The standard deviation results show a similar trend. The default binary and arithmetic crossover values of 0.3 and 0.4 are seen to fall inside the minima of



(a) Pure genetic algorithm mean and median.

(b) Pure genetic algorithm standard deviation.

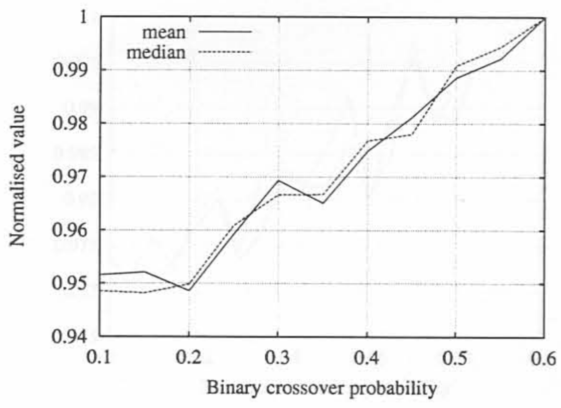
(c) Hybrid algorithm mean and median.

(d) Hybrid algorithm standard deviation.

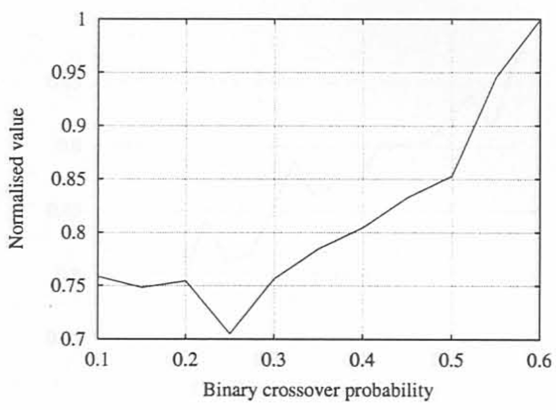
Figure 4.5: Arithmetic crossover results.

the average and standard deviation plots.

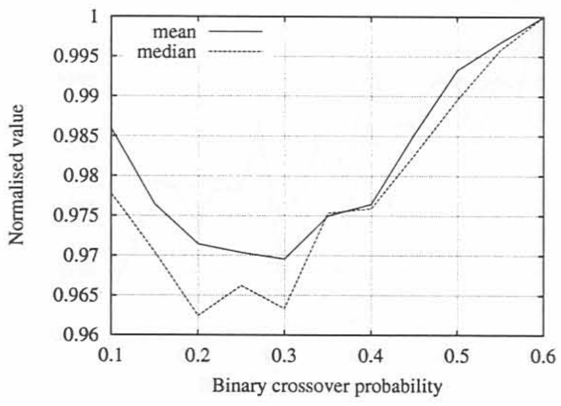
The results for boundary, binary, uniform, and non-uniform mutation are given in Figures 4.7 to 4.10. The average results show little or no trend as the mutation probability increases and the standard deviation results also show only a small trend. Where there is a trend, the error is seen to increase with mutation probability. The only exception to this is obtained with uniform mutation where the error decreases as the probability of uniform mutation increases. The default values for each type of mutation (given in Table 3.1) are



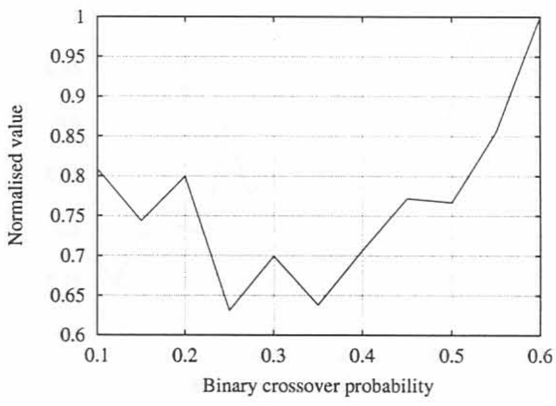
(a) Pure genetic algorithm mean and median.



(b) Pure genetic algorithm standard deviation.



(c) Hybrid algorithm mean and median.

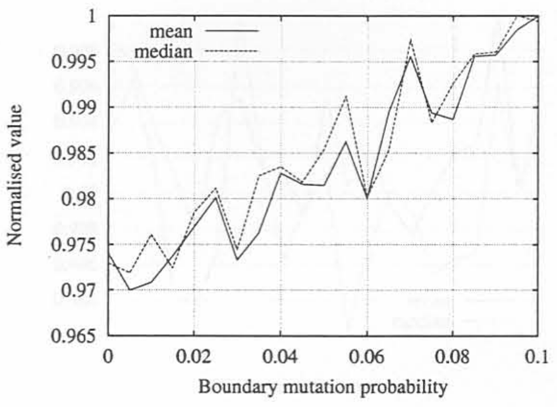


(d) Hybrid algorithm standard deviation.

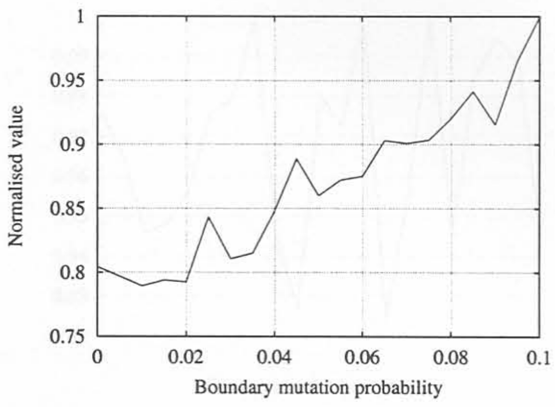
Figure 4.6: Binary crossover results.

thus shown to have only a very small effect on the performance of the algorithm.

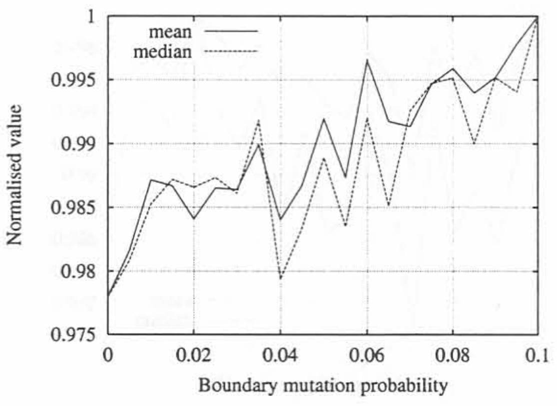
This does not mean that mutation should be ignored as shown in Table 4.14. The values in Table 4.14 are the error obtained without mutation divided by the error with mutation, averaged over all possible lengths. The standard deviation for networks with only one element is zero, so the error cannot be calculated in that case. It is clear from these results that, while mutation has only a small effect on the algorithm, it is still necessary with the mean, median, and standard deviation being higher without mutation in almost all cases.



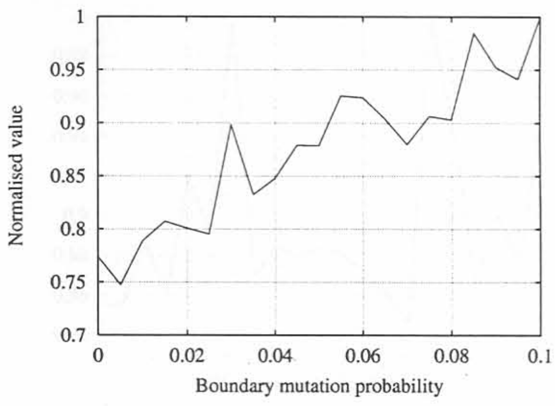
(a) Pure genetic algorithm mean and median.



(b) Pure genetic algorithm standard deviation.



(c) Hybrid algorithm mean and median.



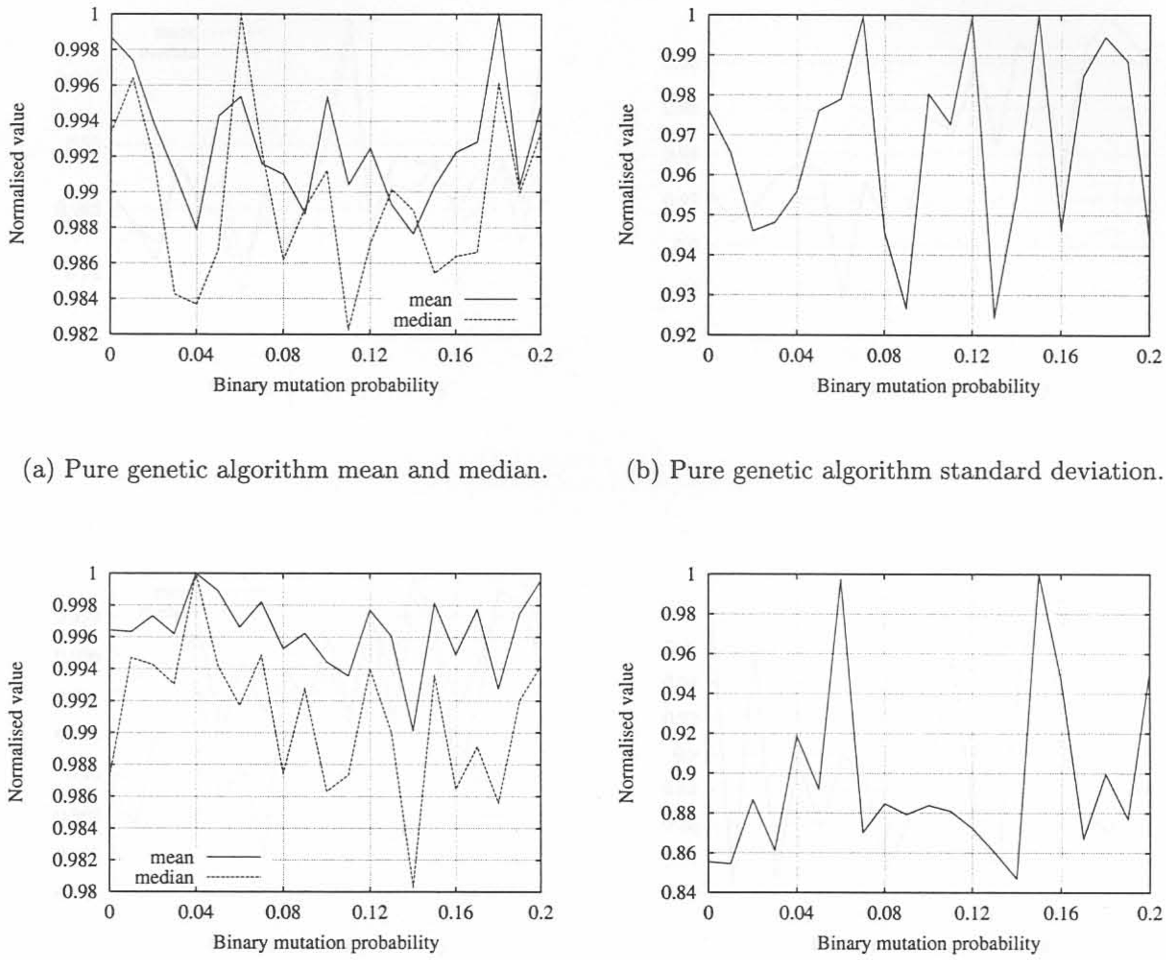
(d) Hybrid algorithm standard deviation.

Figure 4.7: Boundary mutation results.

4.3.4 Tournament Size

The tournament size was varied around its default value of three to determine the effect of the bias towards good individuals. Both a pure genetic algorithm and a hybrid genetic algorithm with an integrated local optimiser were considered.

The combined results as the tournament size varies are given in Figure 4.11. The average results show that the error decreases rapidly and then slowly increases as the tournament



(a) Pure genetic algorithm mean and median.

(b) Pure genetic algorithm standard deviation.

(c) Hybrid algorithm mean and median.

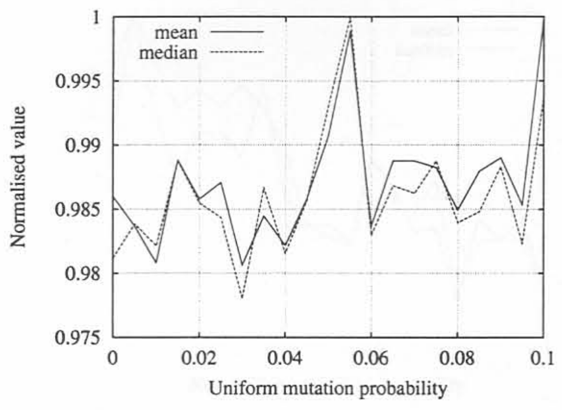
(d) Hybrid algorithm standard deviation.

Figure 4.8: Binary mutation results.

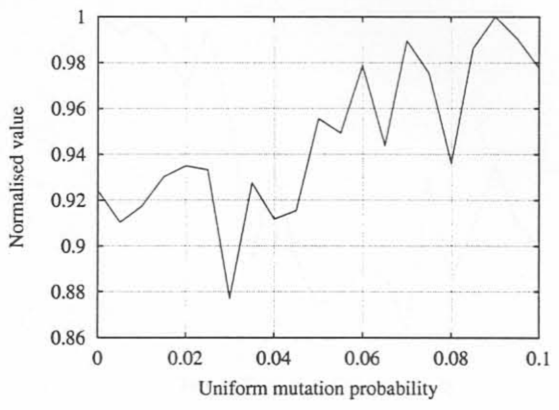
size increases. The standard deviation results show a similar trend. The default value of three for the tournament size is inside the minima of all of these results.

4.3.5 Number of Optimisation Steps

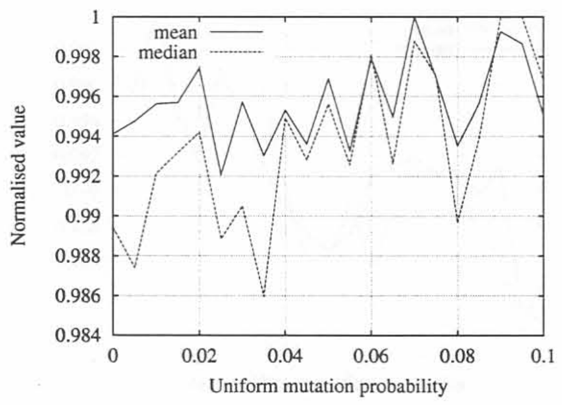
The number of local optimisation steps per iteration of a hybrid genetic algorithm is considered here. This parameter is very important because it has a major influence on both



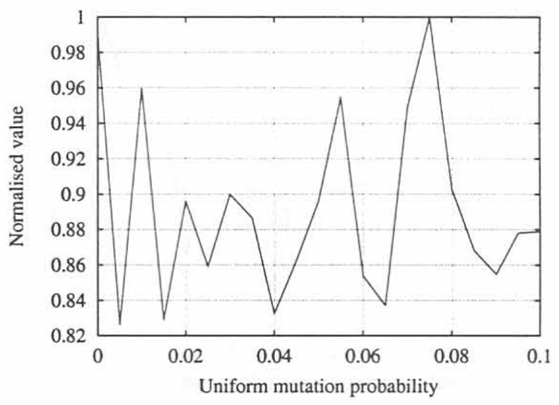
(a) Pure genetic algorithm mean and median.



(b) Pure genetic algorithm standard deviation.



(c) Hybrid algorithm mean and median.



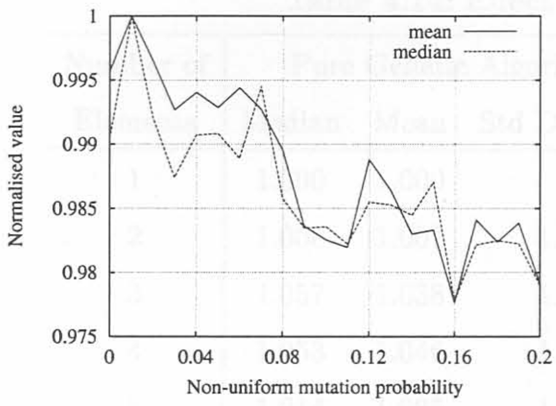
(d) Hybrid algorithm standard deviation.

Figure 4.9: Uniform mutation results.

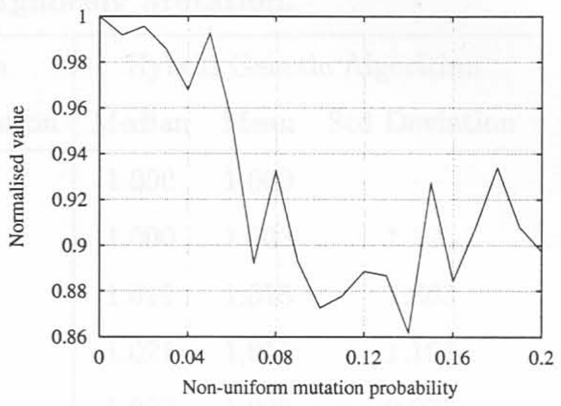
the performance of the algorithm and the amount of time required to run the algorithm.

The default values for the local optimiser parameters were determined during the development of the algorithm and were previously given in Table 3.5 on page 103. These parameters were then fixed and the number of local optimiser steps was varied around its default value of two. Obviously only the hybrid genetic algorithm was considered because the pure genetic algorithm does not have a local optimiser.

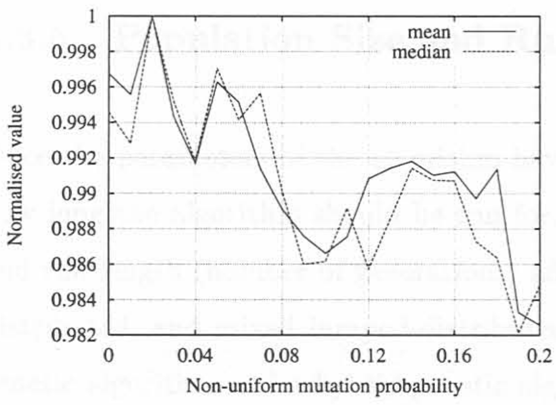
The results as the number of local optimisation steps are varied are given in Figure 4.12.



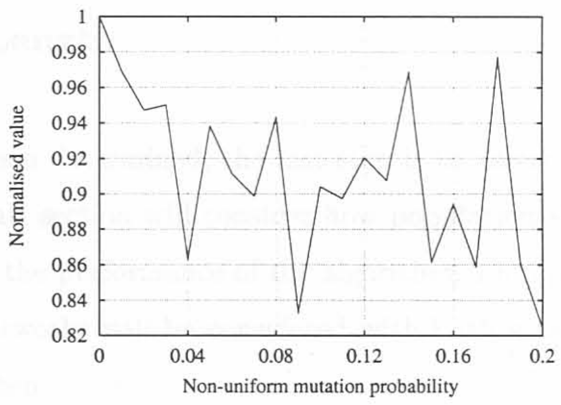
(a) Pure genetic algorithm mean and median.



(b) Pure genetic algorithm standard deviation.



(c) Hybrid algorithm mean and median.



(d) Hybrid algorithm standard deviation.

Figure 4.10: Non-uniform mutation results.

The average results show that the error decreases as the number of local optimisation steps increases and the standard deviation results show a similar trend. The default value of two is seen to be a little low, but it must be borne in mind that the run time increases linearly with the number of local optimisation steps.

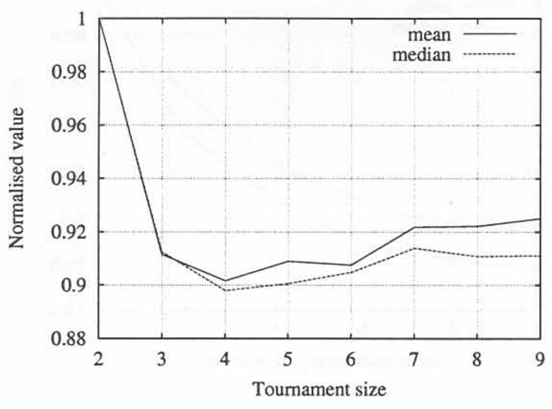
Table 4.14: Effect of Ignoring Mutation.

Number of	Pure Genetic Algorithm			Hybrid Genetic Algorithm		
Elements	Median	Mean	Std Deviation	Median	Mean	Std Deviation
1	1.000	1.000	–	1.000	1.000	–
2	1.008	1.007	1.110	1.000	1.003	1.175
3	1.057	1.038	1.188	1.019	1.015	1.303
4	1.053	1.046	1.274	1.021	1.011	1.107
5	1.014	1.025	1.075	1.007	1.008	0.973
6	1.007	1.012	1.084	0.983	1.005	0.981

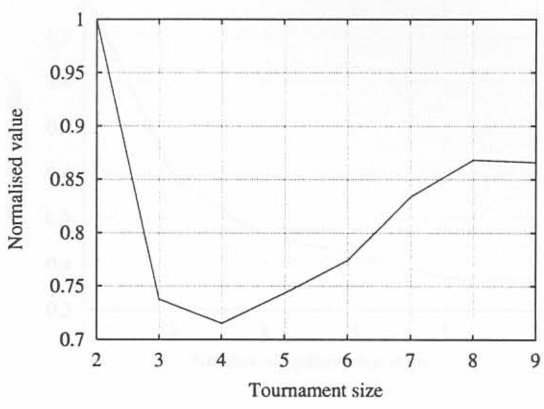
4.3.6 Population Size and Run Length

Once the parameters of the algorithm have been determined, the last step is to ascertain how long the algorithm should be run for. This section will consider how population size and run length (number of generations) affect the performance of the algorithm. Lumped, distributed, and mixed lumped-distributed networks will be considered with both a pure genetic algorithm and a hybrid genetic algorithm.

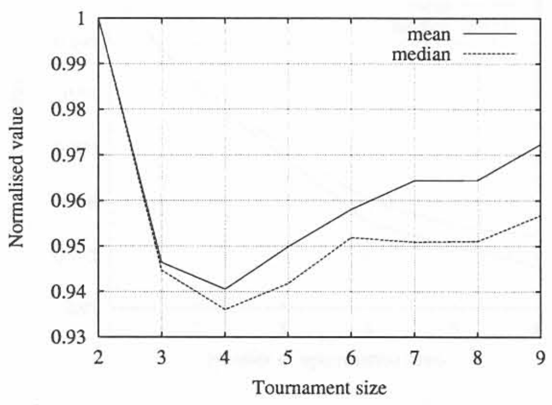
The combined results as the number of local optimisation steps are varied are given in Figures 4.13 and 4.14 on pages 133 and 134. The results show that the error decreases as the population size and the number of generations increase. The change in error slows down dramatically once a certain threshold has been reached, so increasing the population size and number of generations only leads to a small improvement once good results are obtained.



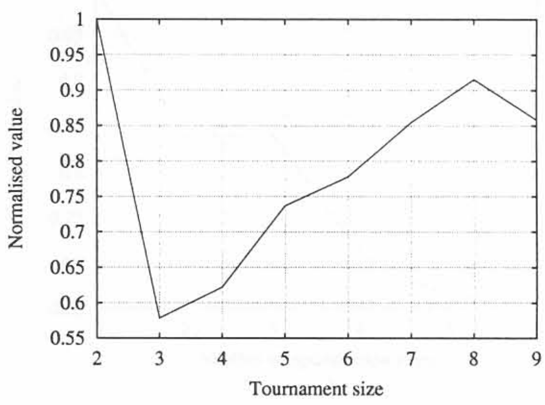
(a) Pure genetic algorithm mean and median.



(b) Pure genetic algorithm standard deviation.



(c) Hybrid algorithm mean and median.



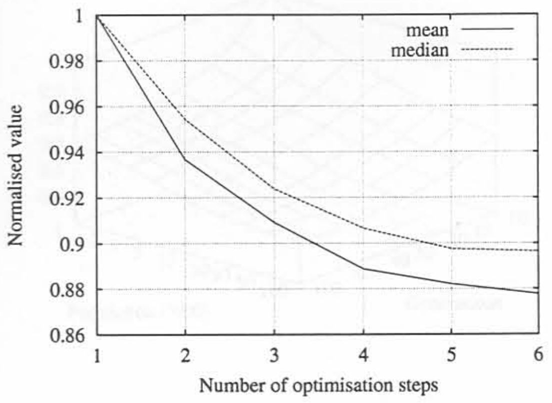
(d) Hybrid algorithm standard deviation.

Figure 4.11: Tournament size results.

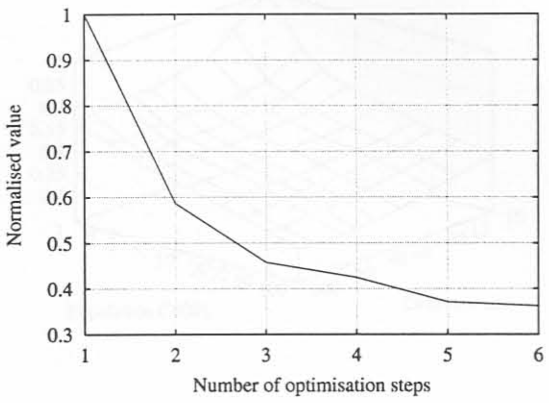
4.4 Run Times

The time required to run the algorithm is considered in this section. The lumped, mixed, and distributed cases are considered, with the distributed case being considered for both perfect transmission lines and microstrip lines.

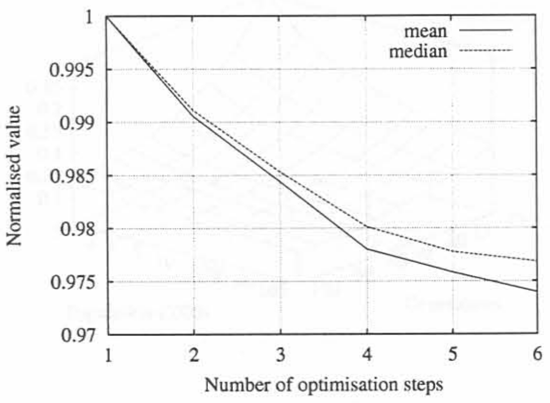
The time was determined using a 500 MHz Personal Computer (PC) with 192 MB of memory running Linux with kernel version 2.4.8–26mdk and version 2.96 of the GNU C++



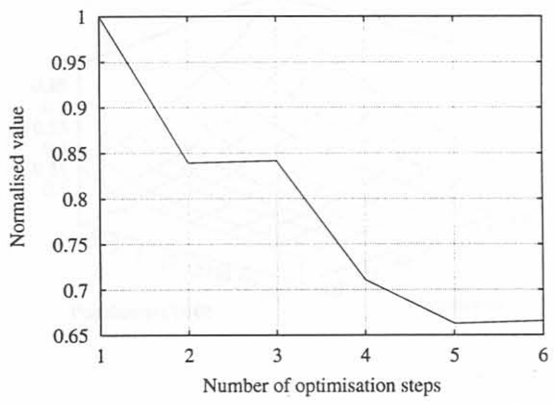
(a) Lumped elements mean and median.



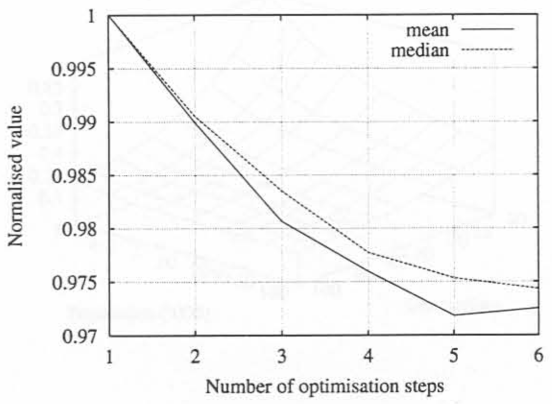
(b) Lumped elements standard deviation.



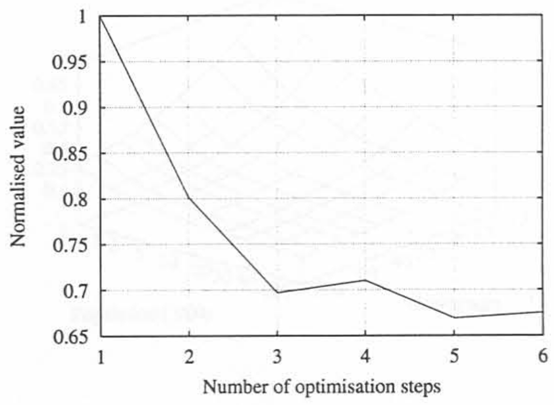
(c) Distributed elements mean and median.



(d) Distributed elements standard deviation.

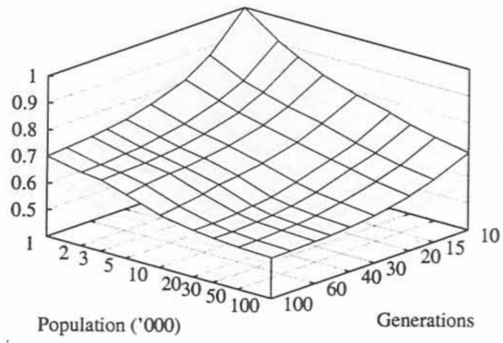


(e) Mixed elements mean and median.

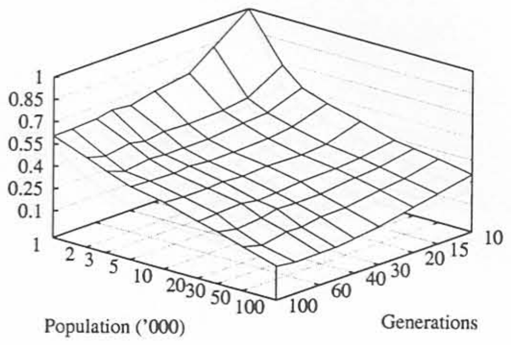


(f) Mixed elements standard deviation.

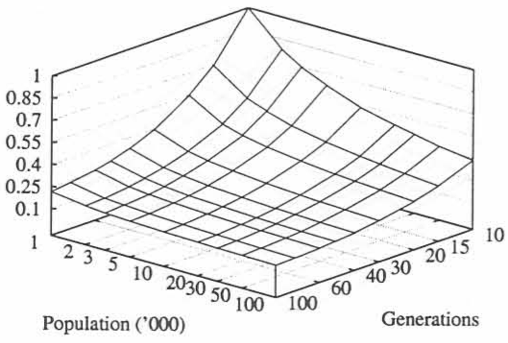
Figure 4.12: Number of local optimisation steps results.



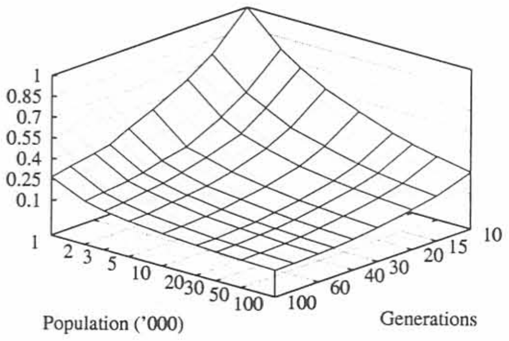
(a) Lumped elements median.



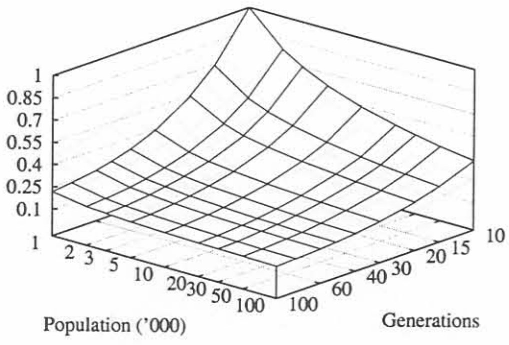
(b) Lumped elements standard deviation.



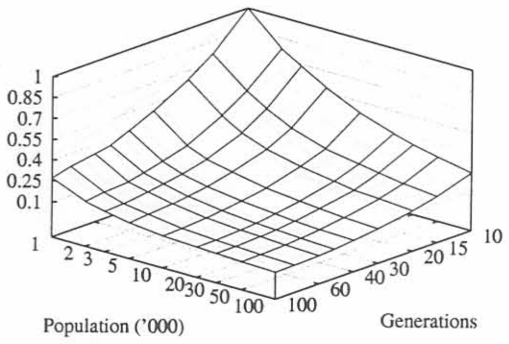
(c) Distributed elements median.



(d) Distributed elements standard deviation.

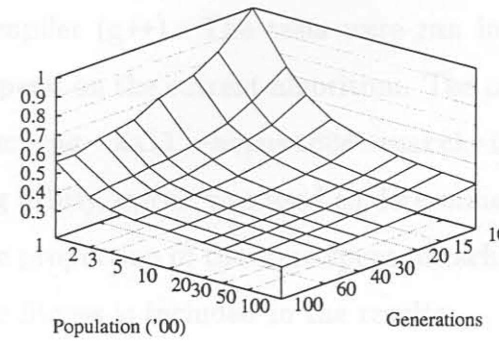


(e) Mixed elements median.

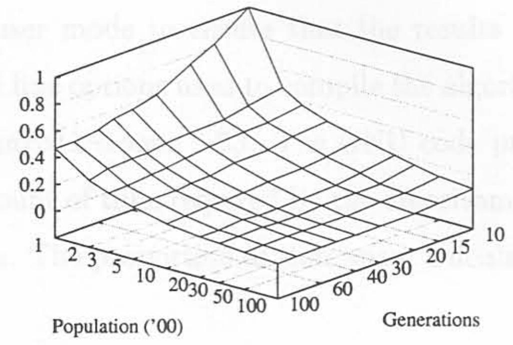


(f) Mixed elements standard deviation.

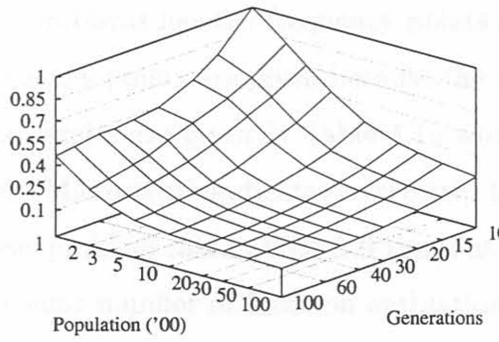
Figure 4.13: Results as population size and number of generations are varied for a pure genetic algorithm.



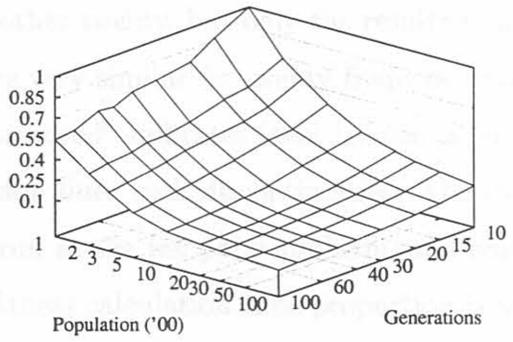
(a) Lumped elements median.



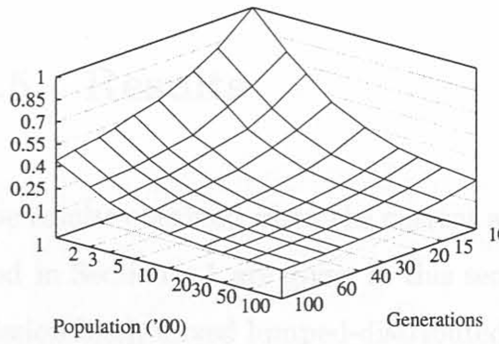
(b) Lumped elements standard deviation.



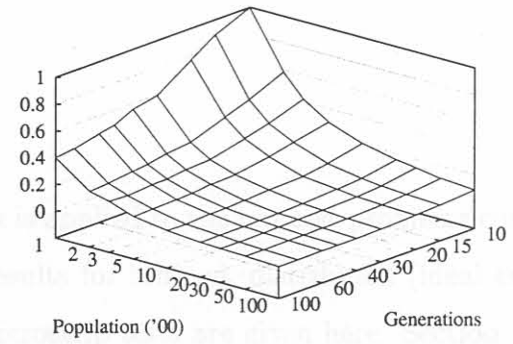
(c) Distributed elements median.



(d) Distributed elements standard deviation.



(e) Mixed elements median.



(f) Mixed elements standard deviation.

Figure 4.14: Results as population size and number of generations are varied for a hybrid genetic algorithm.

compiler (g++). The tests were run in single user mode to ensure that the results only depend on the current algorithm. The command line options used to compile the algorithm are: `-pg -Wall -mcpu=i686 -march=i686 -funroll-loops -O3`. The GNU code profiling utility, `gprof` was used to determine the amount of time required by the algorithm and the proportion of the time spent in each function. The proportion of time spent calculating the fitness is included in the results.

The results were averaged over ten runs of the algorithm in each case to characterise variations that can occur. The test problems have complex impedances at both the input and output with the impedances being constant over the whole frequency range. One of the test problems has ten frequency points and the other twenty, but only the results with ten frequency points are given because the trends are very similar for twenty frequency points. The results are given in Table 4.15 where "Distributed" indicates ideal transmission lines and "Microstrip" indicates microstrip transmission lines with discontinuities. The twenty point problem takes about 1.9 times as long to run as the ten point problem, and requires the same number of function evaluations. The fitness calculation time proportion is about 2 percentage points higher in the microstrip case than in the ideal transmission line case. Significantly, discontinuity calculations (microstrip impedance, width step, and T-junction effects) take almost 80% of the total time in the microstrip case.

4.5 Results

The results obtained when the current algorithm is applied to the ten test problems considered in Section 4.1 are given in this section. Results for lumped, distributed (ideal transmission line), mixed lumped-distributed, and microstrip tests are given here. Section 4.5.1 compares the results obtained by the current algorithm with published results.

The default genetic algorithm parameter values given in Table 3.1 on page 82, the default local optimiser values given in Table 3.5 on page 103, and the default component ranges

Table 4.15: Run Time Results.

Parameter	Lumped	Mixed	Distributed	Microstrip
Minimum time	98.6 s	98.8 s	97.9 s	1700.7 s
Median time	99.7 s	99.7 s	99.7 s	1706.2 s
Maximum time	101.7 s	100.5 s	102.1 s	1717.7 s
Minimum fitness %	97.0%	96.9%	96.9%	99.5%
Median fitness %	97.2%	97.1%	97.2%	99.5%
Maximum fitness %	97.4%	97.2%	97.4%	99.6%
Minimum evaluations	3,408,870	3,042,606	3,406,437	3,362,304
Median evaluations	3,415,986	3,415,792	3,418,184	3,368,012
Maximum evaluations	3,435,256	3,429,292	3,430,717	3,376,443

given in Table 3.3 on page 87 were used for these tests. The microstrip substrate parameters used are given in Table 3.4 on page 87. The transmission line elements in the distributed results given below do not use the default parameters given in Table 3.3, but rather have the same parameters as the microstrip lines to allow comparisons between the ideal case and the case where dispersion and discontinuities are accounted for. The microstrip parameters were converted to characteristic impedances and line lengths using the low-frequency equations given in Sections 2.4.1 and 2.4.2 on pages 66 and 68. The ratios of the widths of microstrip lines at a discontinuity was limited to a maximum value of 5, and crosses were not allowed to ensure that the discontinuity calculations are accurate.

The mixed results do not consider discontinuities and dispersion for the transmission lines, and solder pads for the lumped components. These effects could have been included, but the inclusion of these effects would have dramatically increased the time to run the algorithm to closer to the time required for microstrip tests (see Section 4.4) without greatly adding to the value of the results.

The microstrip networks can start with a parallel element because the algorithm assumes that all microstrip networks are bounded by 50 Ω transmission lines. This is necessary to

Table 4.16: Results for Problem 1 (VSWR).

Present algorithm	Elements	Minimum	Median	Maximum
Lumped	6	1.180	1.184	1.257
Distributed	3	1.242	1.273	1.324
Mixed	3	1.245	1.285	1.347
Microstrip	3	1.245 / 1.268	1.277 / 1.281	1.339 / 1.263

Table 4.17: Results for Problem 2 (VSWR).

Present algorithm	Elements	Minimum	Median	Maximum
Lumped	3	2.222	2.222	2.222
Distributed	3	2.197	2.340	2.433
Mixed	3	2.189	2.222	2.587
Microstrip	6	1.433 / 6.202	1.654 / 13.81	1.934 / 374.7

complete the discontinuities (width steps or T-junctions) at the extremities of the networks.

Note that two results are given in the microstrip cases. The first set of results are those obtained using the present algorithm while the second set are those obtained using EEsof. A comparison between these two sets of results has already been given in Section 4.2.2.

The results for Problems 1, 2, 3, 4, 9, and 10 are listed as VSWR values, and the results for Problems 5, 6, 7, and 8 are given as decibel errors. This was done because the first group of problems require the match to be as good as possible (matching network gain must be as high as possible), and the second group of problems requires a specific gain to be obtained at each frequency.

The results for some specific cases are given in Tables 4.16 to 4.25. The best networks obtained in these cases are given in Figures B.1 to B.10 on pages 175 to 184. The number of elements used in each case is the same as published results where these are available, and produces a VSWR of less than 1.5 or a gain error of less than 0.1 dB where published

Table 4.18: Results for Problem 3 (VSWR).

Present algorithm	Elements	Minimum	Median	Maximum
Lumped	3	1.676	1.676	2.024
Distributed	3	1.779	1.848	2.020
Mixed	5	1.586	1.680	1.795
Microstrip	2	1.308 / 2.514	1.643 / 9.728	2.293 / 33.74

Table 4.19: Results for Problem 4 (VSWR).

Present algorithm	Elements	Minimum	Median	Maximum
Lumped	5	2.929	2.940	2.988
Distributed	6	1.478	2.387	3.892
Mixed	6	2.455	2.980	3.430
Microstrip	6	1.269 / 49.83	1.708 / 251.9	2.324 / 952.7

Table 4.20: Results for Problem 5 (dB error).

Present algorithm	Elements	Minimum	Median	Maximum
Lumped	4	4.538×10^{-2}	0.1302	0.1589
Distributed	3	8.192×10^{-2}	0.1046	0.1353
Mixed	3	8.796×10^{-2}	0.1021	0.1336
Microstrip	3	0.0846 / 0.1017	0.0978 / 0.1150	0.1278 / 0.1336

results are not available. Six element solutions are given where a VSWR of less than 1.5 or a gain error of less than 0.1 dB is not possible. The number of elements in the microstrip case is the same as in the distributed case to allow comparisons to be made. The only exceptions are the solutions to Problems 2 and 3, and the lumped and mixed solutions to Problem 7, where adding extra elements has a small ($< 5\%$) effect.

The complete lumped, distributed, mixed, and microstrip results are given in Tables A.1

Table 4.21: Results for Problem 6 (dB error).

Present algorithm	Elements	Minimum	Median	Maximum
Lumped	5	7.960×10^{-2}	0.1750	0.2046
Distributed	4	6.513×10^{-2}	0.2289	0.3273
Mixed	3	9.332×10^{-2}	0.2335	0.4957
Microstrip	4	0.2302 / 0.2855	0.3165 / 0.3578	0.4621 / 0.4773

Table 4.22: Results for Problem 7 (dB error).

Present algorithm	Elements	Minimum	Median	Maximum
Lumped	5	0.3166	0.4611	0.7700
Distributed	6	0.2214	0.2625	0.2891
Mixed	5	0.2442	0.2861	0.3390
Microstrip	6	0.2041 / 0.2277	0.2558 / 0.2835	0.2778 / 0.3398

Table 4.23: Results for Problem 8 (dB error).

Present algorithm	Elements	Minimum	Median	Maximum
Lumped	6	0.2449	0.3005	0.4640
Distributed	6	6.850×10^{-2}	0.1841	0.3166
Mixed	6	9.805×10^{-2}	0.2451	0.4516
Microstrip	6	0.0967 / 0.2389	0.2304 / 0.3059	0.4620 / 0.5000

to A.40 on pages 159 to 172.

A number of trends can be observed in the results. The worst results are typically very poor, but the good median results show that the solutions are heavily skewed towards good results. Solutions with more elements typically have a higher standard deviation than solutions with fewer elements. The distributed results are usually worse than the lumped results because the range of parameter values is much more limited to ensure that the

Table 4.24: Results for Problem 9 (VSWR).

Present algorithm	Elements	Minimum	Median	Maximum
Lumped	4	1.214	1.268	1.333
Distributed	4	1.377	1.619	1.845
Mixed	3	1.259	1.484	1.867
Microstrip	6	1.576 / 3.050	2.662 / 18.83	3.498 / 460.2

Table 4.25: Results for Problem 10 (VSWR).

Present algorithm	Elements	Minimum	Median	Maximum
Lumped	3	1.467	1.557	2.063
Distributed	3	1.427	1.551	1.711
Mixed	3	1.416	1.592	1.816
Microstrip	3	1.604 / 1.680	1.807 / 1.832	1.964 / 1.987

distributed elements are realisable. The mixed results are typically at least as good as the better of the lumped and distributed results, but this is not always the case. Some of the solutions to Problems 2, 3, 4, and 7 have a number of series transmission lines that all have the same impedance in the ideal transmission line case, and the same line widths in the microstrip case.

The maximum line width ratio at discontinuities in the microstrip case is violated by a small amount in some of the solutions as shown in Table 4.26. The best solutions of every length for each run of the algorithm are considered in Table 4.26. Of a total of 19650 width steps, only 216 (1.1%) violate the maximum width ratio, with 203 (94%) of these violations being accounted for by problem 8. There are 3675 T-junctions in the best solutions with only 6 of these (0.2%) having width ratio violations.

The microstrip results are comparable to the distributed results in most cases, but there are some cases where the microstrip results are worse than the distributed results. The best

Table 4.26: Maximum Line Width Violations.

Problem	Width Step			T-Junction		
Number	Max. Ratio	Total	Violations	Max. Ratio	Total	Violations
1	4.89	2664	0 (0.0%)	3.03	18	0 (0.0%)
2	5.02	1450	1 (0.1%)	5.59	625	1 (0.2%)
3	4.94	1654	0 (0.0%)	5.02	523	1 (0.2%)
4	3.38	1450	0 (0.0%)	5.43	625	1 (0.2%)
5	5.48	2498	8 (0.3%)	3.55	101	0 (0.0%)
6	5.01	1458	1 (0.1%)	5.10	621	1 (0.2%)
7	5.00	2534	0 (0.0%)	4.23	83	0 (0.0%)
8	5.10	2578	203 (7.9%)	4.88	61	0 (0.0%)
9	5.01	2308	3 (0.1%)	5.01	196	1 (0.5%)
10	4.90	1056	0 (0.0%)	5.01	822	1 (0.1%)
total	5.48	19650	216 (1.1%)	5.59	3675	6 (0.2%)

solutions for Problems 1, 2, 5, 7, and 8 are examples of solutions that are very similar in the distributed and microstrip cases, and the best solutions for Problems 3, 5, 8, 9, and 10 are examples of solutions where the microstrip case gives worse results than the distributed case. The best solutions to Problems 2, 4, and 6 show that the microstrip solutions can be better than the distributed solutions.

4.5.1 Comparison to Published Results

This section will compare the results obtained using this algorithm with those that have been published in the literature. Obviously only those test problems that were obtained from the literature can be considered here. The main difficulty with these comparisons is that the vast majority of published results only consider lumped network synthesis, but the current algorithm can consider other cases.

Most of the published results use the transducer gain (G_T) or VSWR to determine how good

Table 4.27: Results for Problem 1 (VSWR).

Network	Result	Elements	Transformers
Abrie (lumped)	1.19	6	0
Abrie (commensurate)	1.06	4	0
Abrie (non-commensurate)	1.25	6	0
Abrie (microstrip)	1.41/1.40	4	0
Present algorithm	Minimum	Median	Maximum
Lumped (6 elements)	1.180	1.184	1.257
Distributed (4 elements)	1.089	1.156	1.258
Distributed (6 elements)	1.048	1.090	1.146
Mixed (4 elements)	1.089	1.179	1.295
Mixed (6 elements)	1.048	1.098	1.184
Microstrip (4 elements)	1.25/1.26	1.28/1.30	1.32/1.35
Microstrip (6 elements)	1.06/1.11	1.11/1.08	1.23/1.16

a match is. Note that, unlike the other error functions considered here, a higher transducer gain indicates a better match, and a perfect match corresponds to a transducer gain of one. The results for the current algorithm given below were derived from the decibel errors obtained during the tests. This was done by using equations (3.25) and (3.28) to convert to transducer gain, and (3.20) and (3.29) to calculate the VSWR. The mean, median, and maximum results were then calculated.

Note that non-ideal transformers are considered as a single element in the determination of the number of elements in the network. Many of the references do not consider an ideal transformer to be an element and this convention will be followed here, but the number of ideal transformers required will be given. The present algorithm will use the same number of elements as the published results without any ideal transformers.

The algorithm parameters used for these tests are the same as the parameters used in Section 4.5.

Table 4.28: Results for Problem 2 (G_T).

Author	Result	Elements	Transformers
Fano [57, 58]	0.816	4	0
Carlin [4]	0.837	3	0
Chen [5]	0.831	4	0
Carlin and Amstutz [5]	0.849	3	0
Dedieu *et al.* [82]	0.855	3	0
Present algorithm	Minimum	Median	Maximum
Lumped (3 elements)	0.8562	0.8562	0.8562

Problem 1 is taken from Abrie [2]. This problem in unique because lumped and distributed results are available. Unfortunately, the distributed results were obtained by fixing either the line length (commensurate case) or the line impedances (non-commensurate case), so the results are not completely comparable. The results published by Abrie and the results obtained using the present algorithm are given in Table 4.27 with the errors given as VSWR values. The best solutions obtained are shown in Figure B.1 on page 176. The lumped solution is similar to the solution obtained by Abrie [2]. The microstrip line results were obtained using one of the examples that is supplied with the demonstration version of MultiMatch Mosaic which was obtained from Ampsa's web page (www.ampsa.com). The only differences in the MultiMatch example are that two extra transmission lines are included at the load and source, and the line impedances are constrained to be between 25 and 85 Ω. These changes were also made to the current algorithm in the microstrip case to allow the results to be compared directly. The second value given in each of the microstrip cases is the value obtained using EEsof. The performance of the current algorithm on this problem is seen to be comparable to the published results, although the worst results are poor. The microstrip results are particularly impressive because they include discontinuity effects.

Problem 2 was originally proposed by Fano [57, 58], but has also been used by Carlin [4],

Table 4.29: Results for Problem 3 (G_T).

Author	Result	Elements	Transformers
Carlin and Yarman [8] (analytic)	0.847	3	1
Carlin and Yarman [8] (real-frequency)	0.932	3	1
Yarman and Fettweis [9]	0.927	3	1
Yarman and Aksen [87]	≈0.94	3	0
Dedieu *et al.* [82]	0.938	3	1
Present algorithm	Minimum	Median	Maximum
Lumped (3 elements)	0.8853	0.9362	0.9362
Mixed (3 elements)	0.9105	0.9362	0.9362

Carlin and Amstutz [5], and Dedieu *et al.* [82]. Carlin and Amstutz [5] have also applied the formulae developed by Chen [63] to this problem, and these analytic results are also given. The results are shown in Table 4.28 with the errors given as transducer gains. The best solutions are shown in Figure B.2 on page 177 where the component values have been scaled back to the original frequency band. The results are similar to the solutions obtained by Carlin [4], Carlin and Amstutz [5], and Dedieu *et al.* [82]. The performance of the algorithm developed here is seen to at least as good as the published results.

Problem 3 is used by Carlin and Yarman [8], Yarman and Aksen [87], Yarman and Fettweis [9], and Dedieu *et al.* [82]. Yarman and Carlin [8] give results for both the analytic and real-frequency algorithms developed in that work. The results by Yarman and Aksen [87] are for a mixed lumped-distributed network where a lumped prototype is transformed to a mixed network with three elements, two of which are series transmission lines, and then optimised. Yarman and Aksen [87] do not give a value for the final network obtained, so the value given in Table 4.29 was read off the frequency response of the final network. The results are shown in Table 4.29 in the form of transducer gains, and the current algorithm is again seen to achieve results that are at least as good as the published results in the vast majority of cases. The best solutions are shown in Figure B.3 on page 178 where the

Table 4.30: Results for Problem 4 (G_T).

Author	Result	Elements	Transformers
Carlin and Yarman [8]	0.742	5	1
Yarman and Fettweis [9]	0.722	5	1
Dedieu *et al.* [82]	0.750	5	1
Present algorithm	Minimum	Median	Maximum
Lumped (5 elements)	0.7515	0.7576	0.7590

Table 4.31: Results for Problem 5 (decibel error).

Network	Result	Elements	Transformers
Abrie (lumped)	0.05 dB	4	0
Present algorithm	Minimum	Median	Maximum
Lumped (4 elements)	0.04538 dB	0.1302 dB	0.1481 dB

component values have been scaled back to the original frequency band. The results are similar to the solutions obtained by Carlin and Yarman [8], Yarman and Fettweis [9], and Dedieu *et al.* [82].

Problem 4 appears in the papers by Carlin and Yarman [8], Yarman and Fettweis [9], and Dedieu *et al.* [82]. The results are shown in Table 4.30 as transducer gains, and the current algorithm is seen to obtain better results than the published algorithms. The best solutions are shown in Figure B.4 on page 179 where the component values have been scaled back to the original frequency band. The results are similar to the solutions obtained by Carlin and Yarman [8], but have a different structure to the solutions obtained by Yarman and Fettweis [9], and Dedieu *et al.* [82].

Problem 5 is a double matching problem taken from Abrie [2]. The results are given in Table 4.31, where the results are given as maximum gain errors in dB. The best solutions are shown in Figure B.5 on page 180. The lumped solution is seen to be similar to one of the solutions obtained by Abrie [2]. The current algorithm has tremendous difficulty obtaining a good result on this problem, despite the fact that the best result is comparable to the best published result.

Problem 6 is also taken from the literature, but unfortunately, Abrie [2] only gives results for the amplifier this network is part of, not the network itself. The best solutions are shown in Figure B.6 on page 181. The lumped solution has a different structure to the solution obtained by Abrie [2].

4.6 Summary

A large number of tests were conducted to quantify the performance of the algorithm developed here, with the results being summarised in this chapter.

Ten test problems from a variety of sources were used to test the algorithm as thoroughly as possible. The problems include many different cases, and are a mix of established and new problems.

The accuracy of the fitness calculations was verified by comparing results to those obtained by EEsof. The models used for ideal components were found to be exact, and the models used for microstrip dispersion and discontinuities were found to be in very good agreement with the EEsof models. The only notable exception is the cross model which is poor. The results obtained using complete circuits are also excellent except where the substrate is thick and has a low dielectric constant.

The algorithm options and parameters were then modified to determine their effect on the performance of the algorithm. The Pareto-like optimality and local optimiser were found to improve the results obtained, and the default parameters were found to be very close to the optimal values.

The results obtained are impressive with the best results being at least as good as the best published results. The main drawback is that the results are not consistent, and the worst results are usually poor. Results for longer individuals, and mixed and distributed networks display more variance than the results for shorter individuals in the lumped case.

The implications of these results are discussed in Chapter 5.

Chapter 5

Conclusion

This section will present a brief discussion of the results obtained in Chapter 4 and their significance. Some suggestions about how this work can be extended are also given.

The verification of the accuracy of the calculations and models implemented is considered in Section 5.1. The effects of the algorithm's parameters and options are discussed in Section 5.2. Section 5.3 reviews the results obtained, and Section 5.3.1 considers the implications of the comparisons between these results and published results. Finally, some suggestions for future development of this work are given in Section 5.4, followed by a brief conclusion in Section 5.5.

5.1 Accuracy Verification

The tests conducted in Section 4.2 to verify the accuracy of the calculations and models used in this dissertation are briefly considered here.

The accuracy of the ideal components (inductors, capacitors, and transmission lines) is very high because these ideal components have exact models that characterise their performance. The microstrip line, open-end effect, and via inductance models used here are also in excellent agreement with those provided with EEsof. The accuracy of the width step is acceptable if the ratio of the two lines' widths is kept low, and the frequency is low. The T-junction model is only accurate at very low frequencies, and when the line widths and relative dielectric constants are small. The cross model is not expected to perform well because it is a modified T-junction model is used rather than a true cross model.

The results with full circuits are very impressive except for the microstrip results for Problems 2 to 4. These problems all have the same substrate which has a relative dielectric constant of 1 and a height of 3 mm. These two factors mean that there will be large fringing fields leading to dramatic discontinuity effects. The disagreement between the current algorithm and EEsof is understandable under these circumstances. Problem 5 uses the same substrate, but does not have large errors because the maximum frequency in this case is much lower than the maximum frequency for Problems 2 to 4. The agreement with EEsof results is thus seen to be good as long as the substrate height is small compared to the wavelength in the substrate, and fringing fields are not excessive.

5.2 Options and Parameter Values

The algorithm developed has a large number of options and parameters that affect its performance. The tests summarised in Section 4.3 will be considered here.

The effect of the Pareto-like optimality is remarkable. The results obtained using Pareto-like optimality to design networks with one to six elements are comparable to, and often better than, the results where the algorithm is applied to only one length at a time. This is despite the fact that the algorithm's total power is concentrated on one network length in the non-Pareto case, whereas the Pareto-like case divides the algorithm's power between a number of network lengths. This causes the processing time required in the non-Pareto case to be much higher than that required in the Pareto-like case. Some reasons for this remarkable result are the fact that different length solutions interact, and diversity is maintained. The various length solutions in a population work together in the Pareto-like case to produce better results in all network lengths simultaneously. For example, a good two element network usually forms the basis for a good three element network. Diversity is essential in a genetic algorithm, and simply means that the entire population does not converge to one part of the search space. Diversity is maintained in the Pareto-like case because the solutions of different lengths, while similar as noted above, are nevertheless different.

Messy genetic algorithms also allow multiple length solutions to coexist, but the approach used here is different. A messy genetic algorithm starts with a large number of short solutions and slowly constructs longer solutions with the sole objective of obtaining the best maximal length solution. The approach used here is to allow solutions of all lengths to coexist at all times with the objective of simultaneously obtaining the best solutions for every length.

The addition of a local optimiser to the genetic algorithm leads to improvements to the results in all cases and dramatic improvements to long individuals. Genetic algorithms are known to have limitations when trying to find highly accurate results and will thus often miss the small range of component values with very good results, despite getting close to these good ranges. The addition of a local optimiser overcomes this problem because the local optimiser will take values close to a good result and move them to that good result (the local optimum). Long individuals are affected more than short individuals because they represent a more difficult problem for the algorithm because there are more possible combinations of parameters.

The default parameters for the genetic algorithm operator parameters produce very good results. The combined probability of arithmetic and binary crossover is 0.7 which is very close to the value of 0.6 recommended by Goldberg [17] for binary crossover in a binary genetic algorithm. When the crossover rates become too high the performance of the algorithm deteriorates because good individuals only have a very small probability of surviving unaltered to the next generation. Increasing mutation probability causes the algorithm performance to decrease because mutation tends to disrupt good results. However mutation is still necessary to ensure that the entire search space is considered and premature convergence does not occur. Unlike the other mutation operators, increasing the non-uniform mutation probability is seen to improve the algorithm performance because non-uniform mutation is similar to simulated annealing which is a good optimisation algorithm in its own right.

The tournament size results show that a compromise must be reached between large tournaments which lead to fast local search, and small tournaments which cover the whole search space. Section 4.3.4 shows that a value of three or four produces the best results in this case. These values close to the value of two that Bäck [43] states is commonly used.

An increase in the number of optimisation steps is expected to decrease the final error because more local search steps will isolate local minima faster. This reasoning is borne out by the results in Section 4.3.5. The problem is that the number of optimisation steps has a large influence on the time required to run the algorithm so large values are undesirable. Furthermore, if the number of local search steps becomes too large (larger than the values considered here), the algorithm will converge on local minima too fast, neglecting the global search and decreasing the algorithm's performance.

The performance and time requirements of the algorithm depend heavily on the population size and number of generations used. The results in Section 4.3.6 show that the performance of the algorithm initially improves rapidly when population size and number of generations are increased and then tapers off. This effect is due to the fact that once the best result is obtained with a high probability, further increases in population size and number of generations will only have a very small effect.

The algorithm takes a comparatively long time to run. The main factors that cause this are the extremely wide ranges of component values used, the time required to calculate fitness, and the number of function evaluations required to obtain a good result.

The wide range of component values was chosen to give worst-case results for the algorithm and better results can be expected with smaller component ranges.

The fitness evaluation comprises the vast majority of the processing time required by the algorithm and this time is very difficult to reduce. This effect could be minimised by reducing the number of function evaluations required, but the current algorithm is comparatively expensive in this regard.

This algorithm requires a comparatively large number of function evaluations to obtain a result because genetic algorithms are not the most efficient optimisation algorithms. The versatility of the current algorithm is largely due to the use of a genetic algorithm, and slower convergence is the price paid for this versatility. The convergence of the genetic algorithm was improved by using a local optimiser, but this requires gradient information which was calculated using finite differences. This means that the gradient is calculated by computing one extra fitness value for each variable thereby increasing the number of fitness evaluations required. Some possibilities for reducing the time and number of function evaluations required are given in Section 5.4.

5.3 Results

The results obtained when the algorithm is applied to the ten test problems are considered here.

While the worst results are often very poor, the best and median results are usually very good, showing that the algorithm finds good results in the vast majority of cases. This effect is probably due to the stochastic nature of the algorithm and is a known limitation of genetic algorithms. Another possibility is that the algorithm is being misled by the way solutions of differing lengths interact. This could be a problem when the best solution with three elements is not the basis of a good solution with four elements, for example.

Longer networks usually have a much higher variation in errors than shorter networks because longer networks represent a more challenging problem for the algorithm. The extra elements in longer networks mean that there are more possible combinations of elements that must be evaluated than for shorter networks.

The distributed solutions are generally worse than the lumped solutions, but these results do not mean that distributed solutions obtained from lumped prototypes will produce better results than those given here. This is because the distributed parameter ranges were limited to ensure that the distributed elements are realisable, while the lumped parameter ranges have no such limitations.

The fact that a number of transmission lines with the same characteristic impedance or line width are cascaded in the distributed solutions to Problems 4 and 7 simply means that a transmission line longer than the maximum specified length is required. Fewer elements could be used by allowing longer transmission lines, but in general, this is not a good approach. The algorithm could be modified to avoid this effect by ensuring that there is some minimum difference between the impedances of connected series transmission lines.

The mixed solutions usually obtain results that are at least as good as the better of the lumped and distributed solutions. This is because the lumped and distributed cases are simply instances of the mixed case. The mixed solutions are sometimes slightly worse than the lumped or distributed solutions because the mixed problem has significantly more possible combinations of elements than the other cases and is thus a much more challenging problem.

Some of the microstrip results violated the maximum allowable line width ratios at a discontinuity. This is expected because a penalty function was used and solutions that violate the constraints by a small amount are known to occur when penalty functions are used.

The microstrip results are similar to the distributed results in most cases, but the microstrip results are generally not quite as good as the distributed results. This is caused by the presence of discontinuities and the limitation placed on the microstrip line widths at a discontinuity. The importance of compensating for discontinuities is clearly shown by this observation.

5.3.1 Comparison to Published Results

The present algorithm is compared to other published results in Section 4.5.1 and the implications of these results are considered below.

The results obtained are impressive. The current algorithm performs at a level comparable to the best published results in all cases except Problem 5. The only algorithms that obtain similar performance are those developed by Abrie [2] and Dedieu *et al.* [82], but this is offset by the fact that these algorithms are more limited than the present one. The algorithm proposed by Dedieu *et al.* can only determine component values and must be supplied with a circuit configuration. The main limitation of the published version of Abrie's algorithm is that it is not able to vary both the impedance and length of transmission lines simultaneously. (However, this limitation has recently been removed, although results have not yet been published.) The consideration of mixed lumped-distributed networks in the new algorithm presented here is much more general than any other algorithm. Additionally, all published algorithms only consider one matching network size at a time whereas the current algorithm considers all lengths from one to a specified maximum. This means that this algorithm is able to achieve similar results to the best published techniques while being more versatile.

Problem 5 is the one case where the current algorithm does not perform as well as the published results. The stochastic nature of genetic algorithms means that there will be some variation in the results. Additionally the local optimiser used is poor, and a better local optimiser could lead to better results. It is significant that Abrie's [2] algorithm, which performs very well on this problem, uses a good local optimiser (a quasi-Newton method). Another possibility is that the current algorithm is being misled by the way solutions of differing lengths interact.

The main disadvantage of the current algorithm is that it is not as efficient as the published algorithms in terms of the number of function evaluations required. This problem has already been considered in Section 5.2, but the important points are repeated here. This algorithm is more versatile than other algorithms implying a more complex problem which is more difficult to solve. The advantages of this versatility include the fact that resonant sections are allowed, multiple network lengths are considered simultaneously, and both the length and impedance of transmission lines can be varied.

5.4 Extensions to This Work

While this work has led to the development of a very powerful optimisation and impedance matching algorithm, a number of further developments are possible. Some possibilities for such further development are considered below.

The current work is limited to the design of ladder networks, but this is not a fundamental limitation of the approach used here. Other structures such as those with component loops could be considered by an extended algorithm.

The natural next step after developing an impedance matching network design algorithm is to move towards a complete amplifier design algorithm. The algorithm described here has the potential to be developed into a complete amplifier design package and could even be extended to automatically select an active device.

Other problems such as filter and antenna design could also be considered because the advantages obtained here would also be useful in these problems. Applying the algorithm to other problems would involve little more than using a different fitness function.

The optimisation algorithm developed here has a number of unique properties and these could be developed further. The use of Pareto-like optimality to allow solutions with varying lengths has tremendous potential and will be considered further. The combination of binary and floating point operators could be extended to consider other genetic operators such as the fuzzy operators suggested by Klir and Yuan [111]. The integration of a genetic algorithm and a local optimiser could form the basis for the integration of further optimisation algorithms into a genetic algorithm.

The time required by this algorithm is its main limitation and could be reduced in a number of ways. A better local optimiser could be used to accelerate convergence, but this would entail major changes to the current algorithm. Eliminating the finite difference calculations by using an optimisation algorithm that does not require gradients could lead to an additional reduction in the number of function evaluations. The only drawback is that local optimisation algorithms that do not use gradients tend to converge more slowly than local optimisation algorithms that use gradients, so this change will not necessarily produce an improvement. Better initialisation of the algorithm could also be used to ensure that the algorithm has a good starting point, thereby reducing the number of function evaluations required to get near a good solution. This initialisation could, for example, be done using analytic techniques to design a matching network for a single frequency and allowing the algorithm to optimise these narrowband results to obtain a good broadband solution.

5.5 Summary

A new impedance matching algorithm based on a hybrid genetic algorithm has been developed. The consequences of the tests performed in Chapter 4 were presented above and are summarised below.

The results achieved with this new impedance matching algorithm compare very well with published results. This algorithm has a number of advantages over published techniques including the simultaneous synthesis of networks with multiple lengths, a very general consideration of mixed lumped-distributed networks, the simultaneous modification of both the length and width of transmission lines, and the inclusion of dispersion and discontinuity effects.

The hybrid genetic algorithm also has a number of unique features including the use of a Pareto-like optimality to obtain variable length solutions, and the simultaneous application of both binary and floating-point genetic operators. The variable length solution capability of the algorithm is particularly notable and warrants further investigation.

While the results achieved so far have been impressive, the algorithm can still be developed further and some suggestions for such development are given. These suggestions mainly relate to ways of overcoming the limitations of this algorithm, and other applications of the new techniques that have been developed.

Appendix A

Detailed Results

The full results for all the test problems are given below. The minimum, median, maximum, mean, and standard deviation values are given for all lumped, distributed, mixed lumped-distributed, and microstrip solutions.

The component ranges used are given in Table 3.3 on page 87 and the microstrip substrates used are given in Table 3.4 on page 87. The transmission line elements in the distributed results given below do not use the default parameters given in Table 3.3, but rather have the same parameters as the microstrip lines to allow comparisons between the ideal case and the case where dispersion and discontinuities are accounted for. The microstrip parameters were converted to characteristic impedances and line lengths using the low-frequency equations given in Sections 2.4.1 and 2.4.2 on pages 66 and 68. The ratios of the widths of microstrip lines at a discontinuity was limited to a maximum value of 5, and crosses were not allowed to ensure that the discontinuity calculations are accurate.

The mixed results do not consider discontinuities and dispersion for the transmission lines, and solder pads for the lumped components. These effects could have been included, but the inclusion of these effects would have dramatically increased the time to run the algorithm to closer to the time required for microstrip tests (see Section 4.4) without greatly adding

Table A.1: Problem 1 lumped results (VSWR).

Solution Length	Minimum	Median	Maximum	Mean	Standard Deviation
1	4.000	4.000	4.000	4.000	0.000
2	3.119	3.119	3.119	3.119	3.807×10^{-6}
4	1.601	1.602	1.607	1.602	1.210×10^{-3}
3	3.050	3.050	3.053	3.050	4.367×10^{-4}
5	1.527	1.539	1.575	1.541	1.156×10^{-2}
6	1.180	1.184	1.257	1.195	1.969×10^{-2}

Table A.2: Problem 1 distributed results (VSWR).

Solution Length	Minimum	Median	Maximum	Mean	Standard Deviation
1	2.763	2.763	2.764	2.763	8.261×10^{-5}
2	1.641	1.652	1.692	1.654	1.114×10^{-2}
3	1.242	1.273	1.324	1.274	1.819×10^{-2}
4	1.089	1.156	1.258	1.159	3.449×10^{-2}
5	1.062	1.116	1.181	1.117	2.662×10^{-2}
6	1.048	1.090	1.146	1.090	1.895×10^{-2}

to the value of the results.

The microstrip networks can start with a parallel element because the algorithm assumes that all microstrip networks are bounded by 50 Ω transmission lines. This is necessary to complete the discontinuities (width steps or T-junctions) at the extremities of the networks.

Table A.3: Problem 1 mixed lumped-distributed results (VSWR).

Solution Length	Minimum	Median	Maximum	Mean	Standard Deviation
1	2.763	2.763	2.764	2.763	1.375×10^{-4}
2	1.641	1.651	1.709	1.656	1.411×10^{-2}
3	1.245	1.285	1.347	1.285	2.339×10^{-2}
4	1.089	1.179	1.295	1.182	4.619×10^{-2}
5	1.077	1.134	1.204	1.133	3.145×10^{-2}
6	1.048	1.098	1.184	1.100	2.656×10^{-2}

Table A.4: Problem 1 microstrip results (VSWR).

Solution Length	Minimum	Median	Maximum	Mean	Standard Deviation
1	2.766 / 2.766	2.766 / 2.766	2.766 / 2.766	2.766	7.385×10^{-5}
2	1.644 / 1.653	1.656 / 1.660	1.704 / 1.657	1.659	1.097×10^{-2}
3	1.245 / 1.268	1.277 / 1.281	1.339 / 1.263	1.277	1.929×10^{-2}
4	1.090 / 1.182	1.186 / 1.179	1.273 / 1.243	1.180	3.900×10^{-2}
5	1.076 / 1.170	1.137 / 1.124	1.242 / 1.216	1.142	3.073×10^{-2}
6	1.057 / 1.107	1.107 / 1.084	1.225 / 1.155	1.108	2.438×10^{-2}

Table A.5: Problem 2 lumped results (VSWR).

Solution Length	Minimum	Median	Maximum	Mean	Standard Deviation
1	3.812	3.812	3.812	3.812	0.000
2	2.586	2.586	2.586	2.586	8.638×10^{-7}
3	2.222	2.222	2.222	2.222	7.199×10^{-5}
4	2.191	2.191	2.192	2.191	1.108×10^{-4}
5	2.099	2.102	2.106	2.102	1.262×10^{-3}
6	2.099	2.102	2.106	2.102	1.396×10^{-3}

Table A.6: **Problem 2 distributed results (VSWR).**

Solution Length	Minimum	Median	Maximum	Mean	Standard Deviation
1	3.813	3.813	3.813	3.813	0.000
2	2.637	2.642	2.658	2.643	4.320×10^{-3}
3	2.197	2.340	2.433	2.316	7.717×10^{-2}
4	2.130	2.244	2.344	2.246	4.072×10^{-2}
5	2.138	2.184	2.268	2.188	2.851×10^{-2}
6	2.089	2.150	2.218	2.154	2.102×10^{-2}

Table A.7: Problem 2 mixed lumped-distributed results (VSWR).

Solution Length	Minimum	Median	Maximum	Mean	Standard Deviation
1	3.813	3.813	3.813	3.813	0.000
2	2.586	2.586	2.607	2.587	2.078×10^{-3}
3	2.189	2.222	2.587	2.251	8.575×10^{-2}
4	2.159	2.193	2.202	2.193	4.367×10^{-3}
5	2.053	2.118	2.201	2.134	3.652×10^{-2}
6	2.034	2.114	2.190	2.118	2.499×10^{-2}

Table A.8: Problem 2 microstrip results (VSWR).

Solution Length	Minimum	Median	Maximum	Mean	Standard Deviation
1	4.227 / 4.393	4.227 / 4.393	4.227 / 4.393	4.227	1.000×10^{-6}
2	2.199 / 7.334	2.433 / 16.35	3.300 / 52.51	2.478	2.120×10^{-2}
3	2.062 / 5.122	2.093 / 10.41	2.327 / 964.4	2.109	4.497×10^{-3}
4	1.861 / 4.123	2.065 / 11.04	2.140 / 109.6	2.051	4.998×10^{-3}
5	1.658 / 5.968	1.897 / 15.02	2.037 / 471.4	1.877	7.229×10^{-3}
6	1.433 / 6.202	1.654 / 13.81	1.934 / 374.7	1.648	8.432×10^{-3}

Table A.9: Problem 3 lumped results (VSWR).

Solution Length	Minimum	Median	Maximum	Mean	Standard Deviation
1	2.500	2.500	2.500	2.500	0.000
2	2.024	2.024	2.024	2.024	2.721×10^{-6}
3	1.676	1.676	2.024	1.680	3.473×10^{-2}
4	1.670	1.670	2.024	1.674	3.533×10^{-2}
5	1.586	1.586	2.024	1.591	4.374×10^{-2}
6	1.586	1.586	2.024	1.591	4.373×10^{-2}

Table A.10: Problem 3 distributed results (VSWR).

Solution Length	Minimum	Median	Maximum	Mean	Standard Deviation
1	2.500	2.500	2.500	2.500	0.000
2	2.107	2.112	2.126	2.113	4.363×10^{-3}
3	1.779	1.848	2.020	1.855	4.341×10^{-2}
4	1.740	1.809	1.897	1.810	3.204×10^{-2}
5	1.742	1.787	1.861	1.790	2.688×10^{-2}
6	1.707	1.762	1.802	1.761	1.795×10^{-2}

Table A.11: Problem 3 mixed lumped-distributed results (VSWR).

Solution Length	Minimum	Median	Maximum	Mean	Standard Deviation
1	2.500	2.500	2.500	2.500	0.000
2	2.024	2.024	2.025	2.024	1.811×10^{-4}
3	1.676	1.707	2.024	1.727	7.873×10^{-2}
4	1.670	1.692	1.815	1.696	2.586×10^{-2}
5	1.586	1.680	1.795	1.665	4.934×10^{-2}
6	1.586	1.674	1.784	1.661	4.749×10^{-2}

Table A.12: Problem 3 microstrip results (VSWR).

Solution Length	Minimum	Median	Maximum	Mean	Standard Deviation
1	2.575 / 2.706	2.575 / 2.715	2.577 / 2.720	2.576	2.321×10^{-4}
2	1.308 / 2.514	1.643 / 9.728	2.293 / 33.74	1.680	2.171×10^{-2}
3	1.230 / 2.811	1.467 / 13.29	1.883 / 82.14	1.466	1.121×10^{-2}
4	1.105 / 4.788	1.264 / 8.664	1.541 / 595.2	1.273	9.473×10^{-3}
5	1.064 / 4.161	1.220 / 12.78	1.414 / 129.2	1.221	7.225×10^{-3}
6	1.047 / 4.458	1.174 / 46.34	1.357 / 992.2	1.179	5.658×10^{-3}

Table A.13: Problem 4 lumped results (VSWR).

Solution Length	Minimum	Median	Maximum	Mean	Standard Deviation
1	24.47	24.47	24.47	24.47	0.000
2	5.940	5.940	5.940	5.940	3.744×10^{-5}
3	4.505	4.505	4.507	4.505	3.230×10^{-4}
4	2.987	2.988	2.996	2.989	1.867×10^{-3}
5	2.929	2.940	2.988	2.944	1.519×10^{-2}
6	2.850	2.932	2.988	2.935	2.119×10^{-2}

Table A.14: Problem 4 distributed results (VSWR).

Solution Length	Minimum	Median	Maximum	Mean	Standard Deviation
1	8.487	8.488	8.492	8.488	9.110×10^{-4}
2	5.979	6.026	6.690	6.057	9.953×10^{-2}
3	3.902	5.229	5.912	5.151	0.4285
4	3.372	4.335	5.285	4.322	0.3991
5	3.064	4.166	4.467	4.074	0.3545
6	1.478	2.387	3.892	2.457	0.4206

Table A.15: Problem 4 mixed lumped-distributed results (VSWR).

Solution Length	Minimum	Median	Maximum	Mean	Standard Deviation
1	8.487	8.491	8.616	8.497	1.929×10^{-2}
2	5.938	5.942	6.401	5.982	8.439×10^{-2}
3	3.912	4.505	5.251	4.385	0.2376
4	2.988	3.075	4.472	3.335	0.4096
5	2.653	2.997	3.903	3.084	0.2039
6	2.455	2.980	3.430	2.952	0.1710

Table A.16: Problem 4 microstrip results (VSWR).

Solution Length	Minimum	Median	Maximum	Mean	Standard Deviation
1	8.493 / 8.628	8.496 / 8.674	8.563 / 8.691	8.498	8.134×10^{-3}
2	3.759 / 10.97	4.059 / 154.7	4.673 / 825.0	4.076	0.2026
3	2.337 / 23.55	3.448 / 171.1	4.441 / 955.1	3.480	0.4811
4	1.607 / 13.11	2.463 / 190.8	3.610 / 816.7	2.476	0.4472
5	1.602 / 47.73	2.015 / 210.6	2.773 / 804.9	2.041	0.2467
6	1.269 / 49.83	1.708 / 251.9	2.324 / 952.7	1.713	0.2475

Table A.17: Problem 5 lumped results (decibel error).

Solution Length	Minimum	Median	Maximum	Mean	Standard Deviation
1	1.004	1.004	1.004	1.004	0.000
2	0.1942	0.1945	0.6953	0.2663	0.1738
3	8.508×10^{-2}	0.1441	0.1953	0.1521	2.948×10^{-2}
4	4.538×10^{-2}	0.1302	0.1589	0.1136	3.188×10^{-2}
5	4.401×10^{-2}	0.1213	0.1481	0.1061	3.358×10^{-2}
6	4.210×10^{-2}	0.1171	0.1453	0.1042	3.380×10^{-2}

Table A.18: Problem 5 distributed results (decibel error).

Solution Length	Minimum	Median	Maximum	Mean	Standard Deviation
1	1.023	1.029	1.050	1.031	6.857×10^{-3}
2	0.1003	0.1144	0.1878	0.1250	2.479×10^{-2}
3	8.192×10^{-2}	0.1046	0.1353	0.1048	1.126×10^{-2}
4	8.311×10^{-2}	9.830×10^{-2}	0.1259	9.988×10^{-2}	9.400×10^{-3}
5	8.072×10^{-2}	9.538×10^{-2}	0.1185	9.617×10^{-2}	7.095×10^{-3}
6	7.231×10^{-2}	9.062×10^{-2}	0.1033	9.112×10^{-2}	5.506×10^{-3}

Table A.19: Problem 5 mixed lumped-distributed results (decibel error).

Solution Length	Minimum	Median	Maximum	Mean	Standard Deviation
1	1.004	1.004	1.004	1.004	0.000
2	0.1003	0.1397	0.2199	0.1395	3.337×10^{-2}
3	8.796×10^{-2}	0.1021	0.1336	0.1039	1.050×10^{-2}
4	8.504×10^{-2}	9.675×10^{-2}	0.1200	9.776×10^{-2}	7.347×10^{-3}
5	7.333×10^{-2}	9.140×10^{-2}	0.1157	9.257×10^{-2}	6.243×10^{-3}
6	3.970×10^{-2}	8.914×10^{-2}	0.1062	8.892×10^{-2}	7.373×10^{-3}

Table A.20: Problem 5 microstrip results (decibel error).

Solution Length	Minimum	Median	Maximum	Mean	Standard Deviation
1	1.082 / 1.103	1.090 / 1.110	1.281 / 1.136	1.100	2.956×10^{-2}
2	0.0963 / 0.1220	0.1194 / 0.1502	0.1885 / 0.1903	0.1285	2.710×10^{-2}
3	0.0880 / 0.1089	0.1105 / 0.1285	0.1397 / 0.1565	0.1126	1.172×10^{-2}
4	0.0881 / 0.1046	0.1047 / 0.1163	0.1352 / 0.1636	0.1053	9.110×10^{-3}
5	0.0870 / 0.1030	0.1023 / 0.1162	0.1268 / 0.1448	0.1027	8.643×10^{-3}
6	0.0846 / 0.1017	0.0978 / 0.1150	0.1278 / 0.1336	0.0983	7.459×10^{-3}

Table A.21: Problem 6 lumped results (decibel error).

Solution Length	Minimum	Median	Maximum	Mean	Standard Deviation
1	1.012	1.012	1.012	1.012	0.000
2	0.6629	0.6629	0.6629	0.6629	1.191×10^{-6}
3	0.2688	0.2694	0.2964	0.2717	5.565×10^{-3}
4	0.1922	0.1942	0.2380	0.1966	7.565×10^{-3}
5	7.960×10^{-2}	0.1750	0.2046	0.1577	3.260×10^{-2}
6	7.366×10^{-2}	0.1229	0.1929	0.1237	2.587×10^{-2}

Table A.22: Problem 6 distributed results (decibel error).

Solution Length	Minimum	Median	Maximum	Mean	Standard Deviation
1	1.257	1.257	1.257	1.257	1.000×10^{-6}
2	0.6641	0.6682	0.6826	0.6690	3.643×10^{-3}
3	0.1990	0.2723	0.4101	0.2751	4.358×10^{-2}
4	6.513×10^{-2}	0.2289	0.3273	0.2297	4.484×10^{-2}
5	1.965×10^{-2}	8.883×10^{-2}	0.2402	9.854×10^{-2}	4.949×10^{-2}
6	4.916×10^{-3}	3.531×10^{-2}	0.1527	4.237×10^{-2}	2.629×10^{-2}

Table A.23: Problem 6 mixed lumped-distributed results (decibel error).

Solution Length	Minimum	Median	Maximum	Mean	Standard Deviation
1	1.012	1.012	1.012	1.012	0.000
2	0.3054	0.3344	0.6629	0.4312	0.1586
3	9.332×10^{-2}	0.2335	0.4957	0.2586	0.1351
4	5.942×10^{-2}	0.1524	0.2288	0.1544	4.813×10^{-2}
5	4.462×10^{-2}	0.1424	0.1983	0.1344	4.560×10^{-2}
6	2.173×10^{-2}	0.1123	0.1978	0.1155	4.694×10^{-2}

Table A.24: Problem 6 microstrip results (decibel error).

Solution Length	Minimum	Median	Maximum	Mean	Standard Deviation
1	1.546 / 1.535	1.546 / 1.536	1.578 / 1.541	1.548	4.829×10^{-3}
2	1.133 / 1.148	1.171 / 1.181	1.221 / 1.220	1.174	2.361×10^{-2}
3	0.3284 / 0.3726	0.4540 / 0.4700	1.0224 / 0.5806	0.4590	9.498×10^{-2}
4	0.2302 / 0.2855	0.3165 / 0.3578	0.4621 / 0.4773	0.3244	4.733×10^{-2}
5	0.0174 / 0.0870	0.1028 / 0.1420	0.3098 / 0.3288	0.1123	4.826×10^{-2}
6	0.0080 / 0.0183	0.0398 / 0.0721	0.1181 / 0.2135	0.0438	2.340×10^{-2}

Table A.25: Problem 7 lumped results (decibel error).

Solution Length	Minimum	Median	Maximum	Mean	Standard Deviation
1	0.9710	0.9710	0.9710	0.9710	0.000
2	0.9164	0.9164	0.9165	0.9164	1.609×10^{-5}
3	0.6282	0.6327	0.8579	0.6702	6.543×10^{-2}
4	0.3194	0.4859	0.7937	0.4967	9.413×10^{-2}
5	0.3166	0.4611	0.7700	0.4719	9.164×10^{-2}
6	0.3014	0.4581	0.7835	0.4530	9.235×10^{-2}

Table A.26: Problem 7 distributed results (decibel error).

Solution Length	Minimum	Median	Maximum	Mean	Standard Deviation
1	0.9752	0.9752	0.9776	0.9753	2.414×10^{-4}
2	0.5977	0.9490	0.9700	0.8743	0.1141
3	0.2673	0.3268	0.4411	0.3341	3.517×10^{-2}
4	0.2560	0.2955	0.3988	0.2983	2.380×10^{-2}
5	0.2513	0.2783	0.3225	0.2806	1.434×10^{-2}
6	0.2214	0.2625	0.2891	0.2608	1.060×10^{-2}

Table A.27: Problem 7 mixed lumped-distributed results (decibel error).

Solution Length	Minimum	Median	Maximum	Mean	Standard Deviation
1	0.9710	0.9710	0.9710	0.9710	0.000
2	0.6491	0.9119	0.9202	0.9087	3.082×10^{-2}
3	0.2791	0.3642	0.5018	0.3700	4.651×10^{-2}
4	0.2608	0.3074	0.4022	0.3074	2.688×10^{-2}
5	0.2442	0.2861	0.3390	0.2879	1.785×10^{-2}
6	0.2408	0.2770	0.3269	0.2785	1.746×10^{-2}

Table A.28: Problem 7 microstrip results (decibel error).

Solution Length	Minimum	Median	Maximum	Mean	Standard Deviation
1	1.543 / 1.575	1.543 / 1.576	1.543 / 1.578	1.543	8.492×10^{-5}
2	0.3879 / 0.3993	0.4260 / 0.4366	0.4872 / 0.4594	0.4257	1.983×10^{-2}
3	0.2636 / 0.2942	0.3263 / 0.3586	0.3994 / 0.4356	0.3261	3.423×10^{-2}
4	0.2496 / 0.2703	0.2886 / 0.3076	0.3462 / 0.4394	0.2902	2.083×10^{-2}
5	0.2474 / 0.2659	0.2771 / 0.3062	0.3270 / 0.3467	0.2786	1.687×10^{-2}
6	0.2041 / 0.2277	0.2558 / 0.2835	0.2778 / 0.3398	0.2547	1.140×10^{-2}

Table A.29: Problem 8 lumped results (decibel error).

Solution Length	Minimum	Median	Maximum	Mean	Standard Deviation
1	1.539	1.539	1.539	1.539	0.000
2	0.6429	0.6429	0.6429	0.6429	9.326×10^{-7}
3	0.5618	0.5618	0.5693	0.5622	1.013×10^{-3}
4	0.2990	0.3794	0.5611	0.3921	6.714×10^{-2}
5	0.2490	0.3367	0.4802	0.3374	3.945×10^{-2}
6	0.2449	0.3005	0.4640	0.3107	3.470×10^{-2}

Table A.30: Problem 8 distributed results (decibel error).

Solution Length	Minimum	Median	Maximum	Mean	Standard Deviation
1	1.155	1.155	1.204	1.160	9.891×10^{-3}
2	0.4911	0.4972	0.5069	0.4975	2.978×10^{-3}
3	0.2793	0.4905	0.5002	0.4515	5.681×10^{-2}
4	0.1493	0.3512	0.4953	0.3481	7.620×10^{-2}
5	0.1137	0.2471	0.3700	0.2497	6.000×10^{-2}
6	6.850×10^{-2}	0.1841	0.3166	0.1831	4.855×10^{-2}

Table A.31: Problem 8 mixed lumped-distributed results (decibel error).

Solution Length	Minimum	Median	Maximum	Mean	Standard Deviation
1	1.155	1.159	1.245	1.167	1.752×10^{-2}
2	0.4426	0.4991	0.5617	0.4967	2.021×10^{-2}
3	0.2381	0.4520	0.5026	0.4223	7.490×10^{-2}
4	0.1580	0.3494	0.4975	0.3520	8.718×10^{-2}
5	0.1179	0.2907	0.4808	0.2847	8.497×10^{-2}
6	9.805×10^{-2}	0.2451	0.4516	0.2522	8.230×10^{-2}

Table A.32: Problem 8 microstrip results (decibel error).

Solution Length	Minimum	Median	Maximum	Mean	Standard Deviation
1	1.683 / 1.687	1.683 / 1.687	1.800 / 1.763	1.690	2.134×10^{-2}
2	0.5281 / 0.5349	0.7159 / 0.7713	0.9223 / 0.8995	0.7157	6.611×10^{-2}
3	0.3673 / 0.5021	0.6094 / 0.6583	0.7139 / 0.8521	0.5884	6.924×10^{-2}
4	0.2419 / 0.3630	0.4817 / 0.5211	0.6227 / 0.7000	0.4675	0.1026
5	0.1210 / 0.2751	0.3281 / 0.3889	0.5654 / 0.6801	0.3380	8.827×10^{-2}
6	0.0967 / 0.2389	0.2304 / 0.3059	0.4620 / 0.5000	0.2324	6.538×10^{-2}

Table A.33: Problem 9 lumped results (VSWR).

Solution Length	Minimum	Median	Maximum	Mean	Standard Deviation
1	39.80	39.80	39.80	39.80	0.000
2	2.889	2.889	2.890	2.889	3.441×10^{-5}
3	2.617	2.617	2.834	2.622	2.849×10^{-2}
4	1.214	1.268	1.333	1.257	2.861×10^{-2}
5	1.096	1.171	1.259	1.178	2.900×10^{-2}
6	1.074	1.135	1.236	1.136	3.713×10^{-2}

Table A.34: Problem 9 distributed results (VSWR).

Solution Length	Minimum	Median	Maximum	Mean	Standard Deviation
1	2.588	2.590	2.657	2.595	1.168×10^{-2}
2	1.823	1.895	2.397	1.926	0.1025
3	1.634	1.749	1.884	1.754	4.245×10^{-2}
4	1.377	1.619	1.845	1.611	8.635×10^{-2}
5	1.253	1.440	1.668	1.440	7.702×10^{-2}
6	1.162	1.372	1.538	1.371	6.435×10^{-2}

Table A.35: Problem 9 mixed lumped-distributed results (VSWR).

Solution Length	Minimum	Median	Maximum	Mean	Standard Deviation
1	2.588	2.591	2.647	2.595	1.009×10^{-2}
2	1.820	1.945	2.470	1.980	0.1481
3	1.259	1.484	1.867	1.530	0.1838
4	1.136	1.331	1.702	1.349	0.1053
5	1.125	1.261	1.532	1.267	7.888×10^{-2}
6	1.109	1.213	1.408	1.224	6.576×10^{-2}

Table A.36: Problem 9 microstrip results (VSWR).

Solution Length	Minimum	Median	Maximum	Mean	Standard Deviation
1	5.962 / 6.316	5.963 / 6.361	6.145 / 6.406	5.966	1.925×10^{-2}
2	5.962 / 6.223	5.988 / 6.362	6.510 / 6.492	6.012	7.058×10^{-2}
3	3.026 / 4.378	4.683 / 4.962	5.484 / 5.754	4.685	0.3658
4	2.128 / 3.987	3.468 / 11.26	4.100 / 682.8	3.420	0.4272
5	2.051 / 3.099	3.023 / 10.81	3.784 / 266.5	3.044	0.3582
6	1.576 / 3.050	2.662 / 18.83	3.498 / 460.2	2.648	0.4027

Table A.37: Problem 10 lumped results (VSWR).

Solution Length	Minimum	Median	Maximum	Mean	Standard Deviation
1	2.513	2.513	2.513	2.513	0.000
2	2.225	2.225	2.354	2.231	2.430×10^{-2}
3	1.467	1.557	2.063	1.590	9.351×10^{-2}
4	1.403	1.423	1.773	1.451	6.378×10^{-2}
5	1.403	1.408	1.607	1.416	2.473×10^{-2}
6	1.403	1.407	1.481	1.410	1.135×10^{-2}

Table A.38: Problem 10 distributed results (VSWR).

Solution Length	Minimum	Median	Maximum	Mean	Standard Deviation
1	2.319	2.319	2.319	2.319	7.145×10^{-5}
2	1.719	1.835	1.972	1.837	6.682×10^{-2}
3	1.427	1.551	1.711	1.554	6.313×10^{-2}
4	1.405	1.451	1.513	1.455	2.549×10^{-2}
5	1.370	1.431	1.470	1.431	1.672×10^{-2}
6	1.354	1.424	1.444	1.419	1.883×10^{-2}

Table A.39: Problem 10 mixed lumped-distributed results (VSWR).

Solution Length	Minimum	Median	Maximum	Mean	Standard Deviation
1	2.319	2.319	2.335	2.320	2.830×10^{-3}
2	1.701	1.832	1.989	1.842	8.269×10^{-2}
3	1.416	1.592	1.816	1.590	8.089×10^{-2}
4	1.404	1.460	1.563	1.466	3.523×10^{-2}
5	1.318	1.435	1.476	1.435	1.936×10^{-2}
6	1.288	1.429	1.461	1.426	1.951×10^{-2}

Table A.40: Problem 10 microstrip results (VSWR).

Solution Length	Minimum	Median	Maximum	Mean	Standard Deviation
1	2.319 / 2.322	2.319 / 2.322	2.319 / 2.322	2.319	5.171×10^{-5}
2	1.943 / 1.953	1.976 / 1.977	2.104 / 2.036	1.987	3.334×10^{-2}
3	1.604 / 1.680	1.807 / 1.832	1.964 / 1.987	1.799	7.480×10^{-2}
4	1.438 / 1.451	1.576 / 1.605	1.714 / 1.689	1.570	6.744×10^{-2}
5	1.425 / 1.429	1.490 / 1.513	1.593 / 1.578	1.496	4.093×10^{-2}
6	1.412 / 1.431	1.466 / 1.478	1.526 / 1.520	1.466	2.267×10^{-2}

Appendix B

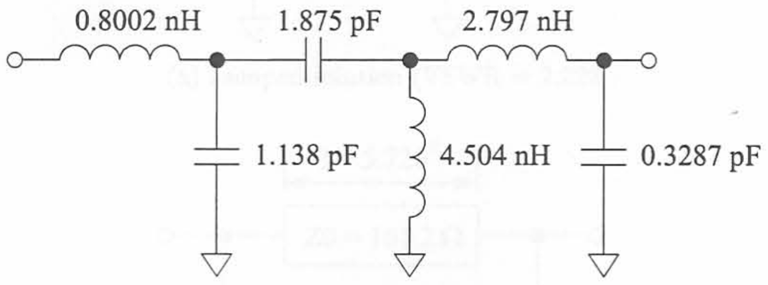
Best Circuits

The best circuits obtained are given below. The number of elements used in the best circuits is the same as published results where available, and produces a VSWR of less than 1.5 or a gain error of less than 0.1 dB where published results are not available. Six element solutions are given where this is not possible. The only exceptions are the distributed solutions to Problems 2 and 3, the mixed solution to Problems 2, 3 and 4, and the lumped and mixed solutions to Problem 7, where adding extra elements has a small ($< 5\%$) effect. All circuits are drawn with the source on the left and the load on the right.

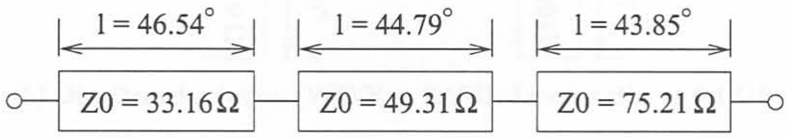
The component ranges used are given in Table 3.3 on page 87, and the microstrip substrates used are given in Table 3.4 on page 87. The transmission line elements in the distributed results given below do not use the default parameters given in Table 3.3, but rather have the same parameters as the microstrip lines to allow comparisons between the ideal case and the case where dispersion and discontinuities are accounted for. The microstrip parameters were converted to characteristic impedances and line lengths using the low-frequency equations given in Sections 2.4.1 and 2.4.2 on pages 66 and 68. The ratios of the widths of microstrip lines at a discontinuity was limited to a maximum value of 5, and crosses were not allowed to ensure that the discontinuity calculations are accurate.

The mixed results do not consider discontinuities and dispersion for the transmission lines, and solder pads for the lumped components. These effects could have been included, but the inclusion of these effects would have dramatically increased the time to run the algorithm to closer to the time required for microstrip tests (see Section 4.4) without greatly adding to the value of the results.

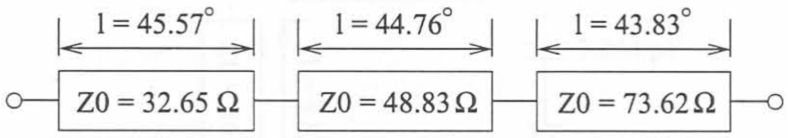
The microstrip networks can start with a parallel element because the algorithm assumes that all microstrip networks are bounded by 50 Ω transmission lines. This is necessary to complete the discontinuities (width steps or T-junctions) at the extremities of the networks.



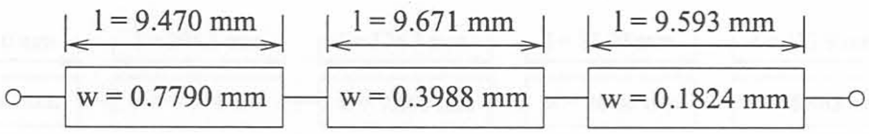
(a) Lumped solution (VSWR = 1.180).



(b) Distributed solution (VSWR = 1.242). Line lengths at 2 GHz.

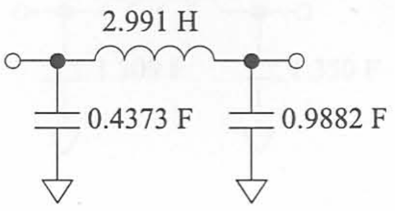


(c) Mixed solution (VSWR = 1.245). Line lengths at 2 GHz.

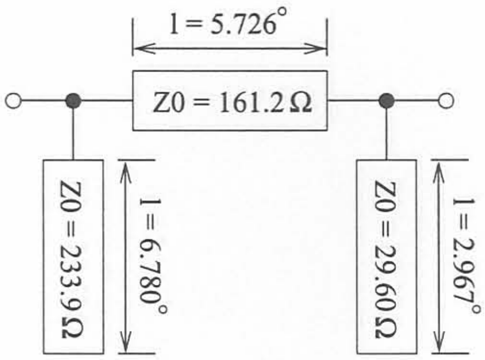


(d) Microstrip solution with $h = 0.25$ mm and $\varepsilon_r = 5$ (VSWR = 1.245).

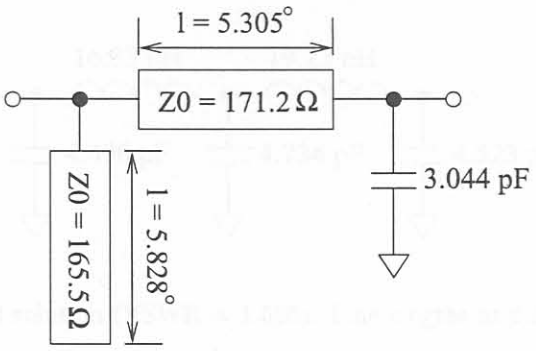
Figure B.1: Test Problem 1 solutions.



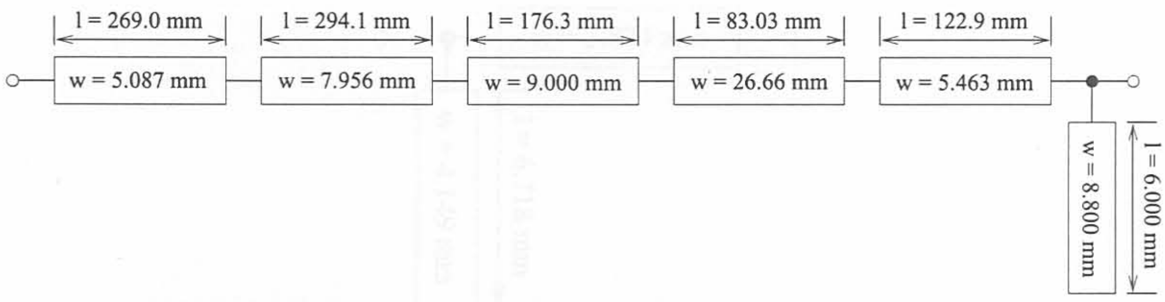
(a) Lumped solution (VSWR = 2.222).



(b) Distributed solution (VSWR = 2.189). Line lengths at 0.1 GHz.

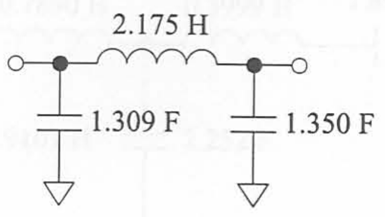


(c) Mixed solution (VSWR = 2.099). Line lengths at 0.1 GHz.

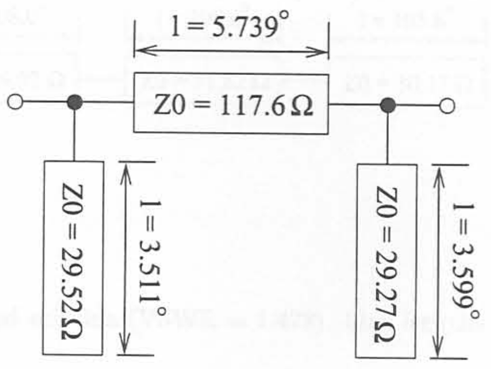


(d) Microstrip solution with h = 3 mm and $\varepsilon_r = 1$ (VSWR = 1.433).

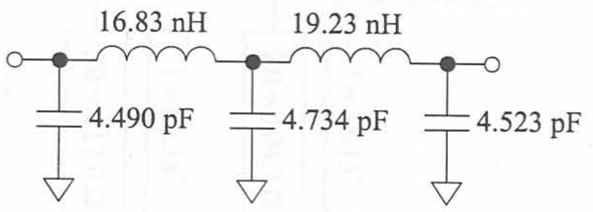
Figure B.2: Test Problem 2 solutions.



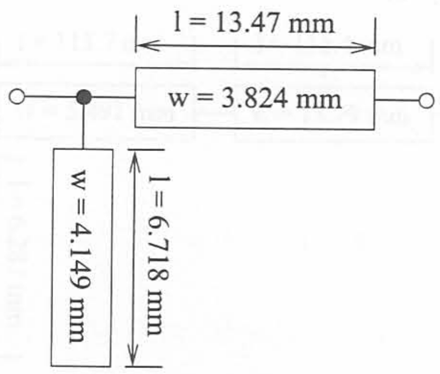
(a) Lumped solution (VSWR = 1.676).



(b) Distributed solution (VSWR = 1.779). Line lengths at 0.1 GHz.

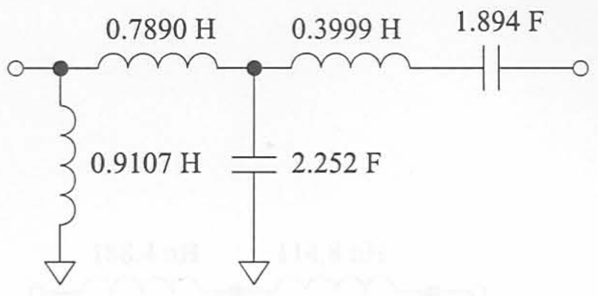


(c) Mixed solution (VSWR = 1.586). Line lengths at 0.1 GHz.

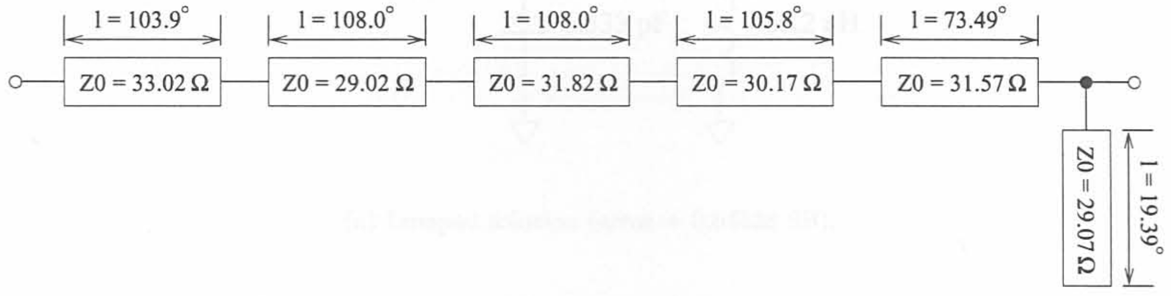


(d) Microstrip solution with $h = 3$ mm and $\varepsilon_r = 1$ (VSWR = 1.308).

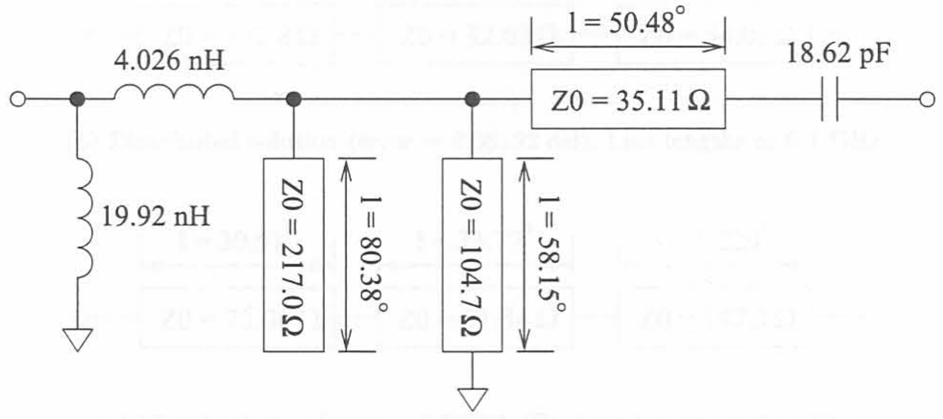
Figure B.3: Test Problem 3 solutions.



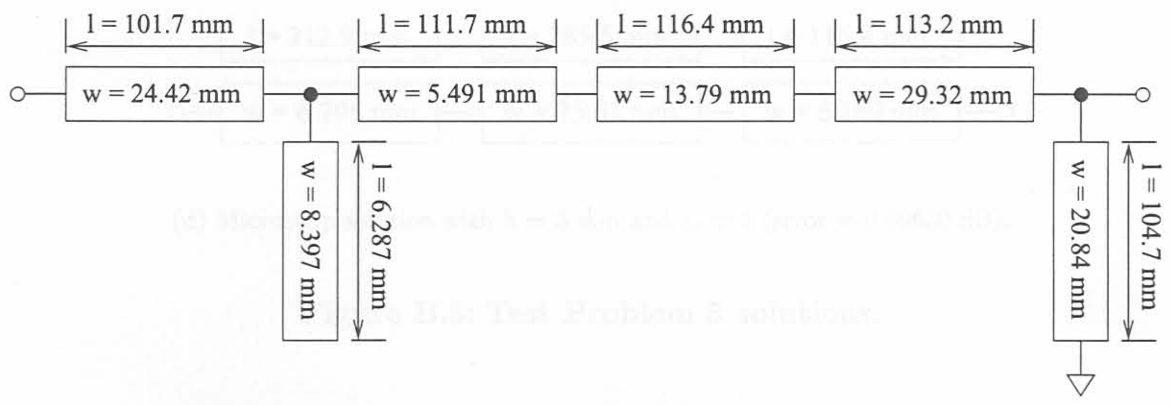
(a) Lumped solution (VSWR = 2.929).



(b) Distributed solution (VSWR = 1.478). Line lengths at 0.3 GHz.

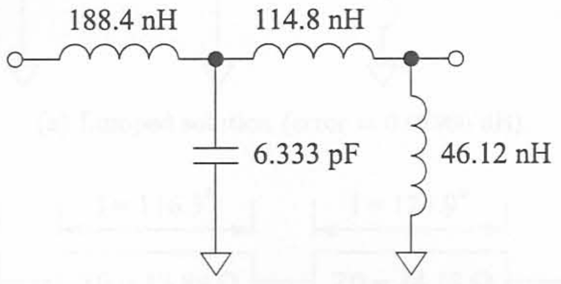


(c) Mixed solution (VSWR = 2.455). Line lengths at 0.3 GHz.

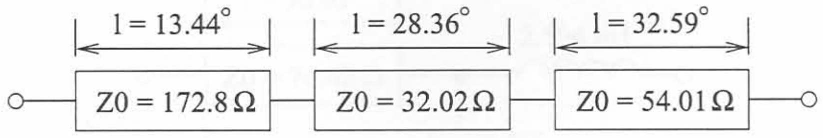


(d) Microstrip solution with $h = 3$ mm and $\varepsilon_r = 1$ (VSWR = 1.269).

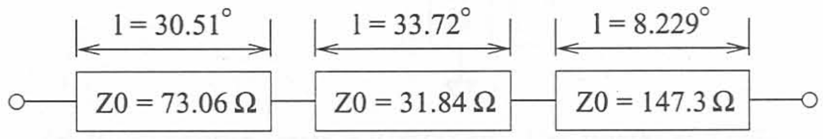
Figure B.4: Test Problem 4 solutions.



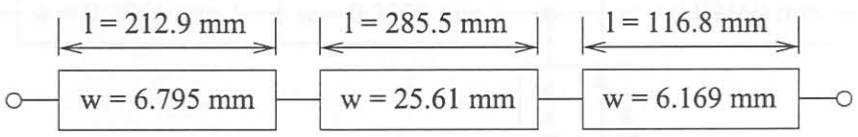
(a) Lumped solution (error = 0.04538 dB).



(b) Distributed solution (error = 0.08192 dB). Line lengths at 0.1 GHz.

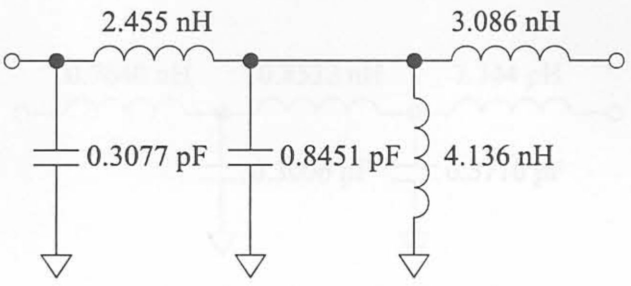


(c) Mixed solution (error = 0.08796 dB). Line lengths at 0.1 GHz.

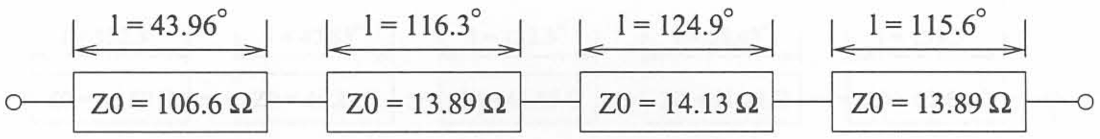


(d) Microstrip solution with $h = 3$ mm and $\varepsilon_r = 1$ (error = 0.09630 dB).

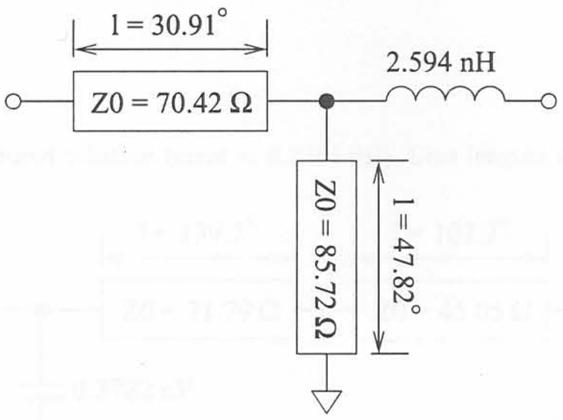
Figure B.5: Test Problem 5 solutions.



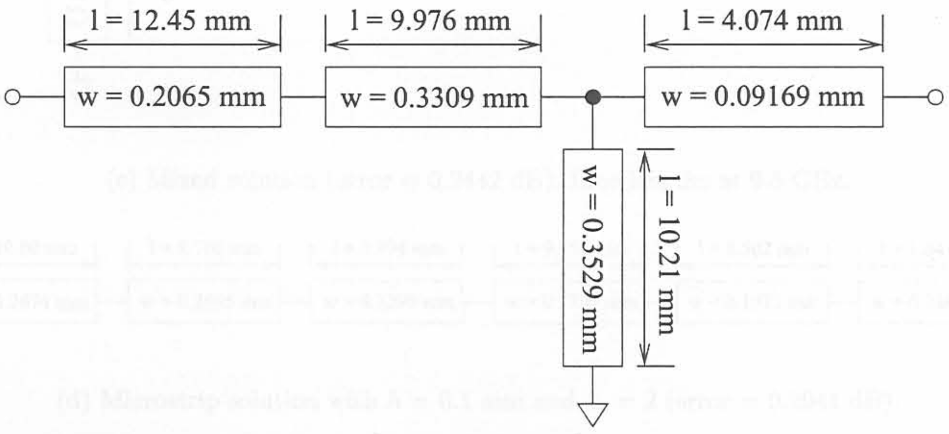
(a) Lumped solution (error = 0.07960 dB).



(b) Distributed solution (error = 0.06513 dB). Line lengths at 2 GHz.

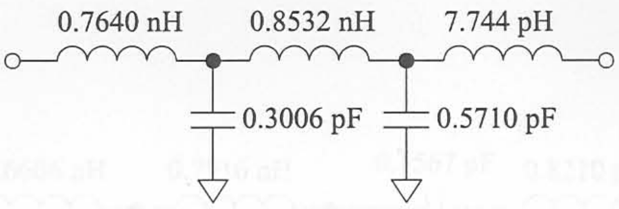


(c) Mixed solution (error = 0.09332 dB). Line lengths at 2 GHz.

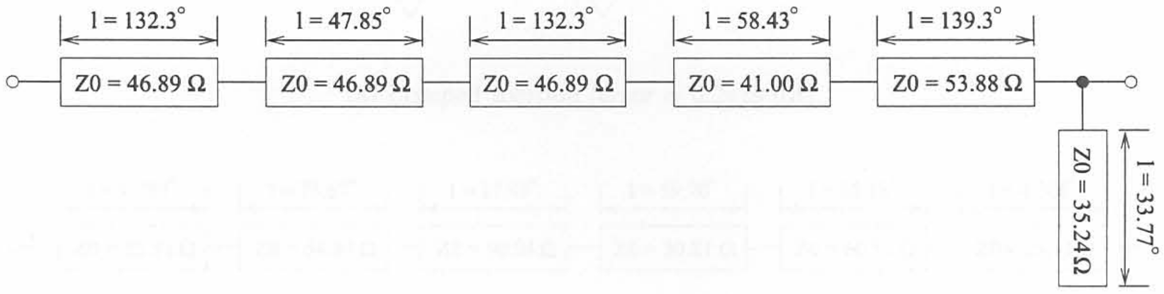


(d) Microstrip solution with $h = 0.25$ mm and $\varepsilon_r = 5$ (error = 0.01738 dB).

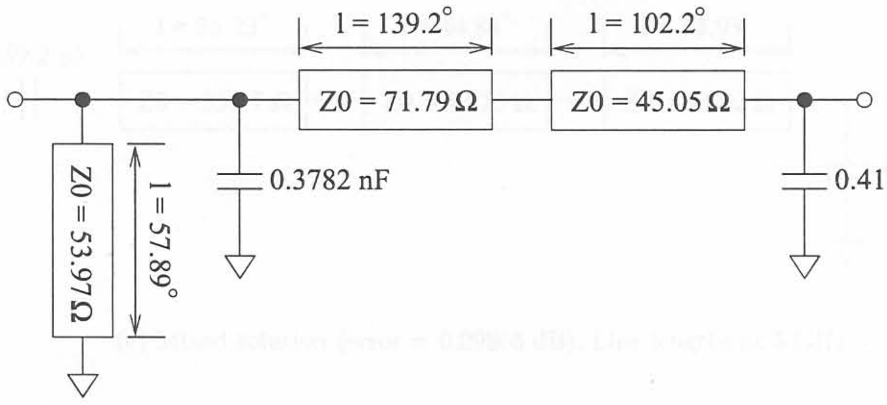
Figure B.6: Test Problem 6 solutions.



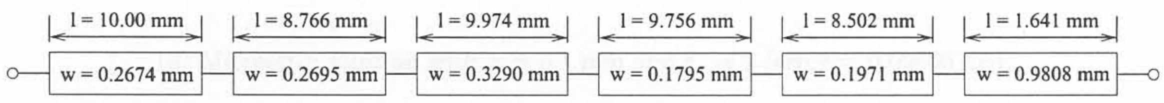
(a) Lumped solution (error = 0.3166 dB).



(b) Distributed solution (error = 0.2214 dB). Line lengths at 9.5 GHz.

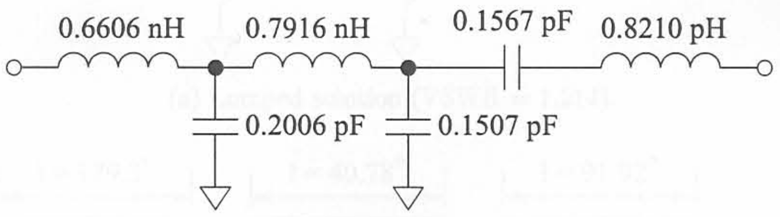


(c) Mixed solution (error = 0.2442 dB). Line lengths at 9.5 GHz.

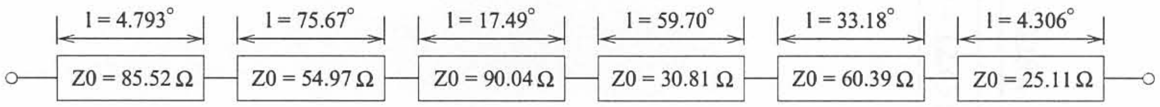


(d) Microstrip solution with $h = 0.1$ mm and $\varepsilon_r = 2$ (error = 0.2041 dB).

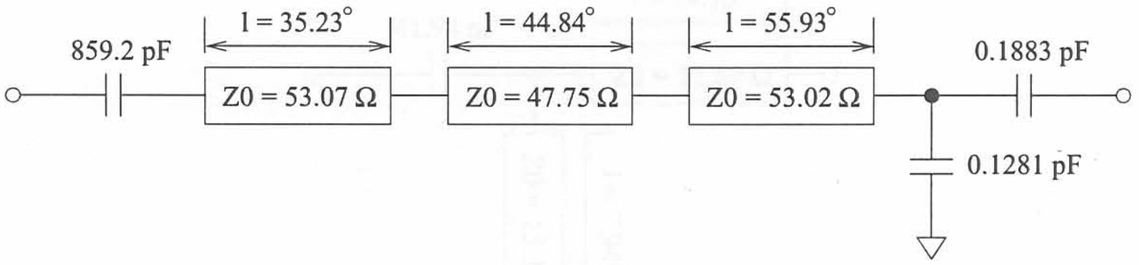
Figure B.7: Test Problem 7 solutions.



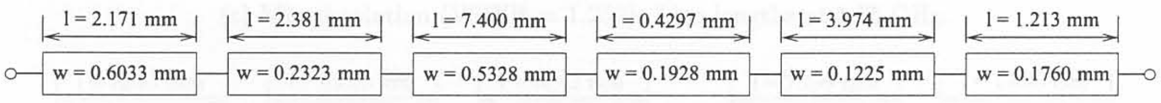
(a) Lumped solution (error = 0.2449 dB).



(b) Distributed solution (error = 0.06858 dB). Line lengths at 5 GHz.

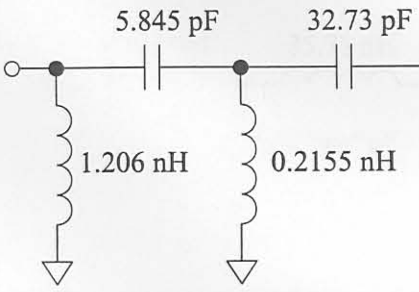


(c) Mixed solution (error = 0.09805 dB). Line lengths at 5 GHz.

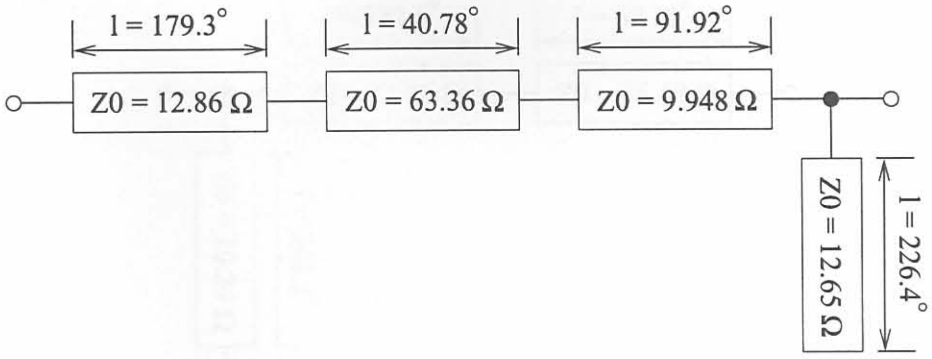


(d) Microstrip solution with $h = 0.1$ mm and $\varepsilon_r = 2$ (error = 0.09666 dB).

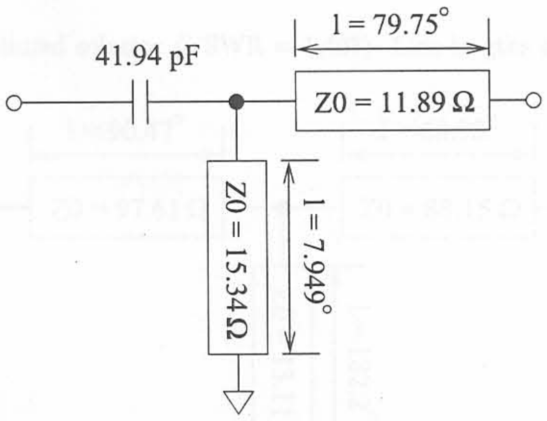
Figure B.8: Test Problem 8 solutions.



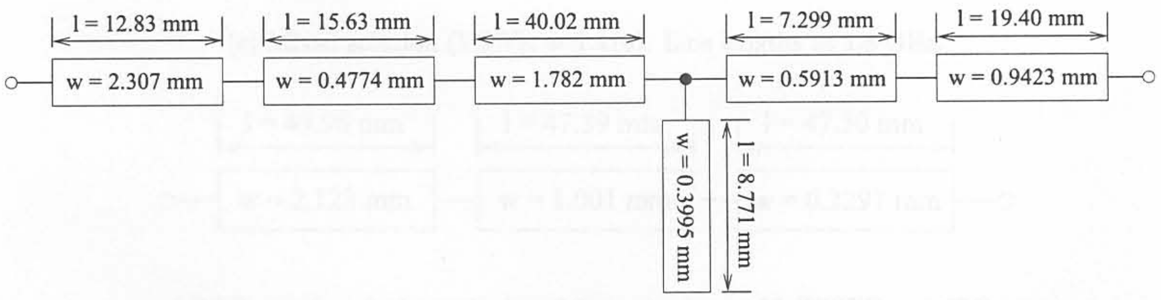
(a) Lumped solution (VSWR = 1.214).



(b) Distributed solution (VSWR = 1.377). Line lengths at 1.75 GHz.

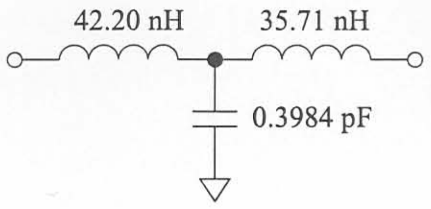


(c) Mixed solution (VSWR = 1.259). Line lengths at 1.75 GHz.

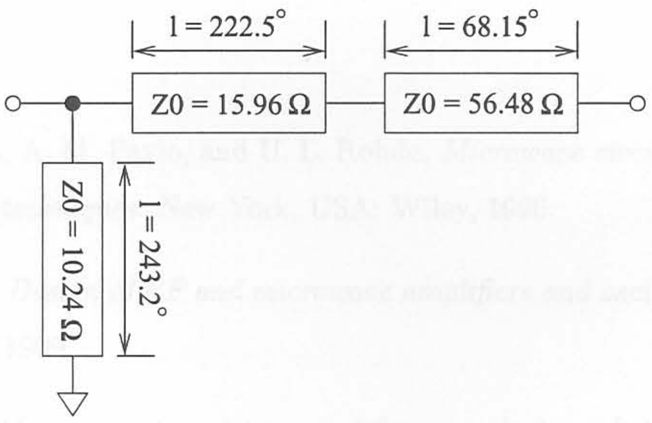


(d) Microstrip solution with $h = 0.5$ mm and $\varepsilon_r = 10$ (VSWR = 1.576).

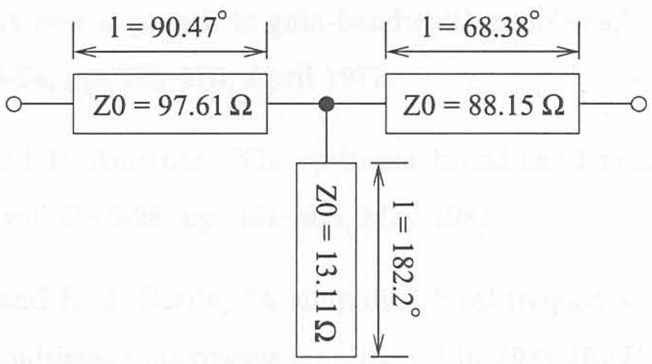
Figure B.9: Test Problem 9 solutions.



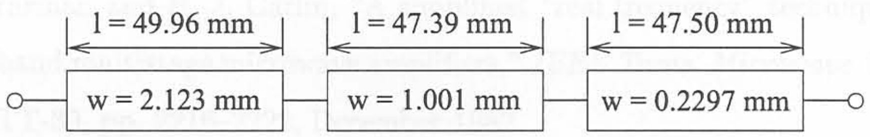
(a) Lumped solution (VSWR = 1.467).



(b) Distributed solution (VSWR = 1.427). Line lengths at 1.8 GHz.



(c) Mixed solution (VSWR = 1.416). Line lengths at 1.8 GHz.



(d) Microstrip solution with $h = 0.5$ mm and $\varepsilon_r = 10$ (VSWR = 1.604).

Figure B.10: Test Problem 10 solutions.

Bibliography

[1] G. D. Vendelin, A. M. Pavio, and U. L. Rohde, *Microwave circuit design using linear and nonlinear techniques.* New York, USA: Wiley, 1990.

[2] P. L. D. Abrie, *Design of RF and microwave amplifiers and oscillators.* Boston, USA: Artech House, 1999.

[3] G. Gonzalez, *Microwave transistor amplifiers, analysis and design.* Upper Saddle River, USA: Prentice-Hall, second ed., 1997.

[4] H. J. Carlin, "A new approach to gain-bandwidth problems," *IEEE Trans. Circuits Syst.*, vol. CAS-24, pp. 170–175, April 1977.

[5] H. J. Carlin and P. Amstutz, "On optimum broad-band matching," *IEEE Trans. Circuits Syst.*, vol. CAS-28, pp. 401–405, May 1981.

[6] B. S. Yarman and H. J. Carlin, "A simplified "real frequency" technique applicable to broadband multistage microwave amplifiers," in *1982 IEEE MTT-S International Microwave Symposium Digest*, pp. 529–531, 1982.

[7] B. S. Yarman and H. J. Carlin, "A simplified "real frequency" technique applied to broad-band multistage microwave amplifiers," *IEEE Trans. Microwave Theory Tech.*, vol. MTT-30, pp. 2216–2222, December 1982.

[8] H. J. Carlin and B. S. Yarman, "The double matching problem: Analytic and real frequency solutions," *IEEE Trans. Circuits Syst.*, vol. CAS-30, pp. 15–28, January 1983.

[9] B. S. Yarman and A. Fettweis, "Computer-aided double matching via parametric representation of Brune functions," *IEEE Trans. Circuits Syst.*, vol. 37, pp. 212–222, February 1990.

[10] H. J. Carlin and P. P. Civalleri, "An algorithm for wideband matching using Wiener-Lee transforms," *IEEE Trans. Circuits Syst.-I: Fund. Theory Appl.*, vol. 39, pp. 497–505, July 1992.

[11] D. M. Pozar, *Microwave engineering*. New York, USA: Wiley, second ed., 1998.

[12] J. W. Bandler, "Optimization methods for computer-aided design," *IEEE Trans. Microwave Theory Tech.*, vol. MTT-17, pp. 533–552, August 1969.

[13] C. Charalambous, "A unified review of optimization," *IEEE Trans. Microwave Theory Tech.*, vol. MTT-22, pp. 289–300, March 1974.

[14] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical recipes in C: The art of scientific computing*. Melbourne, Australia: Cambridge University Press, second ed., 1994.

[15] J. Nocedal and S. J. Wright, *Numerical Optimization*. New-York, USA: SpringerVerlag, 1999.

[16] C. A. Borghi, M. Fabbri, P. D. Barba, and A. Savini, "Loney's solenoid multi-objective optimization problem," *IEEE Trans. Magn.*, vol. 35, pp. 1706–1709, May 1999.

[17] D. E. Goldberg, *Genetic algorithms in search, optimization, and machine learning*. New York, USA: Addison-Wesley, 1989.

[18] J.-S. R. Jang, C.-T. Sun, and E. Mizutani, *Neuro-fuzzy and soft computing: A computational approach to learning and machine intelligence*. Upper Saddle River, USA: Prentice-Hall, 1997.

[19] J. H. Mathews, *Numerical methods for mathematics, science, and engineering*. Englewood Cliffs, USA: Prentice-Hall, second ed., 1992.

[20] J. A. Nelder and R. Mead, "A simplex method for function minimization," *Computer J.*, vol. 7, pp. 308–313, January 1965.

[21] M. J. D. Powell, "Direct search algorithms for optimization calculations," *Acta Numerica*, vol. 7, pp. 287–336, 1998.

[22] J. A. Snyman, "A new and dynamic method for unconstrained minimization," *Appl. Math. Modelling*, vol. 6, pp. 449–462, December 1982.

[23] J. A. Snyman, "An improved version of the original leap-frog dynamic method for unconstrained minimization: LFOP1(b)," *Appl. Math. Modelling*, vol. 7, pp. 216–218, June 1983.

[24] J. A. Snyman, "The LFOPC leap-frog algorithm for constrained optimization," *Computers and Mathematics with Applications*, vol. 40, pp. 1085–1096, 2000.

[25] A. Törn and A. Zilinskas, *Global Optimization*. Heidelberg: Springer-Verlag, 1989.

[26] W. F. Smith, *Principles of materials science and engineering*. New York, USA: McGraw-Hill, second ed., 1990.

[27] K. A. Dowsland, "Simulated annealing," in *Modern heuristic techniques for combinatorial problems* (C. R. Reeves, ed.), ch. 2, pp. 20–69, Oxford, UK: Blackwell, 1993.

[28] A. Carlisle and G. Dozier, "An off-the-shelf PSO," in *Proceedings of the Workshop on Particle Swarm Optimization*, IUPUI. In Press.

[29] F. Glover and M. Laguna, "Tabu search," in *Modern heuristic techniques for combinatorial problems* (C. R. Reeves, ed.), ch. 3, pp. 70–150, Oxford, UK: Blackwell, 1993.

[30] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 67–82, 1997.

[31] A. A. Groenwold and M. P. Hindley, "Competing parallel algorithms in structural optimization," in *Proc. Fourth World Congress of Structural and Multidisciplinary Optimization*, (Dalian, China), May 2001. In Press.

[32] W. Duch and J. Korczak, "Optimization and global minimization methods suitable for neural networks," *Neural Computing Surveys*. submitted.

[33] Z. Michalewicz, *Genetic algorithms + data structures = evolution programs*. Berlin, Germany: Springer-Verlag, 1992.

[34] J.-M. Renders and S. P. Flasse, "Hybrid methods using genetic algorithms for global optimization," *IEEE Trans. Syst., Man, Cybern. B*, vol. 26, pp. 243–258, April 1996.

[35] R. Salomon, "Evolutionary algorithms and gradient search: Similarities and differences," *IEEE Trans. Evol. Comput.*, vol. 2, pp. 45–55, July 1998.

[36] J. A. Snyman and A. M. Hay, "The dynamic-Q optimization method: An alternative to SQP," in *Proc. of the Int. Workshop on Multidisciplinary Design Optimization* (J. Snyman and K. Craig, eds.), pp. 163–172, Univ. Pretoria, Dept. Mech. Eng., 2000.

[37] J. A. Snyman and A. M. Hay, "The spherical quadratic steepest descent (SQSD) method for unconstrained minimization with no explicit line searches," *Computers and Mathematics with Applications*, vol. 42, no. 1–2, pp. 169–178, 2001.

[38] J. R. Koza, F. H. Bennet III, D. Andre, and M. A. Keane, *Genetic programming III*. San Francisco, USA: Morgan Kaufmann, 1999.

[39] X. Yao, "A review of evolutionary artificial neural networks," *Int. Journal Intelligent Systems*, vol. 8, pp. 539–567, 1993.

[40] J. H. Holland, *Adaptation in natural and artificial systems*. Ann Arbor, USA: The University of Michigan Press, 1975.

[41] D. Whitley, "A genetic algorithm tutorial," Tech. Rep. CS-93-103, Colorado State Univ., November 1993.

[42] T. Bäck, U. Hammel, and H.-P. Schwefel, "Evolutionary computation: Comments on the history and current state," *IEEE Trans. Evol. Comput.*, vol. 1, pp. 3–16, April 1997.

[43] T. Bäck, "Selective pressure in evolutionary algorithms: A characterization of selection mechanisms," in *Proc. 1st IEEE Conf. on Evolutionary Computation*, (Piscataway, USA), pp. 57–62, IEEE Press, 1994.

[44] D. E. Goldberg and K. Deb, "A comparative analysis of selection schemes used in genetic algorithms," in *Foundations of Genetic Algorithms*, pp. 69–93, San Mateo, USA: Morgan Kaufmann, 1991.

[45] E. H. L. Aarts and J. Korst, *Simulated annealing and Boltzmann machines: A stochastic approach to combinatorial optimization and neural computing.* John Wiley and Sons, 1998.

[46] K.-S. Leung, Q.-H. Duan, Z.-B. Xu, and C. K. Wong, "A new model of simulated evolutionary computation - convergence analysis and specifications," *IEEE Trans. Evol. Comput.*, vol. 5, pp. 3–16, February 2001.

[47] D. E. Goldberg, B. Korb, and K. Deb, "Messy genetic algorithms: Motivation, analysis, and first results," *Complex Systems*, vol. 3, no. 5, pp. 493–530, 1990.

[48] D. E. Goldberg, K. Deb, and B. Korb, "Messy genetic algorithms revisited: Studies in mixed size and scale," *Complex Systems*, vol. 4, no. 4, pp. 415–444, 1990.

[49] S. V. Marshall and G. G. Skitek, *Electromagnetic concepts and applications.* Englewood Cliffs, USA: Prentice-Hall, third ed., 1990.

[50] M. A. Hamid and M. M. Yunik, "On the design of stepped transmission-line transformers," *IEEE Trans. Microwave Theory Tech.*, vol. 15, pp. 528–529, September 1967.

[51] G. N. French and E. H. Fooks, "Double section matching transformers," *IEEE Trans. Microwave Theory Tech.*, vol. 17, p. 719, September 1969.

[52] G. N. French and E. H. Fooks, "The design of stepped transmission-line transformers," *IEEE Trans. Microwave Theory Tech.*, vol. 16, pp. 885–886, October 1968.

[53] P. I. Somlo, "A logarithmic transmission line chart," *IEEE Trans. Microwave Theory Tech.*, vol. MTT-8, p. 463, July 1960.

[54] R. M. Arnold, "Transmission line impedance matching using the Smith Chart," *IEEE Trans. Microwave Theory Tech.*, vol. 22, pp. 977–978, November 1974.

[55] P. I. Day, "Transmission line transformation between arbitrary impedance using the Smith Chart," *IEEE Trans. Microwave Theory Tech.*, vol. 23, pp. 772–773, September 1975.

[56] H. W. Bode, *Network analysis and feedback amplifier design.* New York: Van Nostrand, 1945.

[57] R. M. Fano, "Theoretical limitations on the broadband matching of arbitrary impedances," *J. Franklin Inst.*, vol. 249, pp. 57–83, January 1950.

[58] R. M. Fano, "Theoretical limitations on the broadband matching of arbitrary impedances," *J. Franklin Inst.*, vol. 249, pp. 139–154, February 1950.

[59] D. C. Youla, "A new theory of broad-band matching," *IEEE Tras. Circuit Theory*, vol. CT-11, pp. 30–50, March 1964.

[60] W.-K. Chen and C. Satyanarayana, "General theory of broadband matching," *IEE Proc. Pt. G*, vol. 129, pp. 96–102, June 1982.

[61] G. L. Matthaei, "Tables of Chebyshev impedance transforming networks of low-pass filter form," *Proc. IEEE*, pp. 939–953, August 1964.

[62] E. G. Cristal, "Tables of maximally flat impedance-transforming networks of low-pass-filter form," *IEEE Trans. Microwave Theory Tech.*, vol. MTT-13, pp. 693–695, September 1965.

[63] W.-K. Chen, "Explicit formulas for the synthesis of optimum broad-band impedance-matching networks," *IEEE Trans. Circuits Syst.*, vol. CAS-24, pp. 157–169, April 1977.

[64] W.-K. Chen and K. G. Kourounis, "Explicit formulas for the synthesis of optimum broad-band impedance-matching networks II," *IEEE Trans. Circuits Syst.*, vol. CAS-25, pp. 609–620, August 1978.

[65] O. Pitzalis and R. A. Gilson, "Tables of impedance matching networks which approximate prescribed attenuation versus frequency slopes," *IEEE Trans. Microwave Theory Tech.*, vol. MTT-19, pp. 381–386, April 1971.

[66] W. H. Ku and W. C. Petersen, "Optimum gain-bandwidth limitations of transistor amplifiers as reactively constrained eactive two-port networks," *IEEE Trans. Circuits Syst.*, vol. CAS-22, pp. 523–533, June 1975.

[67] R. E. Collin, "Theory and design of wide-band multisection quarter-wave transformers," *Proc. IRE*, vol. 43, pp. 179–185, February 1955.

[68] S. B. Cohn, "Optimum design of stepped transmission-line transformers," *IRE Trans. Microwave Theory Tech.*, pp. 16–21, April 1955.

[69] H. J. Riblet, "General synthesis of quarter-wave impedance transformers," *IRE Trans. Microwave Theory Tech.*, vol. 5, pp. 36–43, January 1957.

[70] L. Young, "Stepped-impedance transformers and filter prototypes," *IRE Trans. Microwave Theory Tech.*, pp. 339–359, September 1962.

[71] L. Young, "Tables for cascaded homogeneous quarter-wave transformers," *IRE Trans. Microwave Theory Tech.*, pp. 233–237, April 1959.

[72] C. S. Gledhill and A. M. H. Issa, "Exact solutions of stepped impedance transformerrs having maximally flat and Chebyshev characteristics," *IEEE Trans. Microwave Theory Tech.*, vol. MTT-17, pp. 379–386, July 1969.

[73] F.-C. Chang and H. Mott, "Exact design of stepped-impedance transformers," *IEEE Trans. Microwave Theory Tech.*, vol. MTT-20, pp. 620–321, September 1972.

[74] P. I. Richards, "Resistor-transmission-line circuits," *Proc. IRE*, vol. 36, pp. 217–220, February 1948.

[75] H. J. Carlin and W. Kohler, "Direct synthesis of band-pass transmission line structures," *IRE Trans. Microwave Theory Tech.*, pp. 283–297, May 1965.

[76] P. W. van der Walt, "Short-step-stub Chebyshev impedance transformers," *IEEE Trans. Microwave Theory Tech.*, vol. MTT-34, pp. 863–868, August 1986.

[77] G. L. Matthaei, "Short-step Chebyshev impedance transformers," *IEEE Trans. Microwave Theory Tech.*, vol. MTT-14, pp. 372–383, August 1966.

[78] R. Levy, "Synthesis of mixed lumped and distributed impedance-transforming filters," *IEEE Trans. Microwave Theory Tech.*, vol. MTT-20, pp. 223–233, March 1972.

[79] R. Levy, "A generalized design technique for practical distributed reciprocal ladder networks," *IEEE Trans. Microwave Theory Tech.*, vol. MTT-21, pp. 519–526, August 1973.

[80] J. M. Drozd and W. T. Joines, "Using parallel resonators to create improved maximally flat quarter-wavelength transformer impedance-matching networks," *IEEE Trans. Microwave Theory Tech.*, vol. 47, pp. 132–141, February 1999.

[81] H. J. Carlin and J. J. Komiak, "A new method of broad-band equalization applied to microwave amplifiers," *IEEE Trans. Microwave Theory Tech.*, vol. MTT-27, pp. 93–99, February 1979.

[82] H. Dedieu, C. Dehollain, J. Neirynck, and G. Rhodes, "A new method for solving broad band matching problems," *IEEE Trans. Circuits Syst.-I: Fund. Theory Appl.*, vol. 41, pp. 561–571, September 1994.

[83] K. Madsen, O. Nielsen, H. Schjær-Jacobsen, and L. Thrane, "Efficient minimax design of networks without using derivatives," *IEEE Trans. Microwave Theory Tech.*, vol. MTT-23, pp. 803–809, October 1975.

[84] J. W. Bandler and P. A. MacDonald, "Optimization of microwave circuits by razor search," *IEEE Trans. Microwave Theory Tech.*, vol. MTT-17, pp. 552–562, August 1969.

[85] J. W. Bandler, T. V. Srinivasan, and C. Charalambous, "Minimax optimization of networks by grazor search," *IEEE Trans. Microwave Theory Tech.*, vol. MTT-20, pp. 596–604, September 1972.

[86] S. Mahdi and A. B. Macnee, "Dominant pole synthesis of transmission line networks," *IEEE Trans. Microwave Theory Tech.*, vol. MTT-17, pp. 591–597, August 1969.

[87] B. S. Yarman and A. Aksen, "An integrated design tool to construct lossless matching networks with mixed lumped and distributed elements," *IEEE Trans. Circuits Syst.–I: Fund. Theory Appl.*, vol. 39, pp. 713–723, September 1992.

[88] M. H. Gibson, "A distributed circuit-design technique for the design of broadband, low-noise multistage amplifiers," *ESA Journal*, vol. 8, no. 3, pp. 275–286, 1984.

[89] J. Pandel and A. Fettweis, "Numerical solution to broadband matching base on parametric representations," *AEÜ*, vol. 8, no. 4, pp. 202–209, 1987.

[90] R. L. Haupt, "An introduction to genetic algorithms for electromagnetics," *IEEE Antennas Propagat. Mag.*, vol. 37, pp. 7–15, April 1995.

[91] D. S. Weile and E. Michielssen, "Genetic algorithm optimization applied to electromagnetics: A review," *IEEE Trans. Antennas Propagat.*, vol. 45, pp. 343–353, March 1997.

[92] J. M. Johnson and Y. Rahmat-Samii, "Genetic algorithms in engineering electromagnetics," *IEEE Antennas Propagat. Mag.*, vol. 39, pp. 7–25, August 1997.

[93] T. Günel, "A new approach for the synthesis of nonreciprocal and nonreciprocal transmission line impedance matching sections," *AEÜ*, vol. 52, no. 4, pp. 274–276, 1998.

[94] T. Günel, "A hybrid approach to the synthesis of nonuniform lossy transmission-line impedance-matching sections," *Microwave and Optical Tech. Lett.*, vol. 24, pp. 121–125, January 2000.

[95] J. M. Bornholdt, "Computational electromagnetic exploration of B-spline shaped microwave impedance matching circuits using evolutionary algorithms and information theory," in *1999 IEEE MTT-S Int. Microwave Symp. Dig.*, pp. 699–702, 1999.

[96] A. Raychowdhury, B. Gupta, and R. Bhattacharjee, "Bandwidth improvement of microstrip antennas through a genetic-algorithm-based design of a feed network," *Microwave and Optical Tech. Lett.*, vol. 27, pp. 273–275, 20 November 2000.

[97] Y. Sun and W. K. Lau, "Evolutionary tuning method for automatic impedance matching in communciations systems," in *IEEE Int. Conf. Electonics, Circuits Syst.*, vol. 3, pp. 73–77, 1998.

[98] Y. Sun and W. K. Lau, "Antenna impedance matching using genetic algorithms," in *IEE National Conf. Antennas Propagat.*, pp. 31–36, 1999.

[99] R. K. Hoffmann, *Handbook of microwave integrated circuits.* Boston, USA: Artech House, 1987.

[100] K. C. Gupta, R. Garg, I. J. Bahl, and P. Bhartia, *Microstrip lines and slotlines.* Boston, USA: Artech House, second ed., 1996.

[101] *HP-EEsof microwave and RF circuit design*, vol. 1. Hewlett-Packard, February 1994. HP Part no. E4605-90009.

[102] E. Hammerstad and O. Jensen, "Accurate models for microstrip computer-aided design," in *1980 IEEE MTT-S International Microwave Symposium Digest*, (Washington), pp. 407–409, 1980.

[103] M. Kirschning and R. H. Jansen, "Accurate model for effective dielectric constant of microstrip with validity up to millimetre-wave frequencies," *Electron. Lett.*, vol. 18, pp. 272–273, 18 March 1982.

[104] S. March, "Microstrip packaging: Watch the last step," *Microwaves*, pp. 83–94, December 1981.

[105] E. J. Denlinger, "Losses of microstrip lines," *IEEE Trans. Microwave Theory Tech.*, vol. MTT-28, pp. 513–522, June 1980.

[106] M. Kirschning, R. H. Jansen, and N. H. L. Koster, "Accurate model for open end effect of microstrip lines," *Electron. Lett.*, vol. 17, pp. 123–125, 5 February 1981.

[107] M. E. Goldfarb and R. A. Pucel, "Modeling via hole grounds in microstrip," *IEEE Microwave and Guided Wave Lett.*, vol. 1, pp. 135–137, June 1991.

[108] K. C. Gupta, R. Garg, and I. J. Bahl, *Microstrip lines and slotlines.* Dedham, USA: Artech House, 1979.

[109] E. Hammerstad, "Computer-aided design of microstrip couplers with accurate discontinuity models," in *1981 IEEE MTT-S International Microwave Symposium Digest*, (Trondheim), pp. 54–56, 1981.

[110] R. Garg and I. H. Bahl, "Microstrip discontinuities," *Int. J. Electronics*, vol. 45, pp. 81–87, July 1978.

[111] G. Klir and B. Yuan, *Fuzzy sets and fuzzy logic: Theory and applications.* New York, USA: Prentice-Hall, 1995.